September 6, 2000
RT0378
Human-Computer Interaction   7 pages

Research Report

TextExtractor: Text Part Extraction Using Operation Logs on Web Browser

Yoshinori Hijikata, Yoshinori Aoki, Younosuke Furui, Amane Nakajima

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Paper

TextExtractor: Text Part Extraction Using Operation Logs on Web Browser

Yoshinori Hijikata (IBM Research)
Yoshinori Aoki (IBM Research)
Younosuke Furui (IBM Global Services)
Amane Nakajima (IBM Global Services)

This paper introduces a text selection system called TextExtractor. TextExtractor extracts characteristic operations, which may occur according to the user's interest while the user is browsing a page from DOM (Document Object Model) events. It also extracts the text parts, which are the targets of the extracted operations from the whole text of the page. Users do not have the burden to explicitly input their interest keywords or rate pages they have browsed, because TextExtractor uses the operations which occur during the users' ordinary Web browsing. The hope is that using the extracted text will improve the performance of an information retrieval system, because TextExtractor eliminates many noise keywords.

Keywords: TextExtractor, Web browsing operation, user profile, operation extraction, text part extraction

1. Introduction

There are many search engine services on the Web that support users in acquiring their target information. When the user inputs some keywords as a search key, the search engine recommends pages that include the input keywords. The number of pages accessible by search engines has reached even one billion pages. Technologies for narrowing the number of search results are regarded as important, and many researchers have been working on these technologies. Devices that relieve users from needing special knowledge about search engines on the Web are important, because there are various kinds of users on the Web. Relevance feedback [(1)] is one such technology.

Relevance feedback is a method that (1) asks the user to indicate pages most relevant to his/her interests from the search results, and (2) searches again using keywords in those selected pages. Generally the selection of keywords influences the search results in search engines, and this method selects new keywords from the whole text of the relevant pages. Therefore this method has a problem in that not all the selected keywords have to do with the user's interests. Another problem is that it takes a lot of effort by the users to indicate suitable pages.

In this paper, as a solution for the first problem, we propose using only the text parts that the user might be interested in, instead of using the whole text of the page. As a solution for the second problem, we propose using the user's browsing operations to determine his/her interests instead of asking the user to explicitly indicate the pages that the user had an interest in.

We focus on the situation that the user uses a mouse as an input device while browsing Web pages and solve the above-mentioned problems by the following method:

(1) Extract specific operations that might occur as a result of the user's especial interest from the user's ordinary mouse operations while browsing pages.

(2) Extract by sentence or line the text parts that are the targets of those extracted operations.

We expect the following results by using this method:

(1) The system can automatically find the keywords relevant to the user's interests without requiring any special efforts by the user.

(2) The system can eliminate many noise keywords, the keywords unrelated to the user's interests, from the texts used for relevance feedback.

In Section 2, we introduce related work that gathers the information about users' interests, that we call the "user profile". Section 3 describes the relationship between the users' interests and their mouse operations. Section 4 describes the prototype called "TextExtractor" which extracts the text parts that the user took an interest in, and explains the text extraction method. In Section 5, we evaluate the prototype. Finally, Section 6 offers some conclusions and outlines future work.

2. Related work

Relevance feedback acquires user profiles by asking them to indicate pages relevant to their interests [(1)]. Some researchers have also been working on the acquisition of user profiles in the area of information filtering. Information filtering screens the information coming into the user, while information retrieval supports the user in finding information from remote databases [(2)]. This section describes the existing acquisition methods for user profiles from the research areas of information retrieval and information filtering.

There are two basic kinds of methods for acquiring user profiles [3].

(1) Explicit (Direct) method:

This method acquires user profiles by (1) asking users to answer preliminary questionnaires about topics or keywords which they are interested in, or (2) asking users to grade the pages they have browsed for interest and relevance. Ringo [4] and SIFT [5] use the former approach. GroupLens [6], Syskill & Weber [7] and NewsWeeder [8] use the latter approach. The advantage of this method is that it gives reliable results because it acquires the user profiles directly from the users. However these approaches also have some disadvantages. Generally, completing a preliminary questionnaire sufficiently detailed to allow a user to adequately describe his/her interest as keywords is a troublesome task, and grading pages also takes a lot of efforts from the users.

