We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

21

Automated Methods for Webpage Usability & Accessibility Evaluations

Hidehiko Okada[1] and Ryosuke Fujioka[2]
[1] *Kyoto Sangyo University,* [2] *Kobe Sogo Sokki Co. Ltd.*
Japan

1. Introduction

Because of the rapid growth of the web and its users, web usability and accessibility become more and more important. This chapter describes two automated methods for evaluating usability/accessibility of webpages. The first method, for usability evaluation, is based on interaction logging and analyses, and the second method, for accessibility evaluation, is based on machine learning. In the following Sections 2 and 3, the two methods are described respectively.

Several methods have been proposed and developed for usability evaluation based on user interaction logs. Interaction logs can be recorded by computer logging programs (automated logging) or by human evaluators (observational logging). Analysis methods and tools for the former type of logs are well summarized by Ivory (Ivory & Hearst, 2001; Ivory, 2003), and those for the latter type of logs have also been developed (for example, by Qiang (Qiang et al., 2005)). Our method is for analyzing former type of logs. In the survey by Ivory, analysis methods for automatically captured log files for WIMP (Window, Icon, Menu and Pointing device) applications and web applications are categorized in terms of their approaches including metric-based, pattern-based, task-based and inferential ones. Some of the methods with the task-based approach compare user interaction logs for a test task with desired (expected) interaction sequences for the task and detect inconsistencies between the user logs and the desired sequences (Kishi, 1995; Uehling & Wolf, 1995; Okada & Asahi, 1996; Al-Qaimari & Mcrostie, 1999; Helfrich & Landay, 1999; Zettlemoyer et al., 1999). The inconsistencies are useful cues for finding usability problems: for example, an evaluator can find that users selected some unexpected link on a webpage when another link on the page was expected for the test task and that the link selected by the users may have some usability problem in its design (labeling, layout, etc.).

The existing methods require widget-level logs for the comparisons. For example, the method proposed by Okada (Okada & Asahi, 1996) requires interaction logs to include data of widget properties such as widget label, widget type, title of parent window, etc. This requirement degrades independency and completeness of the methods in logging user interactions with systems under evaluation. Section 2 in this chapter describes our method that detects inconsistencies between user logs and desired sequences based on logs of *clicked points ((x, y) coordinate values)*. Coordinate values of clicked points can be easily and fully logged independently of what widgets are clicked on. Several existing methods have also

utilized the logs of mouse clicked points (for example, "Mousemap" that visualizes mouse moves (Gellner & Forbrig, 2003)), but the methods do not achieve the detection of inconsistencies. The authors have developed a computer tool for logging and analyzing user interactions and desired sequences by our method. The tool is applied to experimental usability evaluations of websites. Effectiveness of the method in usability testing of webpages is evaluated based on the application result.

Besides, to make webpages more accessible to people with disabilities, <table> tags should not be used as a means to visually layout document content: layouting contents by <table> tags may present problems when rendering to non-visual media (W3C, 1999ab). It is reported that the number of tables on pages doubled from 7 in 2000 to 14 in 2003, with most tables being used to control page layouts (Ivory & Megraw, 2005). Therefore, to evaluate the accessibility of webpages, it should be checked whether the pages include layout-purpose <table> tags. Several methods and tools have been proposed and developed for web accessibility (Cooper, 1999; Scapin et al., 2000; Ivory, 2003; Ivory et al., 2003; Abascal et al., 2004; Brajnik, 2004; Beirekdar et al., 2005; Vanderdonckt & Beirekdar, 2005). Accessibility tools are listed in (W3C, 2006; Perlman, 2008) and compared in (Brinck et al., 2002). Some tools detect deeply nested <table> tags that are likely to be layout-purpose ones. Still, a method for automated detection of layout-purpose <table> tags in HTML sources is a challenge: it requires further than simply checking whether specific tags and/or attributes of the tags are included in the sources. Section 3 describes our method for the detection that is based on machine learning. The proposed method derives a <table> tag classifier that classifies the purpose of the tag: the classifier deduces whether a <table> tag is a layout-purpose one or a table-purpose one. The section describes our system that implements the proposed method and report a result of experiment for evaluating classification accuracy.

2. Automated Method for Webpage Usability Evaluation

2.1 Method for analyzing mouse click logs

2.1.1 User logs and desired logs

A user log can be collected by logging mouse clicks while a user (who does not know the desired sequence of a test task) performs the test task in user testing. In our research, a log file is collected for a test user and a test task: if the number of users is N and the number of tasks is M then the number of user log files is N*M (where all the N users completes all the M tasks). A "desired" log is collected by logging mouse clicks while a user (who knows well the desired sequence of a test task) performs the test task. For a test task, one desired log file is usually collected. If two or more different interaction sequences are acceptable as desired ones for a test task, two or more desired log files can be collected (and used in the comparison described later).

