Advances in Computer Science Research (ACRS), volume 54
International Conference on Computer Networks and Communication Technology (CNCT2016)

Authoring of Personalized Web Page from Heterogeneous Web Pages by Content Extraction and Integration

Wei-gang LI*, Ke SUN and Shuo-chen WANG

School of Software and Electronics, Northwestern Polytechnical University, Xi'an, Shaanxi, China

*Corresponding author: liweigang@nwpu.edu.cn

Keywords: Authoring of Web pages, Content extraction, Element similarity, CS-DOM tree.

Abstract. Authoring of personalized Web page by integrating heterogeneous Web page elements from different sites is a challenging task in Web 2.0 applications. An approach to extract various of partitions or elements, which can be the basic HTML elements, CSS definitions, JavaScript source code, etc, from different Web sites, thus implementing authoring of new page from heterogeneous Web pages is proposed in this paper. A novel DOM tree model, CS-DOM tree, is introduced to retrieve the CSS definitions. In order to assure that the new Web pages keep updating synchronized with the source pages, a method based on the structure of DOM and the context of elements to relocate the elements that have been retrieved before is then presented. The similarity calculation algorithm used to judge whether the relocated elements and the elements retrieved before are from the same position is developed at last. The method proposed in this paper has been applied to develop a personalized portal.

Introduction

In Web 2.0 applications the users are not only information consumers, but also information producers. So, allowing the reader to author Web pages and create personalized portals is necessary for Web 2.0. This requires the retrieval of information from heterogeneous Web pages. Normally, information retrieval of Web pages comes in two situations. One is the search engines, which classifies Web pages into multiple kinds of topics. When searching certain topics, the search engines can get relevant pages and then shows them to the users. The other is noises cleaning, which divides Web page into several blocks and identify whether a block is noise such as the navigation panels, copyright and advertisements, and then eliminates them from the Web page.

In the above two situations, the target Web pages are handled as a whole. But with the requirement of giving more personalized services by the Web 2.0 applications, the integration of certain parts that belong to the different Web pages into new pages is getting more and more important. Because a part of a Web page corresponds to some elements in the HTML, the element level of Web pages must be focused in this situation. It is required to wave these fine grained elements into the personalized Web contents to meet the needs of individual users.

The element level of authoring of Web pages mainly reflects in the extraction of the HTML tags and their attributes, CSS information, and JavaScript source code in the elements. Besides, the system should update the new pages to ensure their synchronization with the pages where the elements of the new page come from. This paper will focus on the solution of these problems.

Related Work

Content Extraction

There have been lots of approaches proposed in the field of information extraction and noises cleaning. Wu et al. design a series of tag path extraction features to distinguish the Web content and noise from different perspectives, and propose a features fusion strategy with group feature selection based on the similarity analysis of these features. They apply this method to online Web news extraction [1]. Krishna et al. assume that pages of the same Website are constructed by the same template. Then they find schema that is the structure of Web pages and extract data [2]. Li et al. use tags, such as <body>, and <table>, as the standard of classification of the Web pages, as they believe that these tags fully reflect the visual effects of pages. So, different values are assigned to each tag and the total value is calculated which is used to decide whether a block is noise [3]. C. Kim and K. Shim propose a novel algorithm to extract template from numerous documents constructed by heterogeneous template [4].

The above approaches are used to extract the useful content of Web pages and eliminate the noises instead of extract the content of a certain element or their children elements. The traditional DOM tree doesn't support the retrieval of CSS information. For a Web page, CSS can be defined in external files and <style> tags as well. In this paper, we propose the CS-DOM tree to get the complete CSS information of certain elements so that the elements as well as their CSS information can be reused in the authoring of the new Web pages.

Synchronization between Source Pages and the Generated Pages

As the Web is dynamic and the content of Web pages keep changing all the time, it is necessary for the authoring personalized Web pages to make sure that the new pages keep synchronization with the source pages. This problem can be transformed into how to relocate each element in the source pages and how to judge the relocated elements is similar to the elements extracted before.