(2) Implicit (Indirect) method:

This method acquires user profiles by estimating the users' degree of interest in the pages the users have browsed based on such factors as (1) the time spent reading the pages (browsing time) [9] or (2) the mouse operations performed while reading the pages (browsing operations) [10] [11]. The advantage of this method is that it does not require any mental efforts by the users. One of the problems with the method using browsing time is that the system usually cannot know when the user opens a page and then starts doing some other work or leaves the PC. Existing research on the method using mouse operations monitors for such actions as when the user pushes a button for enlarging an article in a news system or when the user scrolls the window which displays the article. However these kinds of operations are application-specific operations. Therefore there is a problem that the trigger operations for acquiring the user profiles do not always happen while viewing general Web pages. There is also no attempt to extract the text parts that are the targets of the operations.

Our research can be classified with the implicit methods because it estimates the users' interests based on the browsing operations. It differs from the existing research in that (1) it estimates the user's interest from the ordinary mouse operations performed in general Web pages not from application-specific operations and (2) it also extracts the text parts that are the targets of the operations that might be associated with the users' interests.

3. Users' interests and mouse operations

We've examined the relationship between operations performed by users while browsing Web pages, and the users' real interests or lack of interest in the pages [13]. This showed that users tend to perform the following kinds of operations when they are interested in the pages they are browsing. (We eliminated operations to directly specify the targets of the users' interests such as inputting some keywords that the user is interested in into the text field of a search engine.)

- Text selection: Selecting text by dragging the mouse.
- Text tracing: Moving the mouse pointer along a sentence while reading.
- Link clicking: Clicking on a link to move to another page.
- Link pointing: Positioning the mouse pointer on a link, but not clicking the link.
- Scrolling: Scrolling a window at a certain speed.
- Bookmark registration: Registering a page as a bookmark.
- Saving: Saving an HTML file.
- Printing: Printing a page.
- Opening of a page in a new window: Clicking the right mouse button on a link and opening the linked page in a new window.

Out of these operations, the operations whose targets can be text are text selection, text tracing, link clicking, link pointing, and opening of a page in a new window. In this research, we focus on extracting text selection, text tracing, link clicking, and link pointing operation from browsing operations. We excluded the opening of a page in a new window operation because the implementation method used in our system, which is explained in Section 4, can not detect this operation, and the frequency with which users perform this operation is less than normal link clicking operation.

4. TextExtractor

We developed a system called "TextExtractor" which extracts the text parts the user might be interested in while the user is browsing a Web page. This section explains the system structure of TextExtractor and the operation events which are the source of operation extraction. After that it explains the operation extraction method and the text extraction method.

4.1 System structure The design concept of TextExtractor is that the system works on the user's regular Web browser such as Microsoft Internet Explorer by just changing some settings of the browser. The advantages of this design concept are (1) the user can use a familiar environment and (2) the system works even if the browser is updated with no change or with only small changes. We recognized that these advantages are important in order to offer a commercial service on the Web, such as an information retrieval or personalization service.

We developed the system using JavaScript and Java applet. A proxy server embeds the JavaScript code and the Java applet program for the system into the HTML file passing through the proxy server (see Figure 1.) This system consists of (1) an operation event detection module, (2) an operation extraction module, and (3) a text extraction module. These modules perform the following functions:

(1) Operation event detection module:

The operation event detection module, which is written as a JavaScript program embedded in

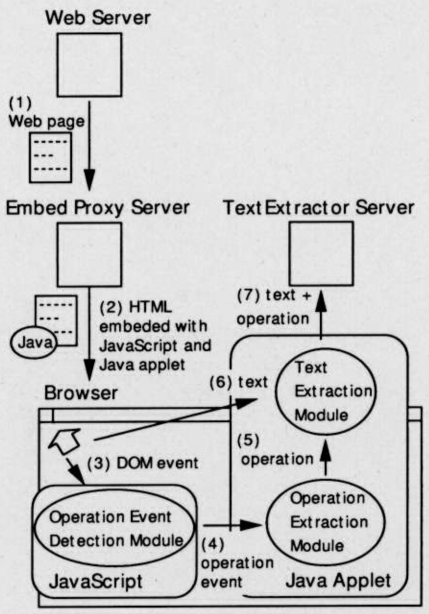


Fig. 1. TextExtractor System Structure

the HTML file, acquires DOM (Document Object Model)[14] operation events. It sends them to the operation extraction module with some parameters such as the coordinates. This event detection mechanism was actually developed for another system called Web Operation Recorder[12].

(2) Operation extraction module:
The operation extraction module analyzes the DOM operation events looking for specific operations that might occur as a result of the users' interests.