2.1.2 Method for detecting inconsistencies in user/desired logs

Our method models two successive clicks as a vector and thus a sequence of operation in a user/desired log as a sequence of vectors. A vector is from the ith clicked point to the (i+1)th clicked point in the screen. To detect inconsistencies in a user log and a desired log, each vector from the user log is compared with each vector from the desired log. If the distance of the two vectors ($\mathbf{v_u}$ from the user log and $\mathbf{v_d}$ from the desired log) is smaller than a threshold, $\mathbf{v_u}$ and $\mathbf{v_d}$ are judged as being matched: the user operation modeled by $\mathbf{v_u}$ is

supposed to be the same operation modeled by $\mathbf{v_d}$. The method defines the distance $D(\mathbf{v_u}, \mathbf{v_d})$ as a weighted sum of D_p and D_v (Fig. 1).

$$D_p = (w_x(x_{u1}-x_{d1})^2 + w_y(y_{u1}-y_{d1})^2)^{0.5} \quad (1)$$

$$D_v = (w_x(x_{u2}-(x_{u1}-x_{d1})-x_{d2})^2 + w_y(y_{u2}-(y_{u1}-y_{d1})-y_{d2})^2)^{0.5} \quad (2)$$

$$D(\mathbf{v_u}, \mathbf{v_d}) = w_pD_p + w_vD_v \quad (3)$$

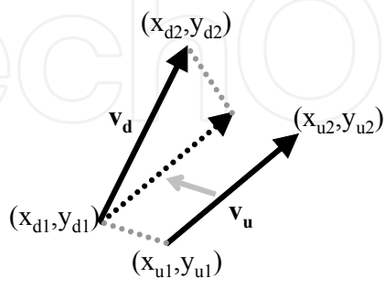

Figure 1. Two vectors $\mathbf{v_u}$, $\mathbf{v_d}$ and their distance

The role of weight factors w_x and w_y used in the calculations of D_p and D_v is as follows. Users click on links in webpages under evaluation to perform their tasks. The width of a link is usually larger than the height of the link, especially of a text link. Therefore, the differences of clicked points for clicking on the same link are likely to become larger for the horizontal axis (the x coordinate values) than for the vertical axis (the y coordinate values). To deal with this, weights w_x and w_y are used so that the horizontal differences can be counted smaller than the vertical differences.

User operations to scroll webpages by using mouse wheels should also be taken into account: scrolls by mouse wheels changes widget (e.g., link) positions in the screen so that the clicked positions may not be the same even for the same widget. Our method records the amount of wheel scrolls while logging interactions. By using the logs of wheel scrolls, coordinate values of clicked points are adjusted. Fig. 2 shows this adjustment. Suppose a user clicked on the point (x_i, y_i) in the screen (Fig. 2(a)) and then clicked on the point (x_{i+1}, y_{i+1}) (Fig. 2(b)). In this case, the vector derived from the two clicks is the one shown in Fig. 2(c). As another case, suppose a user scrolled down a webpage along the y axis by using the mouse wheel between the two clicks and the amount of the scroll was S pixels. In this case, the vector derived from the two clicks is the one in Fig. 2(d).

2.1.3 Two types of inconsistencies as cues of usability problems

As cues of usability problems, the proposed method detects two types of inconsistencies between user interactions and desired sequences. The authors refer to them as "unnecessary" operations and "missed" operations. Fig. 3 illustrates these kinds of operations.

Unnecessary operations are user operations judged as not included in the desired sequences, i.e., unnecessary operations are operations in a user log for which any operation in desired logs is not judged as the same one in the comparison of the user/desired logs. Our method supposes such user operations as unnecessary because the operations may not be necessary for completing the test task. Unnecessary operations can be cues for human evaluators to find usability problems that users clicked on a confusing link when another link is desired (expected) to be clicked on for the task.