A XPATH-based relocation method to extract the content of the same position in pages is proposed in [5]. It is based on structure matching. However, it could only be applied to pages from the same Web sites as it assumes that Web pages always keep in the same structure. The DEIP algorithm is proposed in [6], which creates an index path for each text node. When users input keywords, the system will use the index path to iterate each text node to find the most optimal matching. All the above algorithms assume that pages in a Web site are constructed in the same template. Actually, it is not always the case. Not all the Web sites use templates to construct pages and not all pages from a Web site share the same structure.

Kim et al. put forward the HTMLTree Matching algorithm to calculate the similarity of two elements, which is based on the edit distance algorithms. This method doesn't take into consideration that different tags and same tags in different position should have different weights.

Dynamic Page Integration Procedures

Definition 1. The source page is the page where the elements that are to be integrated come from.

Definition 2. The integrated page is authored by elements from different source pages.

Definition 3. The top-level tag of a block refers the root node of this block's DOM tree.

In the personalized Web page authoring system, the elements needed to be integrated should be selected by users. In such cases, human-machine interaction is a must in this kind of system. After the integration page is formed, it should keep pace with its source pages.

The procedure of element integration from web pages is shown in Fig. 1. The element information is comprised of element content and relocation information, as shown in Fig. 2. Besides, similarity calculating will be performed after relocation in order to judge whether the relocated elements are actually what we need.

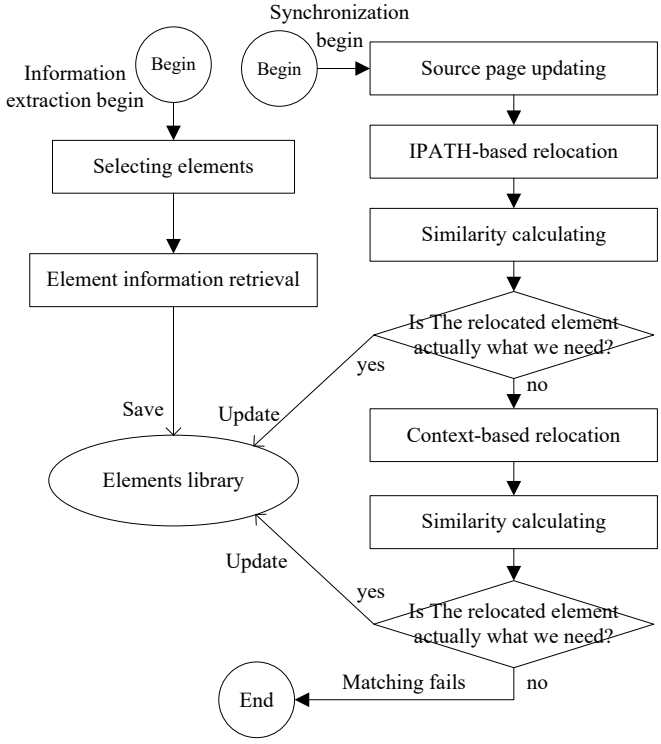

Figure 1. Procedure of element integration from Web pages

As the reuse of JavaScript source code cannot be achieved, this paper focuses on the retrieval of CSS information, the relocation of elements and similarity calculating algorithm. In later sections we will discuss the three problems in detail.

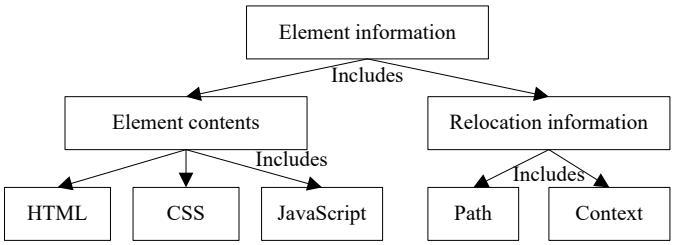

Figure 2. Composition of element information

The Approach

CSS Information Retrieval

The extraction of main content of Web page has long been researched. Meanwhile few researches have been done to extract the CSS information of elements in the Web pages.

CSS extraction is necessary when users want to get the presentation of the element besides the semantic information in it. In this paper, we propose CS-DOM tree to retrieve the CSS information.