(3) Text extraction module:
The text extraction module extracts the target text part of the extracted operations based on the data such as the coordinates.

4.2 DOM operation event The following are the major DOM operation events:

(1) Focus: A focus event occurs when the user clicks an object and the object becomes ready to receive input data from the keyboard.
(2) Blur: A blur event occurs when the user clicks another object and the current object loses the focus.
(3) Move: A move event occurs when the user moves a window.
(4) Mousemove: A mousemove event occurs when the user moves the mouse pointer.
(5) Mousedown: A mousedown event occurs when the user pushes the mouse button.
(6) Mouseup: A mouseup event occurs when the user releases the mouse button.
(7) Resize: A resize event occurs when the user resizes the window.
(8) Scroll: A scroll event occurs when the user scrolls a window.
(9) Click: A click event occurs when the user clicks an object.
(10) Mouseover: A mouseover event occurs when the user moves the mouse pointer over an object.
(11) Mouseout: A mouseout event occurs when the user moves the mouse pointer off an object.
(12) Select: A select event occurs when the user selects text by dragging the mouse pointer.

The operation event detection module detects above-mentioned DOM operation events and sends them in the following format:

Format

operation time, operation event type, frame ID where the event occurred, object type involved in the event, and other data such as the coordinates

Examples of operation events are as follows:

Examples

```
936332393593,blur,frames(0),7,BODY
936332407468,focus,frames(0),7,BODY
936332410218,mouseover,frames(0),7,BODY,215,0
936332410265,mousemove,frames(0),7,BODY,215,0
```

It is not possible to detect clicking the right button of the mouse from the above-mentioned DOM operation events. Therefore we cannot recognize the opening of a page in a new window operation, as described in Section 3, from these DOM operation events. The frequency with which users perform this operation is less than normal link clicking operation. For this reason, we excluded the opening of a page in a new window operation from consideration in our research.

4.3 Operation extraction method The operation extraction module extracts operations from the DOM operation events in the following way:

(1) Text selection
The system regards the operation as a text selection when the mouseup event at the end of a select event occurs.

(2) Text tracing
First, the system detects continuous movement of the mouse pointer in a horizontal direction. For this detection, every time a mousemove event occurs, the system calculates the angle of the mouse movement relative to the horizontal and the time between the current mousemove event and the previous mousemove event. (For calculating the angle, the system uses the current mousemove event, and a mousemove event that occurred n-times before.) If the angle is below a threshold Ar and the time is below a threshold Tr, the system regards the movement as a continuous movement in a horizontal direction.
Second, when the system detects such a movement of the mouse pointer, the system calculates the distance and velocity of the movement. If the distance is longer than a threshold L and the velocity is slower than a threshold V, the system regards this operation as a text tracing operation.

(3) Link clicking
The system regards a click event on a link object as a link clicking operation.

(4) Link pointing
The system regards the operation as a link pointing operation when a mouseover event occurs on a link object, but there is no click event and load event afterwards, and a mouseout event

Table 1. Parameters for detecting text tracing.

Parameter	Value
Angle A_r(| $\tan\theta$ |)	0.25
Time T_r	750(msec)
Length L	40(pixels)
Velocity V	0.45(pixels/msec)
history n	2

occurs after a time Tp.

For the detection of a text tracing operation, the system calculates some parameters such as movement angle, time between mousemove events, movement length, and movement velocity. This is for eliminating noise operations. When the user moves a mouse pointer for a menu operation of the browser or an operation in another application, the mouse pointer may happen to move horizontally. For the detection of a link pointing operation, the system considers the time between the mouseover event and the mouseout event. This is also for eliminating noise operations. When the user moves a mouse pointer for another purpose, the mouse pointer may happen to cross over a link object.

There are differences among individuals in the text tracing operation. We heuristically set values shown in Table 1 as the standard parameters for the detection of text tracing operation. These are based on the record and observation of five users' Web browsing operations. We also set a value 750ms as the standard threshold Tp for the detection of link pointing operation in the same way.

4.4 Text extraction method

In Dynamic HTML[15], when a select event happens, the Web browser generates a selection object, which indicates the selected text. We use this object for extracting the text which is the target of the text selection operation. In Dynamic HTML, we can identify the location of a character, which is in specific coordinates, from the whole text of the page. We use this function for extracting the text which is the target of text tracing, link clicking, and link pointing operation. The text extraction method for each type of operation is as follows:

(1) Text selection
The system identifies the selection object and obtains the text which is indicated by the selection object.