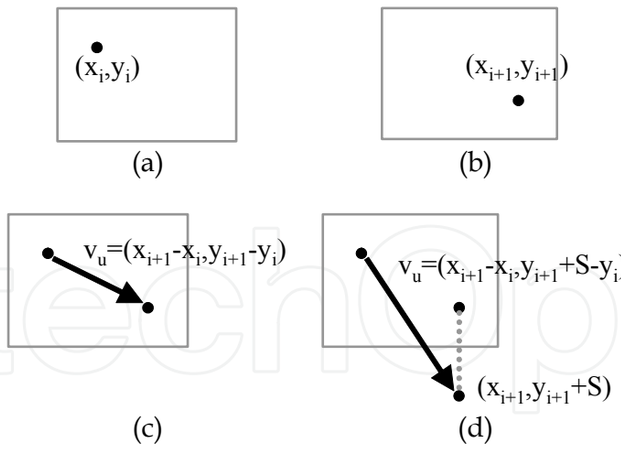

Figure 2. Adjustment of clicked point for mouse wheel scroll

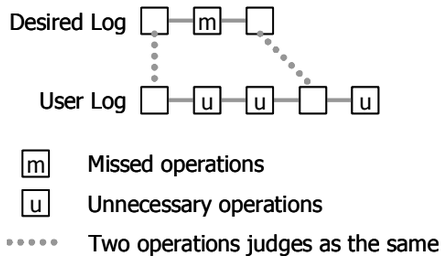

Figure 3. Unnecessary operations and missed operations

Missed operations are desired operations judged as not included in the user interaction sequences, i.e., missed operations are operations in desired logs for which any operation in a user log is not judged as the same one. Our method supposes such user operations as missed because the operations may be necessary for completing the test task but the user finished the task without performing the operations. Missed operations can be cues for human evaluators to find usability problems that a link is not clear enough or not easy to find for users. Our method models an operation in a user/desired log by a vector derived from clicked point logs, so the method detects unnecessary/missed operations as unnecessary/missed vectors.

Suppose two or more successive operations are unnecessary ones in a user log. In this case, the first operation is likely to be the best cue in the successive unnecessary operations. This is because the user might deviate from the desired sequence by the first operation (i.e., the expected operation is not clear enough for the user) and had performed additional operations irrelevant to the test task until the user returned to the desired sequence. Our method can extract the first operations in the successive unnecessary operations and show them to human evaluators so that the evaluators can efficiently analyze usability problem cues (unnecessary operations in this case).

The idea of analyzing unnecessary/missed operations in user interaction logs (inconsistencies in user/desired logs) is not novel (e.g., Okada & Asahi, 1996), but our method described in this section is unique in that it extracts those inconsistent operations from logs of clicked points (logs of (x, y) coordinate values), not widget-level logs.

2.1.4 Unnecessary/missed operations common to users

Unnecessary/missed operations common in many of test users are useful cues for finding problems less independently of individual differences among the users. Our method analyzes how many users performed the same unnecessary/missed operation.

The analysis of the user ratio for the same missed operation is simple: for each missed operation, the number of user logs that do not include the desired operation is counted. To analyze the user ratio for the same unnecessary operation, the method compares unnecessary operations extracted from all user logs of the test task. This comparison is achieved by the same way as operations (vectors) in user/desired logs are compared. By this comparison, unnecessary operations common among multiple users can be extracted (Fig. 4).

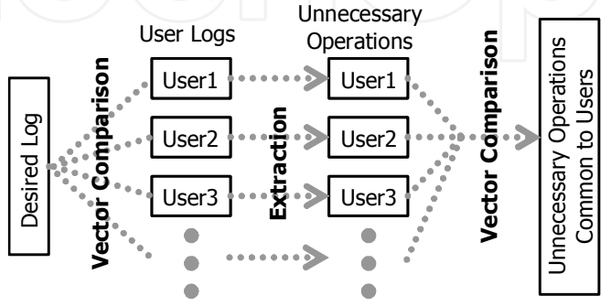

Figure 4. Extraction of unnecessary operations common to users

2.2 Evaluating effectiveness based on case study

2.2.1 Design of experiment

Ten websites of business/public organizations were selected. For each site, a test task was designed. The average number of clicks in the designed sequences for the ten test tasks was 3.9. Five university students participated in this experiment as test users. Each test user was asked to perform the task on each site. They had enough knowledge and experience in using webpages with a PC web browser but they used the websites for the first time. The desired sequences of the test tasks were not told to the test users. Thus, if the desired sequences are not clear enough for the test users, the users are likely to deviate from the desired sequences and unnecessary and/or missed operations are observed. The interaction of each user for each test task was logged into a user log file. To avoid fatigue affecting the results, the time of experiment for each user was limited to 60+ minutes: each test user was asked to perform a test task within five or ten minutes depending on the task size.

Fifty user logs (five users * ten tasks) and ten desired logs (a log per test task) were collected. For each task, our tool implementing the proposed method analyzed the logs and extracted possible cues of usability problems (i.e., unnecessary/missed operations). An evaluator tried to find usability problems based on the extracted cues.

