Structuring Web pages based on Repetition of Elements

Tomoyuki NANNO

Interdisciplinary Graduate School
of Science and Engineering,
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku,
Yokohama, JAPAN
nanno@lr.pi.titech.ac.jp

Suguru SAITO

Precision and Intelligence Laboratory,
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku,
Yokohama, JAPAN
suguru@pi.titech.ac.jp

Manabu OKUMURA

Precision and Intelligence Laboratory,
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku,
Yokohama, JAPAN
oku@pi.titech.ac.jp

Abstract

The World Wide Web is a vast source of information accessible to computers, but most of its information is not able to be processed by computer applications because Web pages are described in layout description languages, such as HTML. In this paper, we propose a method of automatically segmenting and structuring Web pages based on repetition of elements.

1. Introduction

The World Wide Web is a vast source of information accessible to computers, but most of its information is not able to be processed by computer applications because Web pages are described in layout description languages, such as HTML. Thus, in recent years, there have been efforts to provide computers with a means of processing (rather than simply displaying) the information on the Web (e.g. Semantic Web and Web API, such as Google APIs[1]). For computers to be able to use the Web effectively, the introduction of metadata (meta information) will be inevitable. The structure of Web pages can be considered to be metadata. Therefore, automatic segmenting and structuring of Web pages would be helpful when making annotations of metadata.

When people see a Web page, they can easily understand the segmentation and structure of the page. What is the key to understanding the segment and structure? We consider that it is the uniformity of certain information. For example, if some fragments of the Web page are described in the same font color and font size, they could belong to the same group. We consider that such a "uniformity" can be useful for detecting the repetition of elements in the Web page.

In this paper, we propose a method of automatically segmenting and structuring Web pages based on repetition of elements.

Knowledge of the structure of Web pages is useful for various applications. For example, a voice browser can read information on a Web page in a manner that reflects the structure of the page, and a system that presents a web page on a hand-held device's small display can use the structure for segmentation [2].

---

[1]Google Web APIs: http://www.google.com/apis/

Shop	Auctions, Autos, Classifieds, Real Estate, Shopping, Travel
Find	HotJobs, Maps, People Search, Personals, Yellow Pages
Connect	Chat, GeoCities, Greetings, Groups, Mail, Messenger, Mobile

Figure 1. A part of a Yahoo! page

2. Related Work

Chen et. al. [2] proposed a technique to browse a Web page on a device with a small display. To divide and reconstruct a Web page, they structure it by using the technique proposed by Yang et. al. [1].

[1] proposed a technique to structure a Web page by using the visual similarity of HTML content objects. However, they only deal with the "repetition structure without separators," mentioned in 3.1. Therefore, their heuristic-based segmentation technique by detecting frequent patterns cannot discover suitable segment boundaries in case where a Web page has the "repetition structure with separators." On the other hand, we consider both types of repetition structure and structure a Web page by detecting all possible repetition structures.

Yu et. al. [3] proposed a technique for efficient information retrieval using Web page segmentation. While their technique structures a Web page roughly in a top-down strategy, our method structures it in detail by using a bottom-up strategy.

3. Outline of Proposed Method

3.1. Repetition Structure

The example in Fig. 1 is a part of a Web page[2]. By simply looking at this page, we can understand that "Shop" and "Find" belong to the same level, and "Shop" has a substructure which consists of "Auctions", "Autos", and so on. How do we understand this so quickly? We consider that the author writes the Web page so that it can be understood easily. For example, when the author writes a segment containing multiple elements of the same type, s/he expresses them with the same font color and font size.

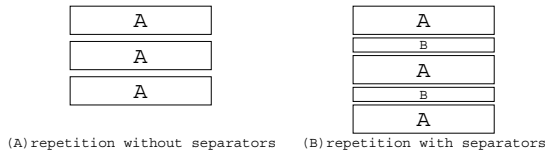In Fig. 1, the clue to understanding the segment and structure is that "Shop" and "Find" are expressed in the

---

[2]Yahoo! http://www.yahoo.com/

(A) repetition without separators   (B) repetition with separators

Figure 2. Two Types of Repetition Structure

Figure 3. Repetition Structures

same font, and "Shop" and "Auctions" are expressed in different fonts. We think that such uniformity can be useful for detecting the repetition of elements in the Web page. We call such a repetition of elements a "repetition structure." For example, "Auctions" and "Autos" are in the same repetition structure characterized by `<a>` tags.