(2) Text tracing
The system extracts text consisting of the line or sentence which includes the text in the area where the text tracing operation was performed. Concretely the system extracts a text which is (1) in a line on the average y-coordinates of the mousemove events while the text tracing operation is being performed, and (2) from the start point to the end point of the text tracing operation on the x-axis.

(3) Link clicking
The system extracts text consisting of the line or sentence which includes the character on the coordinates of the mouse pointer when the user clicks the link.

(4) Link pointing
The system extracts text consisting of the line or sentence which includes the character on the coordinates of the mouse pointer when the user points at the link.

For extracting a line or sentence of text, the system looks at one character in the candidate text and checks if it is a special character, such as a return, period, comma or space. When the system extracts text which is the target of text tracing operation, it also extract the text which exists on the line above the line where the mouse pointer is. This is because of what we discovered from the observations described in Section 4.3. When users read text doing text tracing operation, they move the mouse pointer but tracing the line exactly where they are reading. We found two cases: (1) they move the mouse pointer on a line within the line where they are reading, or (2) they move the mouse pointer on a line below the line where they are reading.

There is no function to distinguish text lines in Dynamic HTML. This system identifies the upper line in the following way:

(1) Record the character which the mouse pointer indicates and the characters which exist within m-characters before and after it.

(2) Move the search point by a few pixels up based on the mean coordinates of the mouse pointer.

(3) Record the character which is on the search point and the characters which exist within m-characters before and after it. Compare them with the characters recorded in Step (1).

(4) Determine that the search point is on the upper line if all the characters are same.

The reason why we use several characters to distinguish lines is to improve the confidence of identification. The reason why we select characters which exist within m-characters before and after the target is that the same word may appear in the line above the selected line. In this system, we use 8 as the parameter m. We decided on this value because there are not many words which have more than 16 letters.

5. Evaluation

5.1 Objective

This section evaluates the effectiveness of TextExtractor by considering the following:

(1) Is the extracted text something the user was interested in?

(2) Are the keywords in the extracted text related to the users' interests?

5.2 Experiment method

The experiment method we used is as follows:

(1) The subject searches for the pages he/she wants to browse in advance of the experiment.

(2) When the experimental observations begin, the subject freely browses the selected pages.

(3) Every time the subject moves from a page to another page, he/she answers a questionnaire about the previous page. In this questionnaire, the system displays all keywords extracted from the page, and the subject checks only keywords

he/she was interested in.

(4) The experimenter compares the keywords checked by the subject and the texts/keywords extracted by the system. The experimenter determines the effectiveness by calculating some parameters and executing an analysis of variance.

Figure 2 shows a sample window for the questionnaire. The list box area with a scroll bar displays all keywords extracted from the text of the page. The subject checks only the check boxes whose keywords are ones the user is interested in. For reducing the subjects' efforts in selecting keywords, there are some buttons to sort the keywords above this list box area. We built a proxy server with a morphological analyzer to extract the keywords in the page and generate the questionnaire.

We calculated the following evaluation parameters:

Parameter for text-level evaluation

(1) TextExtractor Text Precision
$=|A|/|B|$

Parameters for keyword-level evaluation

(1) Full Keyword Precision
$=|D|/|C|$

(2) TextExtractor Keyword Precision
$=(|D\cap E|)/|E|$

(3) TextExtractor Keyword Recall
$=(|D\cap E|)/|D|$

(4) TextExtractor Keyword Narrowing Rate
$=|E|/|C|$

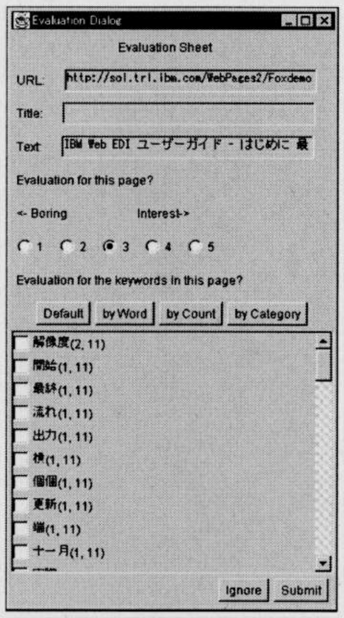
A, B, C, D, E in the above equations have the following meanings:

- A: The set of extracted texts which have at least one keyword that the subject has checked as an interesting one in the questionnaire.
- B: The set of extracted texts.
- C: The set of keywords included in the whole text of the page.
- D: The set of keywords that the subject has checked as interesting ones in the questionnaire.
- E: The set of keywords included in the extracted texts.