2.2.2 Weight factors and thresholds for vector distance

Our method requires us to determine the values of weight factors w_x, w_y, w_p and w_v and the threshold value of vector distance (see 2.1.2). To determine these values, a pre-experiment was conducted with another test user. Based on the analysis of the log files collected by the pre-experiment, appropriate values were determined that led an accurate result in detecting unnecessary/missed operations. Values in the row labeled as “Original” in Table 1 show the obtained values.

	w_x	w_y	w_p	w_v	Threshold (pixel)
Original	0.4	1.0	0.5	1.0	100
Variation A	0.4	1.0	0.0	1.0	67
Variation B	0.4	1.0	1.0	0.0	34

Table 1. Values of weight factors and threshold for vector distance

In our method, distance of two operations (vectors) is defined by Eqs. (1)-(3). In the case where $w_v = 0$ and $w_p = 1$, $D(\mathbf{v_u}, \mathbf{v_d}) = D_p$ so that two operations are not compared on the basis of two successive clicks (i.e., on the basis of vectors) but compared on the basis of a single click. In this case, a click in a user log and a click in a desired log are judged as the same operation if the clicked points are near. Similarly, in the case where $w_p = 0$ and $w_v = 1$, $D(\mathbf{v_u}, \mathbf{v_d}) = D_v$ so that two operations are compared by the size of vector difference only (i.e., the absolute position on which the click is performed in the screen is not considered). These two variations of the method were also evaluated in addition to the original one. Variations A/B in Table 1 denote them in which $(w_p, w_v) = (0.0, 1.0)$ and $(1.0, 0.0)$, respectively.

2.2.3 Number of problems found

To evaluate the effectiveness of our method in finding usability problems, the number of problems found by the method were compared with the number by a method based on manual observation of user interactions. In addition to record click logs in user interaction sessions, PC screen image had been captured to movie files (a screen recorder program was used in the PC). A human evaluator observed user interactions with the replay of the captured screen movies and tried to find usability problems. This manual method is expected to require much time for the interaction observation but contribute to find problems thoroughly. In this experiment, the evaluator who tried to find problems by the proposed method and the evaluator who tried to find problems by the manual method were different so that the result with a method did not bias the result with another method.

Table 2 shows the number of problems found by each of the methods. The values in the table are the sum for the ten test tasks (sites). Eleven problems were shared in the four sets of the problems, i.e., the proposed (original) method contributed to find 61% (=11/18) of the problems found by the manual method. Although the number of problems found by the proposed method was smaller than the manual method, the time for a human evaluator to find the problems by the proposed method was much less than the time by the manual method. In the case of the manual method, an evaluator had to investigate all clicks by the users because user clicks that would be possible problem cues were not automatically extracted. In the case of the proposed method, an evaluator required to investigate smaller number of clicks extracted as possible problem cues by the method. In this experiment, the number of clicks to be investigated in the case of the proposed method was 10-20% of the number in the case of the manual method.

This result of case study indicates that the proposed method will

- contribute to find usability problems to a certain extent in terms of the number of problems, and
- be much efficient in terms of the time required.

Method	#Problems
Manual	18
Original	15
Variation A	14
Variation B	13

Table 2. Number of problems found by each method

2.2.4 Unnecessary/missed operations contributing to finding problems

Not all unnecessary/missed operations extracted by the proposed method may contribute to finding usability problems. As the number of unnecessary/missed operations that contribute to finding problems is larger, the problems can be found more efficiently. The contribution ratio was investigated for the proposed method and its variations (Table 3). Values in the "Counts (first)" column are the counts of unnecessary operations that were the first in two or more successive unnecessary operations (see 2.1.3). For example, the original method extracted four missed operations in total from log files of the ten test tasks, and 25.0% (one) of the four operations contributed to finding a problem. Similarly, the original method extracted 375 unnecessary operations in total, and 7.5% (28) of the 375 operations contributed to finding problems.

	Missed Operations		Unnecessary Operations			
Method	Counts	Ratio	Counts (all)	Ratio	Counts (first)	Ratio
Original	4	25.0%	375	7.5%	51	49.0%
Variation A	8	37.5%	422	5.7%	58	39.7%
Variation B	1	0.0%	299	9.4%	72	37.5%

Table 3. Number of unnecessary and missed operations found by each method and the ratio of contribution in finding problems

Findings from the result in Table 3 are as follows.

- In the three methods, the ratios were larger for unnecessary operations (first) than those for unnecessary operations (all). This result supports our idea that an evaluator can find usability problems more efficiently by analyzing the first operations only in successive unnecessary operations.
- In the result of unnecessary operations (first), the ratio for the original method was larger than either of the two variations. In the result of missed operations, the ratio for the original method was larger than that for the variation B but smaller than that for the variation A. This indicates that both the original method and its variation A are promising ones.

See Section 4 for the conclusion of this research on the log-based usability evaluation method.

3. Automated Method for Webpage Accessibility Evaluation

3.1 Basic idea

<Table> tags are used for (a) expressing data tables as the tags are originally designed for or (b) adjusting the layout of document contents. In the case where a <table> tag is used for the

table-purpose, the data in the same row or column are semantically related with each other (e.g., values of the same property for several data items, or values of several properties for the same data item) and the relation can be expressed by row/column headers. On the contrary, in the case where a <table> tag is used for the layout-purpose, the data in the same row or column may not be semantically related. Thus, to deduce the purpose of a <table> tag in a fundamental approach, it should be analyzed whether or not the data in the same row/column of the table are the semantically related ones. To make the analyses automated by a computer program requires a method for semantic analyses of table contents, but the automated semantic analyses with enough precision and independency for any kinds of webpages is hard to achieve.

