POSTER ABSTRACT

A Web-page Fragmentation Technique for Personalized Browsing

Bouras Christos Kapoulas Vaggelis Misedakis Ioannis

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

+30-2610-960375 +30-2610-960355 +30-2610-996954

bouras@cti.gr kapoulas@cti.gr misedaki@cti.gr

ABSTRACT

In this paper, a technique is presented that allows web sites viewers to build personalized web pages, using specific thematic areas of their preferred sites. This technique, besides saving from the trouble of having to browse in different sites in order to find the desired content, saves users time and reduces the cost of browsing the web by minimizing the data that have to be downloaded. It is based on an algorithm, which fragments a web page in discrete fragments using the page's internal structure. A training and update procedure is utilized for recognizing the instances of the web page components in different time points.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information Filtering*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based Services*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Web Components, Web Fragments, Portal Personalization.

1. INTRODUCTION

Most web sites have a static structure for the presentation of their content. This structure rarely changes, even if the content of the web site changes very often. In content-rich web sites this structure comprises of areas of content of common semantic. These areas are called 'Web Components'.

In this paper we present a technique that could assist the Web users in their browsing sessions by building 'personalized pages' containing content from the users' favorite sites. This technique premises the usage of a software tool that works centrally (as a data source for the web server) and which analyzes selected web pages and fragments them in the thematic areas they are composed of. Web Components (WC) are extracted from a web page by parsing the HTML code, identifying the parts of the code that belong in WCs and retrieving these parts of code as independent entities.

'Web Components' was introduced as a concept in [1]. Fragmentation of web pages and manipulation (transcoding) of the fragments has been applied also in numerous systems that intend to offer WWW services to handheld devices, such as PDAs and mobile phones.

2. FRAGMENTATION ALGORITHM

A browser renders a web page based on its HTML code. The tags inside the HTML file are nested, and can be represented as a tree (HTML tree). The parts of the page that represent the different Web Components can be extracted by extracting some particular nodes of the HTML tree.

Most of the web sites use tables for building their layout, which lead to the decision to use the nested table structure of a web page as the leading criterion for its fragmentation. If we ignore all the tags (nodes) of the HTML tree except the TABLE tags, the HTML tree is reduced significantly in complexity. The algorithm uses this reduced tree ('index tree') to make the calculations for the fragmentation of the page. The fragmentation algorithm is used for the *web pages' analysis and fragmentation*, which includes two phases: training and update. The steps of the fragmentation algorithm are presented in the procedure below:

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

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. In case there are differences, execute the fragmentation correction algorithm.
6) Extract the Web Components from the HTML tree and store them.

Step 3 of the fragmentation algorithm takes as input the HTML tree and constructs the reduced tree ('index tree'), which is used

SAC'04, March 14–17, 2004, Nicosia, Cyprus

in step 4 for recognizing the Web Components of the page. Each node in the index tree has a link to its corresponding node in the HTML tree and also stores some additional information about the node.

The decisions about the fragmentation of a web page are taken in step 4. The algorithm traverses the index tree, searching for nodes that match some criteria. When a node matching those criteria is found, the algorithm stops traversing its children and the node is marked as a Web Component. The fragmentation criteria are related to the content size of each node and its internal structure.

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) of the node. If node *p* meets the following criterion then it is marked as a Web Component without even examining its internal structure:

$$l \leq Ratio_p * (Number\ of\ Content\ Nodes) \leq u \ (1)$$

where $0 \leq l \leq u \leq u_{max} = (Number\ of\ Content\ Nodes)$ and

$$Ratio_p = \frac{Pure\ Text\ Length\ in\ the\ node\ p}{Pure\ Text\ Length\ of\ the\ root\ node}$$

Relation 1 expresses the intuitive criterion that a Web Component must be ‘medium’-sized, in comparison with the whole page size. Ratio_p is calculated by dividing the pure text length included in node p by the text length of the entire page, giving the percentage of the node’s content to the content of the whole page. This expresses the relative size of the Web component (regarding the size of the whole page). The values of *l* and *u* express the lower and upper bound for the length of a node’s text in order to be considered ‘medium-sized’. Relation 1 means that if a node’s text length is greater or equal than l/u_{max} and smaller or equal than u/u_{max} of the whole page text length, then this node is considered ‘medium-sized’ and is selected as a Web Component. The values that are used for the constants *l* and *u* are $l=1$ and $u=2$.

The other major criterion for fragmenting a web page is based only on the structure of the index tree. The areas that intuitively are perceived as Web Components are usually composed of more then one TABLE tags, one of which contains the main body of the Component’s content, while the others are layout tags or tags with insignificant amount of content. So, when the fragmentation algorithm finds out one node of the index tree that contains *less than four children* and *less than five (in total) descendants* (not including layout nodes) it selects this node as a Web Component. This criterion helps ‘refine’ the results of the criterion that is based on the content’s size.

The algorithm includes two more steps, which are used only in the update phase.

3. PROPOSED TECHNIQUE

In this section the methodology for constructing personalized web pages based on Web Components is presented.

A training and update procedure is used for overcoming the problems created by major changes in the structure of web pages. The training and update procedure is the most complex issue of the presented technique, but due to limited space we cannot present it here in details.

The training phase analyzes the web page and decides how the Web page will be fragmented and how its Web Components will be selected during the update phase. In the end of the training phase, the training algorithm’s output is a vector containing a *unique identifier (signature)* for each one of the Web Components of the page. This signature is used for identifying a Web Component in a web page instance that has changes in the page structure or changes in the number of the Web Components. It uniquely diversifies it from all the other WC of the page.

Web Components can be classified in three categories, based on the *changes of their content*: There are some WC that their content never changes, there are some others that their entire content changes and there is a third category of components that some part of their content changes, while another part remains constant. The training phase uses the constant part of the components content for the first and third categories in order to assign a unique identifier for them, while it uses the relative position and the average content size of the components for the third category.

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 web components of the web page. Following this, it stores the latest instances of the web components in the Web Server of the system, in order to be used by the users for their personalized portals’ creation. If major changes in a web page result to an index tree with different structure or the fragmentation algorithm fails to correctly recognize the instances of the Web Components that were identified during the training phase, a special ‘fragmentation correction algorithm’ is utilized in order to extract the expected instances.

4. FUTURE WORK - CONCLUSIONS

Future work plans include examining situations where the training or the update procedure fail and trying to improve the algorithms. Also, the fragmentation algorithm can be further improved by using more advanced heuristics.

Concluding, in this paper we presented the concept of ‘Web Components’ 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 sites that they are interested in.

5. REFERENCES

[1] C. Bouras and A. Konidaris, “Web Components: A Concept for Improving Personalization and Reducing User Perceived Latency on the World Wide Web”, Proceedings of the 2nd International Conference on Internet Computing (IC2001), Las Vegas, Nevada, USA, June 25th - 28th 2001, Vol 2, pp.238-244