From the TextExtractor Text Precision, we can determine whether the texts extracted by TextExtractor are the parts the user was interested in. By comparing the Full Keyword Precision and the TextExtractor Keyword Precision, we can see how much the ratio of keywords that the user was interested in will be improved by TextExtractor. We can determine how much TextExtractor filters the text part from the whole text with the TextExtractor Keyword Narrowing Rate. We can also see how many keywords TextExtractor could extract from all keywords the user was interested in with the TextExtractor Keyword Recall.

In addition, we calculated the Full Keyword Precision and the TextExtractor Keyword Precision for all of the pages and looked to see whether there is a significant difference in keyword precision by executing an analysis of variance.

5.3 **TextExtractor evaluation** Five subjects (three women and two men in their twenties or thirties)


Evaluation Dialog

Evaluation Sheet

URL: http://sol.trl.ibm.com/WebPages2/Foxdemo

Title:

Text: IBM Web EDI ユーザーガイド - はじめに 最

Evaluation for this page?

<- Boring Interest->

1 2 [✓ 3] 4 5

Evaluation for the keywords in this page?

- 解像度(2, 11)
- 開始(1, 11)
- 最終(1, 11)
- 流れ(1, 11)
- 出力(1, 11)
- 横(1, 11)
- 個個(1, 11)
- 更新(1, 11)
- 端(1, 11)
- 十一月(1, 11)

Fig. 2. Questionnaire Window.

participated in the experiment. We used data from 120 Web pages for the analysis. Table 2 shows the average TextExtractor Text Precision for the four targeted operations. Although there is a spread of 20%, we can see that TextExtractor extracts the text that the user was interested in with high precision.

Table 3 shows the average value of Full Keyword Precision, TextExtractor Keyword Precision, TextExtractor Keyword Recall, and TextExtractor Keyword Narrowing Rate for the four targeted operations. By comparing the Full Keyword Precision and TextExtractor Keyword Precision, it turned out that the average TextExtractor Keyword Precision for all subjects (about 7.2%) is five times of the average Full Keyword Precision (about 1.5%). From this we can see that TextExtractor extracts the keywords that the subjects were interested in.

The average TextExtractor Keyword Narrowing Rate for the five subjects is about 7.8%. Therefore it removed at least about 92% of the keywords the subjects were not interested in. The average TextExtractor Keyword Recall for all subjects is about 37%. From this it appears that TextExtractor removes more than 90% of the keywords that the subjects were not interested in and preserves about 40% of the keywords the subjects were interest in.

Table 4 shows the results of analysis of variance. Out of the five subjects, there is a significant difference for two subjects at the 5% level of significance and for two subjects at the 1% level of significance, although there is not a significant difference for one subject. From this it appears that TextExtractor can be effective for many users.

5.4 **Discussions** The experiment showed that TextExtractor removes more than 90% of the noise keywords that the users were not interested in and preserves about the 40% of the keywords that the users were in-

Table 2. Text precision.

Subject	TextExtractor Text precision(%)
Subject A	53.8
Subject B	75.0
Subject C	56.1
Subject D	61.3
Subject E	67.6
All subject	60.7

Table 3. Keyword precision, recall, and narrowing rate.

Subject	All Keyword Precision (%)	Text Extractor Keyword Precision (%)	Text Extractor Keyword Recall(%)	Text Extractor Keyword Narrowing Rate(%)
Subject A	1.86	6.36	33.3	9.74
Subject B	0.87	7.87	31.7	3.50
Subject C	1.99	8.81	38.2	8.62
Subject D	0.76	4.74	48.1	7.76
Subject E	2.51	6.83	38.9	14.3
All subject	1.52	7.18	37.0	7.83

Table 4. Result of analysis of variance.

Subject	Unbiased variance ratio	F(5%)	F(1%)	Significant difference
Subject A	5.33	4.6	9.33	Yes (5% level)
Subject B	7.24	4.84	9.65	Yes (5% level)
Subject C	11.3	4.24	7.77	Yes (1% level)
Subject D	14.5	4.75	9.33	Yes (1% level)
Subject E	3.07	4.38	8.18	No
All subject	22.28	4.17	7.56	YES (1% level)

terested in. Searching Web pages using relevance feedback based on keywords extracted by TextExtractor is expected to result in better efficiency and higher precision.