Our basic idea focuses on machine-readable design attributes of a table instead of analyzing semantics of data in a table. If some common design pattern is found in some attribute values of layout-purpose <table> tags and another common design pattern is found in the attribute values of table-purpose <table> tags, the two purposes can be discriminated by denoting classification rules based on the design patterns. However, it is unknown whether we can find such design patterns, and even if we can, it will be hard for us to manually investigate common patterns by analyzing design attribute values in a large number of <table> tag instances and denote sufficient rules to precisely classify the tags. In our research, the authors manually investigate design attributes only: to automatically derive classification rules, a machine learning method is utilized.

Among several kinds of machine learning methods available, ID3 (Quinlan, 1986), a method for deriving a decision tree from a set of data instances, is utilized. An advantage of a decision tree as a classifier over other forms of classifiers (e.g., a multi-layered neural network) is that classification rules are obtained as tree-formed explicit if-then rules and thus easy to read for human users of the method.

3.2 <Table> tag design attributes for classification rules

The authors first collected and investigated <table> tag instances (webpage HTML sources in which <table> tags were included) and extracted design attributes of <table> tags that were likely to contribute to the classification. Webpages were collected from various categories in Yahoo webpage directory so that the pages were not biased from the viewpoint of page categories. The authors manually judged purposes of the <table> tag instances in the collected pages. By this way, 200 <table> tags were collected, a half of which were layout-purpose ones and the others table-purpose ones. The authors then extracted common design patterns for each set of <table> tags. Findings were as follows.

Common design patterns in layout-purpose <table> tags

- <Table> tags are nested.
- Some cell(s) in the table include(s) image(s).
- The number of HTML tags that appear before the <table> tag in the source is small.
- Some cells in the table are spanned.
- The width and/or height are/is specified.

Common design patterns in table-purpose <table> tags

- The table has visible border lines.
- The table includes many rows.
- <Th> tags for row/column headers are included.
- Table titles are specified.

By denoting classification rules based on these common design patterns, it will be possible to deduce the purposes of <table> tags to a certain extent. To derive the rules in a form of decision tree by ID3, 10 attributes in Table 4 were determined.

- Binary values for the eight attributes of "border"-"width" mean whether or not the table has visible borders, captions, etc. For example, if the border attribute is specified for the <table> tag and the attribute value is 1 or more then border = Y (Yes), else border = N (No).
- A value of the attribute "num_tag" is the number of HTML tags that appear before the <table> tags in the source. In counting the tags, those in the header part are not counted.
- A value of the attribute "num_tr" is the number of rows (<tr> tags) in the table.

Name	Meaning	Values
border	Whether the table has visible border lines.	Binary
caption	Whether the table has a caption.	
height	Whether the table height is specified.	
img	Whether a cell in the table includes an item of image data.	
nest	Whether the table includes a nested table in itself.	
span	Whether some cells are spanned.	
th	Whether a row/column has a title header for the data in the row/column.	
width	Whether the table width is specified.	
num_tag	The number of HTML tags that appear before the <table> tags in the source.	Positive integer
num_tr	The number of rows in the table.	