Definition 4. CS-DOM tree is based on the DOM tree and adds the CSS information from external CSS files and/or <style> tags into the attribute sets of DOM nodes. The integration of CSS information is reflected by a collection consisting of multiple CSS class instances. And this collection acts as an element of the attribute set.

Only the CSS information of elements that are selected by users should be saved. So the procedure of construction of CS-DOM tree is shown as algorithm 1.

Algorithm 1. Generating the CS-DOM tree:

Step1. Construct the DOM tree in traditional way.

Step2. After users select certain elements, the system iterates all the external CSS files to match the tag names, the class attribute and the id of tags of the selected nodes. If matching succeeds, then create a CSS class instance and add it into the collections of these nodes.

Step3. Retrieve the CSS information definition in <style> tags and add it into the collections as well. If there are reduplications, then the CSS attributes defined in <style> tags have the priority and rewrite these from external CSS files.

Step4. Retrieve the CSS information in nodes, which have top priority and can cover that retrieved in <style> tags and external CSS files.

Step5. Save the above information into the database.

Element Relocation Algorithms

When the data of source pages change, the authored pages should update to keep pace with their source pages. In such cases, to relocate the elements in the source pages is a must in order to judge whether changes have happened.

Some researchers adopt the XPATH technique to record the position of elements based on DOM [1, 5]. This method could solve problems when the target Web site uses the same template to construct pages and the pages' structure always remains the same. Obviously, not all Web sites use templates and the structure of pages in one Web site can be diverse greatly. Besides, a tiny change in one Web page may lead to a wrong relocation of an element.

We propose an element relocation algorithms based on the structure and the context of the DOM tree. As shown in Fig. 1, we firstly adopt the IPATH (Index Path [6]) structure-based method to relocate the elements. To ensure the correctness of relocation, similarity calculation will be applied between the elements retrieved by the IPATH method and these retrieved before. If the similarity is below a certain threshold, then context-based relocation algorithm will be applied and another similarity calculation will be adopted for the same purpose. Detailed descriptions of this method are shown as followings.

Structure-based Relocation

We adopt the Index Path algorithm proposed in [6]. However, there are differences. Paper [6] creates Index Path for all the text nodes in DOM so that it could iterate all the text nodes in a rapid speed to match the content to be searched. Meanwhile, we only create Index Path for the top-level tag of the element. When source pages update their content, the Index Path will be used to relocate the elements in source pages and retrieve the content again, thus ensuring that the integrated pages keep pace with the source pages.

Definition 5. Assuming the path of node *T* in DOM tree is $root \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots \rightarrow t_n \rightarrow T$, and that the above nodes are the *i*-th child nodes of their fathers, then the IPATH of node T is: $root \rightarrow t_1[i_1] \rightarrow t_2[i_2] \rightarrow \ldots \rightarrow t_n[i_n] \rightarrow T$.

Context-based Relocation

Let *TF* and *TL* be the starting and ending tag of node *T* respectively. When retrieving information from a certain block of Web pages, the two tags right before *TF*, named *P1* and *P2*, and the two tags right after *TL*, named *N1* and *N2*, should be saved into the database as well as *TF* and *TL* themselves. When applying the context-based relocation algorithm, strings formed by part of these tags will be used to do pattern matching. As there are 6 tags, the number of different combinations is $\sum_{i=1}^{6} C_6^i$. Taking the importance of each tag and the acceptable number of tags into consideration, we define the following combination as acceptable matching patterns.

P1, *P2*, *TF* and *TL*, *N1*, *N2*
P1, *P2*, *TF*, *TL* or *TF*, *TL*, *N1*, *N2*
P1, *P2*, *TF* or *TL*, *N1*, *N2*
TF, *TL* or *P2*, *TF* or *TL*, *N1*

Thus we can relocate the elements by the above matching patterns in turn. If there are elements that match, element similarity will be operated to judge whether they are from the same position. If not, the next matching pattern will be used until all the matching patterns are employed.