In the experiment, five subjects browsed general Web pages on the Internet as usual. TextExtractor extracted the text parts the subjects were interested in based on the operations performed in such browsing. From this we can see that such a system can acquire user profiles without the users' special efforts to answer questionnaires or grade the pages they have browsed. Therefore we believe that TextExtractor makes users feel free to use such functions as relevance feedback.

One drawback is that the current implementation of TextExtractor extracts specific operations using common parameters for all users. However there are differences among individuals in their browsing operations, and therefore there are limitations on text and keyword precision and recall using such uniform parameters for all users. If we can ask users to perform sample operations and acquire information about such individual variations, we can improve the performance by changing parameters and doing user-adapted operation extraction.

6. Conclusions

This paper describes a text selection system called TextExtractor. This system identifies specific operations that may occur according to the users' interests during ordinary browsing operations. After that it extracts the text parts which are the targets of these significant operations from the complete text of the Web pages.

TextExtractor has been implemented using Java and JavaScript, and basic experiments have been conducted with it. Five subjects participated in this experiment and browsed some Web pages on the Internet as usual. The result of the experiment shows a high probability that the text part extracted by TextExtractor is the part the subjects were interested in. It also shows the ratio of keywords that the subjects were interested in, in relation to all keywords was higher in the texts extracted by TextExtractor than in the whole text. From these result we confirmed that the system can acquire the user profiles without requiring users to answer questionnaires or grade pages they have browsed.

Future research will involve methods for dealing with individual variations and examine the change of performance of relevance feedback based on texts extracted by TextExtractor. We also plan to investigate the browsing operations performed in small devices such as mobile phones and PDAs.

Acknowledgements

We would like to thank Hiroshi Nomiyama of IBM Research for his generosity in providing his morphological analyzer. We also acknowledge the participation in the experiment of staff members of IBM Research.

(Manuscript received xxx , , revised xxx ,)

References

(1) Salton, G.: Automatic Text Processing: The Transformation , Analysis, and Retrieval of Information by Computer, *Addison-Wesley*, 1989

(2) Belkin, N. J. and Croftk, W. B.: Information Filtering and Information Retrieval: Two Sides of the Same Coin?, *Comm. of the ACM, Vol.35, No.12, pp.29-38*,1992

(3) Sugimoto, M.: User Modeling and Adaptive Interaction in Information Gathering Systems, *Journal of Japanese Society for Artificial Intelligence, Vol.14, No.1, pp. 25-32*, 1999

(4) Shardanand, U. and Maes, P.: Social Information Filtering: Algorithm for Automating 'Word of Mouth', *Proceedings of CHI'95, pp. 210-217*, 1995.

(5) Yan, T. W. and Garcia-Molina, H.: SIFT - A Tool for Wide-Area Information Dissemination, *Proceedings of 1995 USENIX Technical Conference, pp. 177-186*, 1995.

(6) Resnick, P., et al.: GroupLens : An Open Architecture for Collaborative Filtering of Netnews, *Proceedings of C-SCW'94, pp.175-186*, 1994.

(7) Pazzani, M. and Billsus, D.: Learning and revising user profiles: the identification of interesting web sites, *Machine Learning, 27(3), pp. 313-331*, 1997.

(8) Lang, K.: NewsWeeder: Learning to Filter NetNews, *Proceedings of ICML'95, pp. 331-339*, 1995.

(9) Morita, M. and Shinoda, Y.: Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 272-281*, 1994.

(10) Sakagami, H. and Kamba, T.: Learning Personal Preferences on Online Newspaper Articles from User Behaviors, *Proceedings of the Sixth International World Wide Web Conference, In Computer Networks and ISDN Systems, 29, pp. 1447-1456*, 1997.

(11) Yan, T. W. et al.: From User Access Patterns to Dynamic Hypertext Linking, *Computer Networks and ISDN Systems, 28, pp. 1007-1014*, 1996.
(12) Aoki, Y., Ando, F., and Nakajima, A.: Web Operation Recorder and Player, *Proc. of International Conference on Parallel and Distributed Systems (IEEE ICPADS2000), pp.501-508*, 2000
(13) Hijikata, Y.: Estimating a User's Degree of Interest in a Page during Web Browsing, *1999 IEEE International Conference on Systems, Man, and Cybernetics, No.IV, pp.105-110*, 1999
(14) http://www.w3.org/DOM/
(15) D. Goodman: Dynamic HTML The Definitive Reference, *O'Reilly*, 1998