Table 4. Attributes for decision tree

3.3 System configuration for the proposed method

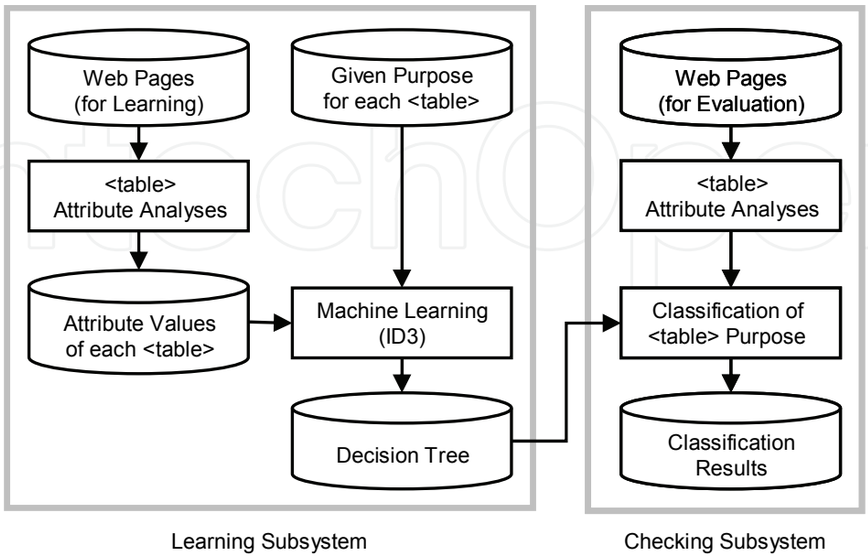

Figure 5. System configuration for the proposed method

Fig. 5 shows the system configuration for our method. In the learning phase, the method derives rules for classifying <table> tags from a set of webpages in which layout/table-purpose <table> tags are included. First, for each <table> tag in the learning data, attribute values are obtained by analyzing the webpage HTML source in which the tag is included. The purpose of each tag in the learning data is given by manual judgments in collecting the learning data. From these data of attribute values and given purposes, classification rules are derived by a machine learning method. In the case where ID3 is applied as the learning method, the rules are obtained as a decision tree (i.e., tree-formed if-then rules). In the checking phase, <table> tags (of which the purpose is unknown) are classified by the rules obtained in the learning phase. Attribute values of each <table> tag for the classification are obtained by the same way as in the learning phase (the <table> attribute analyses module in Fig. 5 is shared by the learning and checking subsystems). The obtained attribute values are applied to if parts of the rules, and the purpose of each <table> is deduced by the rule of which the if part is satisfied.

3.4 Evaluation of the proposed method

3.4.1 Evaluation method

To evaluate the effectiveness of our method, the authors investigated the classification accuracy by 10-fold cross validation (CV) with the 200 <table> tags collected (see 3.2). The 200 tags were randomly divided into 10 groups (G1, G2, …, G10). A decision tree was derived by ID3 applying to 180 <table> tags in 9 groups excluding Gi, and the classification accuracy was checked with 20 <table> tags in Gi (i=1,2,…,10). The accuracy rate was calculated as follows.

$$\text{Accuracy Rate} = (\text{Number of <table> tags correctly classified})/20 \quad (4)$$

Ten values of the accuracy rate were obtained by a trial of 10-fold CV. The authors tested the 10-fold CV three times and statistically investigated the accuracy rates.

3.4.2 Decision trees obtained by ID3

By the three trials of 10-fold CVs, 30 decision trees were obtained in total. Examples of the decision trees are shown in Tables 5-7 (only the nodes in the depth 3 are shown). Values in the "No." column are the serial numbers of the nodes where the #0 node is the root node. Tables 5-7 denote parent-child node relationships by indents in the "Rule Element" column. For example, in Table 5, the #1 and #8 nodes are child nodes of the #0 node and the #2 and #5 nodes are child nodes of the #1 node. Values in the "Rule Element" column are the conditions in if-parts of rules. Conditions that appear in a path from the root node to a leaf node are connected with AND. Values in the "Table" and "Layout" columns are the numbers of table/layout-purpose <table> tags in the learning data included in the node. For example, Table 5 shows the followings.

- The root node includes 89 table-purpose tags and 91 layout-purpose tags (see #0 node).
- Of the 180 <table> tags in the root node,
  - those with border=N are 86. Nine tags are table-purpose ones and the other 77 tags are layout-purpose ones (see #1 node), and
  - those with border=Y are 94. Eighty tags are table-purpose ones and the other 14 tags are layout-purpose ones (see #8 node).
- Leaf nodes are those of which the value in the "Table" column or the "Layout" column

is 0. A leaf node corresponds to an if-then rule. For example, the #3 node denotes that all tags that meet the condition:
(border = N) and (num_tr <= 7) and (num_tag <= 12)
are layout-purpose ones. Thus, the following rule is obtained from the #3 node.
"If (border = N) and (num_tr <= 7) and (num_tag <= 12) then the tag is a layout-purpose one."

The examples of decision trees shown in Tables 5-7 are typical ones in the 30 trees obtained. The other trees had similar structures from the root node to nodes in depth 3 as either of the three examples. The 30 trees reveal that the attributes border, num_tr, num_tag and nest is likely to appear in the upper layers of the tree, i.e., these attributes well contribute to the classification.