Similarity Calculation between Two Elements

Based on the tree edit distance algorithm, HTMLTreeMatching algorithm is proposed to calculate the similarity of two elements [7]. This algorithm first calculates the weight of each tag by assigning different contents with different mapping values. Assume that the value of the root tag is 1, each tag's value is calculated. The similarity is then defined as the ratio between the total value of all the same tags in the two elements and the average value of all the tags' value in the two elements, just as in Eq.1.

$$Sim_{html}(T_a, T_b) = \frac{HTMLTreeMatching(T_a, T_b)}{AVG(V_{all}(T_a) + V_{all}(T_b))} \quad (1)$$

This algorithm doesn't consider that different tags and same tags located in different layers of DOM tree should have different mapping weights. For example, the mapping value of text node is calculated as Eq. 2.

$$\inf oValue(text) = B(text) * fs(text) \quad (2)$$

Where *B*(*text*) stands for the number of bytes occupied by the text content and *fs*(*text*) stands for the size of these fonts. According to these definitions, the text <h1>hello</h1> should have the same mapping values as <h2>hello</h2> , for they share the same text content. However, as the first text content is surrounded by <h1> and in higher layer than the second one, the first text content should have greater weight than the second. So this algorithm needs to be optimized before applying it.

Because different tags influence the visual effect of the text in varying degrees, we give a sequence of multiple tags according to their impact on the representation:

H1>H2>…>H6>P, DIV, DL, CENTER, UL, OL, DTR, TABLE>LT, DT, CAPTION>DD, TR>TH, TD

According to the sequence, we assign different mapping weights to different tags, which are marked as W_i (i=1, 2, 3,…,n>2).

Usually, the higher layer a tag is located in, the more important the tag is. We define that the serial number of the top-level tag is 0. The mapping weight of tags in the first layer is W_i. The mapping weight of tags in the second layer is $W_i/2$. Meanwhile, the rest tags share the same mapping weight, which is 1.

In such cases, if the layer a tag is located in is greater than 2, then the algorithm of calculating its mapping value remains the same as before, or its calculation follows the following Eq. 3 and Eq. 4. (The symbol W_i shown below stands for the final mapping weight of tags, which have considered the tags' location level)

For images:

$$\inf oValue(image) = \frac{width * height * W_i}{\partial} (\partial > 0) \tag{3}$$

For text content:

$$\inf oValue(text) = B(text) * fs(text) * W_i \tag{4}$$

By introducing the mechanism that different tags and even the same tags located in different layers have different mapping weights, we can better evaluate each tag's final value and weights.

Conclusion

To fully satisfy users' demand for making their own personalized Web page from the existed heterogeneous Web pages, this paper proposes the methods of retrieving the contents of the elements in pages and updating the customized personal Web page synchronously. It has been proved by experiments that the first method can effectively extract the HTML and CSS information of the elements in pages and that the difficulty of updating synchronously can be solved by combining the similarity calculation algorithm with the structure and context based relocation algorithm. The optimization of the algorithm in detail will be further researched.

Acknowledgement

This work is supported by the Science and Technology Program of Shaanxi Province, China (No. 2014K09-21).

References

[1] G.Q. Wu, J. Hu, L. Li, et al, Online Web news extraction via tag path feature fusion, Journal of Software, 2016, 27(3):714-735 (in Chinese).

[2] S.S. Krishna, J.S. Dattatraya, Schema inference and data extraction from templatized Web pages, International Conference on Pervasive Computing, 2015.

[3] Y.C. Li, J. Yang, A novel method to extract informative blocks from Web pages, International Joint Conference on Artificial Intelligence, 2009.

[4] C. Kim, K. Shim, Text: Automatic template extraction from heterogeneous Web pages, IEEE Transactions on Knowledge and Data Engineering, 2010.

[5] Y. Fu, D.Q. Yang, S.W. Tang, T.J. Wang, J. Gao, Using XPATH to discover informative content blocks of Web pages, Third International Conference on Semantics, Knowledge and Grid, 2007.

[6] Y. Gao, F. Yuan, M. Zhang, Data extraction based on index path in Web, International Workshop on Education Technology and Computer Science, 2010.

[7] Y. Kim, J. Park, T. Kim, J. Choi, Web information extraction by HTML tree edit distance matching, International Conference on Convergence Information Technology, 2007.