Figure 2 shows the two types of repetition structure that we consider. The repetition structure on the left is without separators, and the one on the right includes separators. Please note that the fragment "Auctions, Autos, ..." in Fig. 1 has a type with separators, and the structure can be considered to appear frequently in a Web page.

3.2. Structuring by Detecting Repetition Structures Recursively

By detecting repetition structures, we can group the fragments which belong to the same level and detect a segment.

Therefore, we first detect the most primitive repetition structures (for example, the meshed parts shown in Fig. 3), and we replace them with tokens. At this time, if elements that are repeated are the same in the different repetition structures, those structures are replaced with the same token, even if the number of the elements is different. In Fig. 3, the meshed fragments are considered to be part of the same repetition structure and are replaced with identical tokens.

After replacing the detected repetition structures with tokens, by detecting repetition structures again, we can obtain larger repetition structures like in Fig. 4.

In this way, the Web page is structured in the bottom-up strategy by detecting repetition structures recursively.

4. System Configuration

Fig. 5 shows the flow chart of our system. Our system consists of the following three steps: "Preprocessing," "Segmentation and Structuring" and "Postprocessing."

4.1. Preprocessing

First, our system applies Tidy [4] to the HTML document in order to translate it into a well-formed XML document. This is done to confirm that begin tags and end tags are well-balanced in the parts of the HTML document.

Next, unessential parts of the HTML document, such as comments and scripts, are removed. The following tags are also deleted, because our system does not use these tags to detect repetition structures.
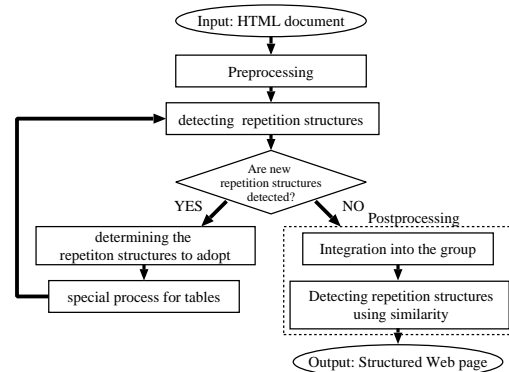

Figure 4. Structuring

Figure 5. Flow Chart of Our System

- `<br>`,`<nobr>`
- `<font>`,`<b>`,`<i>`,`<s>`,`<tt>`,`<u>`

Then, the tags used for modification of texts are made attributes of the texts. This is done to equate `<small><a>text</a></small>` with `<a><small>text</small></a>`.

Finally, all the texts except tags are replaced with tokens "(text)", because our system only uses the information that "there are some texts". Our current system does not use linguistic information at all, because we wish to develop a framework that is language-independent and universal. If we did otherwise, the system would have to handle many languages that are used in the WWW.

4.2. Segmentation and Structuring

This step consists of three substeps: "Detecting repetition structures," "Determining which repetition structures to adopt" and "Special process for tables."

Detecting Repetition Structures

In this step, our system detects all the repetition structures shown in Fig. 2 from the whole HTML document. Each repetition structure must satisfy the following restrictions:

- For repetition structures without separators

  - all the "A"s must be the same and must not contain repetition structures inside them.
  - all the tags contained in "A" must be well-balanced.

- For repetition structures with separators

  - all the "A"s must be the same and must not contain repetition structures inside them.
  - the elements contained in "B" must be the same before replacing them with tokens.
  - all the tags contained in "A" and "B" must be well-balanced.
  - the size of "B" must be smaller than "A."
  - "B" must not contain `<a>` tags.

8

Determining which Repetition Structures to adopt

After the above steps have been completed, there may be multiple detected repetition structures whose spans in the HTML document overlap with each other. Consider, for example, the following fragment:

A–B–A–B–A

This fragment can be considered as a repetition structure with separators if all "B"s satisfy the separator's restrictions. Similarly, we can consider this fragment as a repetition structure without separators ("AB"s are repetitions and the last "A" is a remainder.). To decide which repetition structure is more suitable it is inevitable to take into account the repetition structures which are detected in the adjoining fragments. Consider the following two case:

(1) A–B–A–B–A–C–D–C
(2) A–B–A–B–A–C–A–C

In case (1), it is reasonable to consider that there are two repetition structures with separators if all "B"s and "D" satisfy the separator's restrictions ("ABABA" and "CDC"). In case (2), on the other hand, it is reasonable to consider that there are two repetition structures without separators if all "A"s do not satisfy the separator's restrictions (even if all "B"s satisfy the separator's restrictions) ("ABAB" and "ACAC"). Thus, we have to select the best one from all possible repetition structures by taking into account the repetition structures detected in the adjoining fragments.

Since our system structures the Web page with a bottom-up strategy from the most primitive elements, the set of repetition structures whose total number of repetitions is the biggest is considered to be the best combination.

To find this combination, our system first builds a graph whose nodes correspond to the detected repetition structures. Then, two nodes that have no overlaps are connected by an edge and this process is repeated until every such pair has been connected. Our system can find the best combination by detecting all the maximal cliques [6] from this graph, and selecting the one whose total repetitions is the biggest.

After determining the repetition structures to adopt, the system groups them by replacing them as tokens. The repetition structures whose repeated elements are the same are replaced with identical tokens.

Special Process for Tables

Since tables have rich expressiveness, special processes are needed for certain kinds of tables. For example, consider

A	B
C	D
C	D
C	D

the table on the left. In this table, if "A" has the same type as "B" and "C" has the same type as "D", our system considers that this table should be read in the vertical direction. In this case, "A" and "CCC" should belong to the same group. However, since the order in the HTML document is "A"→"B"→"C"→"D", we cannot get such a structure. Therefore, our system transposes such a table.

A	A
A	A
A	A

Next, we illustrate another kind of table that needs special processes. For example, consider the table on the left. In this table, if all the "A"s have the same type and they also
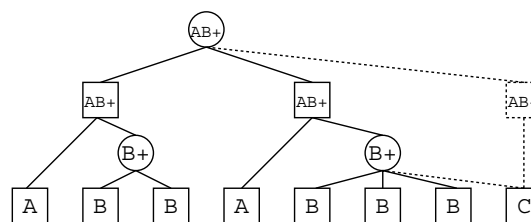

Figure 6. Integration into the group

have substructures or elements which have link attributes, our system considers that this table is used for layout purposes, such as for making two columns. Therefore, our system removes tags that show rows from such a table.

Building up the Structure

The above steps are repeated until new repetition structures are no longer detected. Since the spans where repetition structures exist are replaced with tokens, bigger repetition structures are detected as the procedure is executed.

4.3. Postprocessing

There are some cases where our system cannot structure the entire document, because the definition of the repetition structure is too strict (exact match). In this section, we explain the two postprocesses: "Integration into the group" and "Detecting repetition structures using similarity". Our system structures Web pages with a two-step approach: first by using "exact match" and then by using "similarity." The reasons why we do not introduce similarity in the first step are as follows:

- many erroneous repetition structures may be detected.
- the amount of computation increases because the number of candidate repetition structures increases.

DP (Dynamic Programming) Matching [5] is used for the similarity calculation, and when the score for matching parts exceeds a threshold, our system judges that they are similar.

Integration into the Group

Here, we explain the process for cases where the elements that should be contained in a repetition structure are not contained therein due to lack or addition of elements. For example, consider the following fragment.

Book – Movie – Music – NEW!TV

"Book", "Movies", "Music" and "TV" should be in a repetition structure. However, since a image "NEW!" is attached to "TV", only the first three can be part of a repetition structure.

Next, consider the example shown in Fig. 6. "C" is an element which should be contained in a repetition structure. The position where "C" should be connected is "AB+" or "B+" in the figure; therefore, if "C" is similar to "AB+", it is connected with "AB+", and if "C" is similar to "B+", it is connected with "B+".

Detecting Repetition Structures using Similarity

In this process, we introduce similarity into the definition of repetition structures and loosen the strict definition of the previous step in order to deal with cases that could not be dealt with in the previous step. The basic idea is the same as in the previous step: if the tags which are contained in a span and its successive span are well-balanced and they are similar to each other, our system regards them as a repetition structure and structures them.

This step is repeated until new repetition structures are no longer detected.

5. Evaluation

5.1. Testing Robustness

For the robustness test, we collected 80 Web pages at random. There were ten pages for each of the following eight categories: portal sites, top pages of companies, site maps, dynamic pages generated by cgi etc, pages using CSS (Cascading Style Sheets), pages in languages except Japanese, pages in our lab., and pages fetched by using yahoo's random link.

Results of the Robustness Test

Our system could not process 15 out of 80 pages. The causes were based on the following two errors due to the tools our system used.

- Tidy's errors
- errors of the character set conversion utility

Our system could not finish processing 9 pages in 24 hours. This problem comes from the fact that our system needs much computation to find the best combination of the detected repetition structures. As for the processing time, however, 45% of the pages can be processed within 10 seconds, and 71% of the pages can be processed within 1 minute (using Athlon XP1800+, 1.5GB Memory, Linux 2.4.18, Perl 5.6.1).

5.2. Accuracy of Segmentation and Structuring

We collected another test set of 70 pages that our system could process at random for all categories except "languages other than Japanese" and conducted the subjective evaluation. The experiment focused on whether the outputted structure would be accepted by human subjects or not. Three subjects carefully examined the outputted structure of each page, looking at the rendered page as well, and were asked to rate it on a scale of one to five for segmentation and structuring.

They were also asked to point out errors in it.

Results of the Accuracy Test

Table 1 shows the results, which are the averages of the three subjects' ratings. "Seg." and "Str." mean the scores for segmentation and structuring, respectively. The scores are close to 5; thus, the structures made by our system were considered to be close to the structure understood by the subjects.

Category	Seg.	Str.
portal	4.43	3.40
top	4.63	4.33
sitemap	4.87	3.97
dynamic	4.30	3.53
CSS	4.33	3.83
lab.	4.40	3.13
random	4.20	3.50
All	4.45	3.67

Table 1. Result (average for each category)

5.3. Discussion

In this section, we briefly discuss one of the problems that were pointed out by the subjects.

The scores for "portal", "top" and "sitemap" are better than the others. This is considered to be related to the following question: "Which tags should we use for detecting repetition structures?" Our current system uses `<a>` tags. However, there are cases where `<a>` tags should not be used. Consider the following examples (Assume that underlined words have a link.):

(A) Yesterday, I went here and here.
(B) Movie - Music - TV

If our system uses `<a>` tags, in case (A), the sentence is divided into three segments. However, in this case, we feel that there is no segment boundary, and the sentence should belong to a segment. On the other hand, if our system does not use `<a>` tags, case (B) cannot be structured at all.

Therefore, the tags used for detecting repetition structures should be determined dynamically depending on the Web pages or the parts of the Web pages to be structured.

6. Conclusion & Future Work

In this paper, we proposed a technique to segment and structure Web pages. The results show that most of the structures that our system outputs correspond well to the structures that human subjects understand, even though our current system still has some problems.

As future work, we plan to refine our system so that it will be able to structure the Web pages by combining the current bottom-up strategy with a top-down strategy using the information of the DOM (Document Object Model) structure [3]. We also plan to automatically annotate a segment with type by using the structural feature.

References

[1] Y. Yang, and H.-J. Zhang, "HTML page analysis based on visual cues", Proc. 6th International Conference on Document Analysis and Recognition, pp.859–864, Sept., 2001.

[2] Y. Chen, W.-Y. Ma, and H.-J. Zhang, "Detecting web page structure for adaptive viewing on small form factor devices", Proc. Twelfth International World Wide Web Conference, pp.225–233, May, 2003.

[3] S. Yu, D. Cai, J.-R. Wen, and W-Y. Ma, "Improving pseudo-relevance feedback in web information retrieval using web page segmentation", Proc. Twelfth International World Wide Web Conference, pp.11–18, May, 2003.

[4] Dave Raggett, "Clean up your Web pages with HTML TIDY", URL: http://www.w3.org/People/Raggett/tidy/.

[5] Gonzalo Navarro, Mathieu Raffinot, "Flexible Pattern Matching in Strings", Cambridge University Press, 2002.

[6] Jay Yellen, Jonathan L. Gross, "Graph Theory and Its Applications", CRC Press, 1998.