No.	Rule Element	Table	Layout	Total
0	(root)	89	91	180
1	border = N	9	77	86
2	num_tr <= 7	1	71	72
3	num_tag <= 12	0	66	66
4	num_tag > 12	1	5	6
5	num_tr > 7	8	6	14
6	nest = N	8	1	9
7	nest = Y	0	5	5
8	border = Y	80	14	94
9	nest = N	78	5	83
10	img = N	59	2	61
11	img = Y	19	3	22
12	nest = Y	2	9	11
13	height = N	0	5	5
14	height = Y	2	4	6

Table 5. Example (1) of decision trees obtained in the experiment

No.	Rule Element	Table	Layout	Total
0	(root)	89	91	180
1	num_tr <= 6	15	82	97
2	border = N	3	69	72
3	img = N	3	18	21
4	img = Y	0	51	51
5	border = Y	12	13	25
6	nest = N	12	5	17
7	nest = Y	0	8	8
8	num_tr > 6	74	9	83
9	nest = N	72	1	73
10	border = N	9	1	10
11	border = Y	63	0	63
12	nest = Y	2	8	10
13	num_tag <= 7	0	8	8
14	num_tag > 7	2	0	2

Table 6. Example (2) of decision trees obtained in the experiment

No.	Rule Element	Table	Layout	Total
0	(root)	86	94	180
1	num_tr <= 6	15	84	99
2	num_tag <= 12	3	76	79
3	img = N	3	20	23
4	img = Y	0	56	56
5	num_tag > 12	12	8	20
6	nest = N	12	4	16
7	nest = Y	0	4	4
8	num_tr > 6	71	10	81
9	nest = N	69	1	70
10	border = N	10	1	11
11	border = Y	59	0	59
12	nest = Y	2	9	11
13	border = N	0	8	8
14	border = Y	2	1	3

Table 7. Example (3) of decision trees obtained in the experiment

3.4.3 Evaluation of classification accuracy

Accuracy rates obtained by the three trials of the 10-fold CVs are shown in Table 8. Ten values of the accuracy rates are obtained by a single trial of CV, and the values in Table 8 are the minimum, maximum, mean and SD values of the ten accuracy rates for each trial. In all of the three trials, the maximum rate was 100% so that all checking <table> tags were correctly classified. The mean values were around 90% and the SD values were small, which supports the effectiveness of our method. Improvements for better values of the minimum accuracy rates are the further research challenges.

4. Conclusions

In this chapter, Section2 described our method that extracts cues for finding usability problems from user/desired logs of clicked points. To detect inconsistencies between user and desired logs, the method compares operations in the logs. The method compares user/desired operations by modeling each operation as a vector derived from coordinate values of the clicked points and checking the distance between two vectors. The distance was defined as a weighted sum of distance between start points and size of difference for the two vectors. The method extracts two types of inconsistencies: unnecessary and missed operations. Effectiveness of the proposed method was evaluated based on a case study. Each of the two human evaluators tried to find usability problems for ten websites by the proposed method and the manual method respectively. The proposed method contributed to find 61% of the usability problems found by the manual method in much smaller amount of time: the number of clicks analyzed by an evaluator with the proposed method was only 10-20% of that with the manual method. This result indicates that the method will help evaluators to quickly and roughly focus their attentions to problems cues in user interactions. In our future work, additional case studies are necessary for further evaluations and improvements of the method.

	1st CV	2nd CV	3rd CV
Min.	85	80	80
Max.	100	100	100
Mean	92	89	94
SD	6.0	7.5	6.7

Table 8. Classification accuracy rates (%)

Section 3 described our method for detecting layout-purpose <table> tags in webpage HTML sources. A machine learning method was utilized for deriving <table> tag classifiers. A system was developed that utilized ID3 as the machine learning method. The system derives a decision tree as the classifier from a set of <table> tag data for learning. Classification accuracy was evaluated by 10-fold CVs with 200 webpages collected from the web. It was found that the purposes could roughly be discriminated with attributes of border, num_tr, num_tag and nest shown in Table 4: these attributes were likely to appear in upper layers in decision trees. In the experiment with the 200 <table> tags collected, mean accuracy rates were around 90%. In our future work, it will be investigated whether machine learning methods other than ID3 (e.g., C4.5, multi-layered neural network, support vector machine) can improve the accuracy. These methods will be utilized in our system and classification accuracy rates will be compared within the methods.

5. References

Abascal, J.; Arrue, M.; Fajardo I.; Garay, N. & Tomas, J. (2004). Use of guidelines to automatically verify web accessibility, *Universal Access in the Information Society*, Vol.3, No.1, pp.71-79.

Al-Qaimari, G. & Mcrostie, D. (1999). KALDI: a computer-aided usability engineering tool for supporting testing and analysis of human computer interaction, *Proc. of the Third Int. Conf. on Computer-Aided Design of User Interfaces*, pp.337-355.

Beirekdar, A.; Keita, M.; Noirhomme, M.; Randolet, F.; Vanderdonckt, J. & Mariage, C. (2005). Flexible reporting for automated usability and accessibility evaluation of web sites, *Lecture Notes in Computer Science, Vol.3585 (Proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction (INTERACT2005))*, pp.281-294.

Brajnik, G. (2004). Comparing accessibility evaluation tools: a method for tool effectiveness, *Universal Access in the Information Society*, Vol.3, Nos.3-4, pp.252-263.

Brinck, T.; Hermann, D.; Minnebo, B. & Hakim, A. (2002). AccessEnable: a tool for evaluating compliance with accessibility standards, *CHI'2002 Workshop on Automatically Evaluating the Usability of Web Sites*, available at http://simplytom.com/research/AccessEnable_workshop_paper.pdf.

Cooper, M. (1999). Evaluating accessibility and usability of web sites, *Proc. of 3rd Int. Conf. on Computer-Aided Design of User Interfaces (CADUI'99)*, pp.33-42.

Gellner, M. & Forbrig, P. (2003). ObSys - a tool for visualizing usability evaluation patterns with Mousemaps, *Proc. of the Tenth Int. Conf. on Human-Computer Interaction*, pp.469-473.

Helfrich B. & Landay, J. A. (1999). QUIP: quantitative user interface profiling, available at http://www.helcorp.com/bhelfrich/helfrich99quip.pdf.

Ivory, M. Y. (2003). *Automated web site evaluation: researchers' and practitioners' perspectives*, Kluwer Academic Publishers.

Ivory, M. Y. & Hearst, M. A. (2001). The state of the art in automated usability evaluation of user interfaces, *ACM Computing Surveys*, Vol.33, No.4, pp.1-47.

Ivory, M. Y.; Mankoff, J. & Le, A. (2003). Using automated tools to improve web site usage by users with diverse abilities, *IT&Society*, Vol.1, No.3, pp.195-236. available at http://www.stanford.edu/group/siqss/itandsociety/v01i03/v01i03a11.pdf.

Ivory, M. Y. & Megraw, R. (2005). Evolution of web site design patterns, *ACM Trans. on Information Systems*, Vol.23, No. 4, pp.463-497.

Kishi, N. (1995). Analysis tool for skill acquisition with graphical user interfaces based on operation logging, *Proc. of the 6th Int. Conf. on Human-Computer Interaction*, pp.161-166.

Okada, H. & Asahi, T. (1996). GUITESTER: a log-based usability testing tool for graphical user interfaces, *IEICE Transaction on Information and Systems*, Vol.E82-D, No.6, pp.1030-1041.

Perlman, G. (2008). Accessibility links – tools, *HCI Bibliography*, available at http://www.hcibib.org/accessibility/#TOOLS.

Qiang, Y.; Suzuki, T.; Sakuragawa, S.; Tamura, H. & Kurosu, M. (2005). The development of OBSERVANT EYE to effective implement observation records for usability testing, *Proc. of the 11th Int. Conf. on Human-Computer Interaction*, CD-ROM.

Quinlan, J. R. (1986). Induction of decision trees, *Machine Learning*, Vol.1, pp.81-106.

Scapin, D.; Leulier, C.; Vanderdonckt, J.; Mariage, C.; Bastien, C.; Farenc, C.; Palanque, P. & Bastide, R. (2000). A Framework for organizing web usability guidelines, *Proc. of the 6th Conf. on Human Factors & the Web*, available at http://www.isys.ucl.ac.be/bchi/publications/2000/Scapin-HFWeb2000.htm.

Uehling, D. L. & Wolf, K. (1995). User action graphing effort (UsAGE), *Proc. of the Conf. on Human Factors in Computing Systems*, pp.290-291.

Vanderdonckt, J. & Beirekdar, A. (2005). Automated web evaluation by guideline review, *Journal of Web Engineering*, Vol.4, No.2, pp.102-117.

W3C. (1999a). HTML 4.01 specification, available at http://www.w3.org/TR/html4/.

W3C. (1999b). Web content accessibility guidelines 1.0, available at http://www.w3.org/TR/WCAG10/.

W3C. (2006). Complete list of web accessibility evaluation tools, available at http://www.w3.org/WAI/ER/tools/complete.

Zettlemoyer, L. S.; St.Amant, R. S. & Dulberg, M. S. (1999). IBOTS: Agent control through the user interface, *Proc. of the Int. Conf. on Intelligent User Interfaces*, pp.31-37.

Advances in Human Computer Interaction

Edited by Shane Pinder

ISBN 978-953-7619-15-2
Hard cover, 600 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

In these 34 chapters, we survey the broad disciplines that loosely inhabit the study and practice of human-computer interaction. Our authors are passionate advocates of innovative applications, novel approaches, and modern advances in this exciting and developing field. It is our wish that the reader consider not only what our authors have written and the experimentation they have described, but also the examples they have set.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hidehiko Okada and Ryosuke Fujioka (2008). Automated Methods for Webpage Usability & Accessibility Evaluations, Advances in Human Computer Interaction, Shane Pinder (Ed.), ISBN: 978-953-7619-15-2, InTech, Available from: http://www.intechopen.com/books/advances_in_human_computer_interaction/automated_methods_for_webpage_usability__amp__accessibility_evaluations

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen