The Thirteenth International Symposium on Artificial Life and Robotics 2008(AROB 13th '08),
B-Con Plaza, Beppu, Oita, Japan, January 31-February 2, 2008

An Intelligent Web Information Searching Technology

Young Im Cho
Dept. of Computer Science, University of Suwon
San 2-2, Wau-ri, Bongdam-eup, Hwaseong, Gyeonggi-do, KOREA, 445-743
e-mail : ycho@suwon.ac.kr

Abstract

In this paper, we propose an intelligent web information searching technology. The searched documents are reordered by the proposed technology according to the user profile. The user profile is updated by the user's behaviors. If a user interests the searched papers, the keywords in the searched documents are updated in the user profile. As a result, the only documents having the interested keywords are retrieved in the system as a personalized method.

I. Introduction

Artificial Intelligence (AI) is the area of computer science focusing on creating machines that can engage on behaviors that humans consider intelligent.

With the flood of information through World Wide Web, users usually spend heavy surfing time to find relevant information or web pages. To reduce users' burden, various information retrieval engines such as Google, Yahoo, and AltaVista have been developed. Even though those engines reduce users' effort and time to find appropriate information, the web page list retrieved from the engines is still not enough for user effort free. Even though a query is provided on the same search engine by different users, the intention of search for each user can be different according to user profile. However, existing search engines provide identically the found list to all users. It is needed to reorder web page list according to user profile.

In this paper, we propose an algorithm of reordering web pages retrieved from a search engine which reorders web page list according to user's profile. The proposed algorithm builds up user's individual keyword database from his/her query history on search engines. The qualification of keyword is words except prepositions and articles. If a word is appeared more than a predefined appearance frequency, called threshold, in a web page, the word is stored into database as a keyword. Then, we provide weights to the keywords according to the rank of web pages containing the keywords. Our system reorders the web pages retrieved from the search engine by overall keyword weights of keywords included in each web page. In this paper, we will explain the proposed web information searching system in section 2. And the simulation results are explained in section 3. And finally we will conclude in section 4.

2. Web Information Searching System

There have been a lot of works on web page ranking of information retrieval systems. Those works can be classified into two techniques: one is done based on link of pages and the other is based on keywords[1].

However, the link-based techniques have a difficulty in making data structure for the links. So, many researches for search systems have been done based on keyword search[2-6].

Even though many researchers developed various search algorithms to save the effort and time for users to find appropriate information, the web page list retrieved using their algorithms is still not enough for user effort free. It is because the search target for each user is usually different according to user profile. Thus, it is needed to reorder web page list according to user profile. From the following section, we demonstrate an algorithm of reordering web pages retrieved from a search engine.

Fig. 1 shows the overall process of our system. If a user requests a query to a retrieval search engine, the engine shows the list of web pages having relevance with the query.

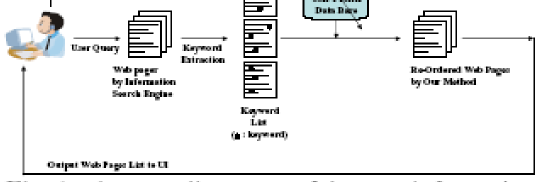
User Behavior
(Downloading, Clicking, No-action)
Keyword Weight Update
User Profile Data Base
Output Web Page List
User Query
Web pages by Information Search Engine
Keyword Extraction
Keyword List
(● : keyword)
Re-Ordered Web Pages by Our Method
Output Web Page List in UI

Fig. 1. The overall process of the wen information searching system

Keywords can be extracted from the web pages. In our previous work [9], the keyword was determined by the appearance frequency, denoted as $freq_i$, of word i in the web page. If $freq_i$ is greater than a threshold, then the word is eligible for a keyword, otherwise it is not. The optimal threshold is determined by exhaustive empirical experiment according to types of web pages.

The position of web pages in a searched list indicates importance or relevance of the web pages with the query. For instance, if a web page is high ranked, it means the web page is usually highly relevant with the query. Also, the keywords in the high ranked web pages can be considered as high relevant to the query. From the point of view, we provide the weight to each keyword according to the rank of web pages including the keyword and store it into the user profile database.

The structure of the database consists of the index of keyword, keyword, and weight of the i^{th} keyword k_i, denoted as w_i.

Keyword weight w_i is initialized by the rank of web pages containing keyword weight w_i in the searched list in the sense of respecting the ability of search engine. In a list, any keyword can be appeared in multiple web pages. The weight value of keyword i, w_i, can be expressed in a numerical form as Equation (1).

$$w_i = \sum_j \frac{N - R_{i,j} + 1}{N} \tag{1}$$

where N is the total number of web pages in the searched list and $R_{i,j}$ ($j \in N$) is the rank of the j^{th} web page including keyword i. From Equation (1), the value of w_i increases exponentially as the number of high ranked web pages increases.

To avoid the problem, we normalize keyword weight w_i with the log sigmoid function.

Using the normalized weights, we compute the overall relevance of each web page j with user query, denoted as Rev_j, obtained by the summation of the weights of keywords in the web page j as seen in Equation (2).

$$\mathrm{Re}\,v_j = \sum_{k_i \in K_j} \tilde{w}_i \tag{2}$$

where K_j means a set of keywords included in web page j. By the value of each Rev_j, the web pages from the search engine is reordered from large to small value.

Existing search engines and the algorithm explained in the previous section provide the same ordered list for the same user-provided query even though users' current interest is different. In general, users' search targets can be different according to user profile. In order to take into consideration, we modify the keyword weights by user profile which can be obtained from user behavior for the searched web pages. User behaviors such as downloading, clicking, and no-action on a web page can be a measure of the user's interest of the keywords in the web pages. For instance, the action of downloading a web page from the searched list is understood that the user is very interested in the web page. From this point of view, the keywords with downloading action can be considered as preferred, compared to the keywords in the web pages on which the user just clicks or does no-action.

From the above explanation, the normalized weight $\tilde{w}_i$ is updated according to user behavior on the web pages including keyword i as Equation (3).

$$\tilde{w}_i = \tilde{w}_i \cdot (1 + u) \tag{3}$$

where u is the numerical value of user behaviors. The magnitude of u for downloading is greater than clicking, and that of clicking is greater than that of no-action. As seen in Equation (3), the optimal value of u for each user behavior can critical for updating keyword weights. The optimal values of u are obtained from empirical experience in the experimental section.

3. Simulation Results

In this section, we demonstrate our system using a user's search information on Google search engine during one week and show our search list on a client system implemented by Visual C++ in the Window environment.

The experimental conditions are as follows: (a) Only noun is regarded as keyword in this experiment for the convenience. (b) We make the 10 tested collection file (total web pages or documents) per query using 6 queries{Mashup, long tail, rss, blog, ajax, wiki}, respectively. So, totally 60 web pages(=documents) exist in the tested collection file for the simulation. And we acquired that keywords appearing 4 times usually trend to be relevant to user's query from exhaustive examination. (c) We set the variable s in the log sigmoid function for normalizing keyword weights computed using Equation (1) equals to 0.2 from our empirical experience. (d) After the retrieval, if user acts some action about the web pages such a download, click, no-action etc. the system automatically increases the keyword's weights in the corresponding web pages, 0.2, 0.1, 0.0 respectively.

The simulated process is explained here in detail.

(1) If a user query to the system using a query 'mashup', the information retrieval results in the traditional method(=vector method) are showed in Fig.2. Where, $D_i (i \in Collection)$ is the ith document in the results.

$R_{i,j}$ =1, D_1={<u>mashup</u>, registration, identity, conference …} $R_{i,j}$ =2, D_2 = { <u>mashup</u>, mashups, days, raquo, popular, matrix, apis …} $R_{i,j}$ =3, D_3 = { discuss, <u>mashup</u>, com, webmashup, bury, web, tags, mashups ….} Here, the underline means that the keyword.

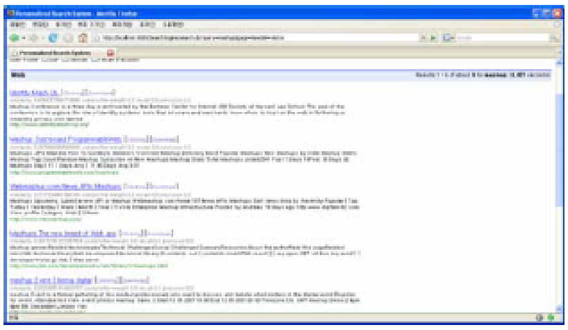
Fig.2. The result of the traditional method(vector)

(2) The user profile is in Table 1. The initial weights are changed according to user's behaviour as shown in the Table 1. Table 1 is a part of the user profile in this system

(3) According to the Table 1., the reordered result is shown in Fig.3.

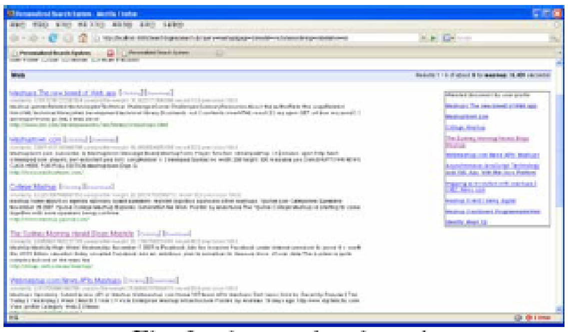
Fig. 3. The reordered result

(4) The high ranked 4 keywords, such as 'mashup', 'mashups'. 'apis'. 'com' are extracted in the user profile which is composed of keywords like in Table 1, and then it makes the relative documents using the keywords.

(5) Therefore, the relative documents are obtained the following lists. The list is shown in the right part of Fig.4.

(6) If a user query an 'ajax' in this system, after a user profile is applied to the system, the result is changed shown in Fig.4.

The recall and precision reatios at the retrieved documents are shown in Fig. 5 and Fig.6, respectively

Table 1. The weighted keyword list (omitted)

keyword	repeated_no	init_weight	changed_weight
mashup	154	0.98201379	1.555509843
mashups	138	0.96862879	1.146590003
apis	34	0.791391473	1.044636744
com	81	0.825714679	0.990857615
news	44	0.809141956	0.970970347
days	26	0.660756369	0.872198407
raquo	28	0.660756369	0.872198407
google	33	0.708660825	0.85039299
web	102	0.708660825	0.85039299
map	22	0.708660825	0.85039299
identity	13	0.660756369	0.792907643
enterprise	13	0.609317542	0.73118105
api	22	0.609317542	0.73118105
read	32	0.609317542	0.73118105
search	17	0.609317542	0.73118105
view	20	0.582570206	0.699084248
posted	51	0.527749235	0.633299082
registration	8	0.5	0.6
conference	12	0.5	0.6
blog	16	0.5	0.6
http	20	0.472250765	0.566700918
www	19	0.472250765	0.566700918
profile	16	0.444671945	0.533606334
bury	16	0.444671945	0.533606334
link	16	0.444671945	0.533606334
category	16	0.444671945	0.533606334

…………

Fig.4. The reordered result using user profile

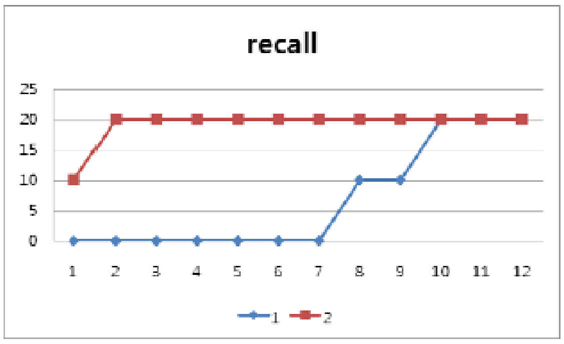

Fig. 5. The recall ratio

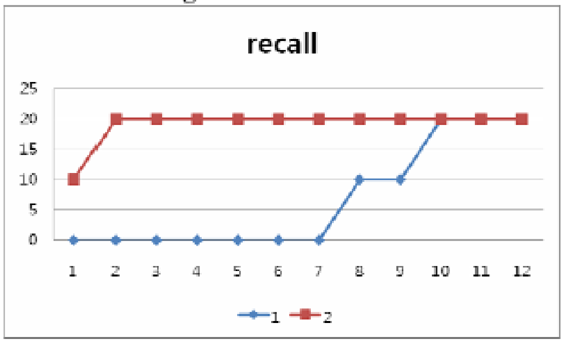

Fig. 6. The precision ratio

The '1' and '2' mean that the case the system using user prpfile and the case the system with user profile, respectively. The case without user profile, the average recall ratio is 6.67, but in case of with user profile, the average recall ratio is 19.17. As a result, 288% is raised in case of with user profile. Also, the case without user profile, the average precision ratio is 6.54, but in case of with user profile, the average precision ratio is 43.39. As a result, 663.5% is raised in the case of with user profile. Generally, as the higher precision ratio, the performance is getting higher. So, the 43.39% precision ratio in our system shows the high performance result more than the traditional method which is in case of without user profile.

The weight changing process shown in Table 1. is the followings using equation(1). The changed weight is in the right part in Table 1.

i) Initial: Sigmoid function for normalization of weight:

$$f = 1 / 1 + e ** -(\alpha - \theta) \quad (\theta = 1.0)$$

ii) Initial weight setting process:

Weight Keyword ser $K=\{K_1, K_2, \ldots K_n\}$

K_1= 'mashup':

W_1 = (9-1+1)/9 + (9-2+1)/9 + (9-3+1)/9 + (9-4+1)/9 + (9-5+1)/9 + (9-6+1)/9 + (9-7+1)/9 + (9-8+1)/9 + (9-9+1)/9 = 5 : The normalized w_1 = 0.9820

K_2='mashups',

W_2 = 0 + 8/9 + 7/9 + 6/9 + 0 + 4/9 + 0 + 0 + 1/9 = 26/9 = 2.8889 : The normalized weight w_2 = 0.8686

K_3= 'com' :

W_3 = 0 + 0 + 7/9 + 6/9 + 0 + 4/9 + 3/9 + 2/9 + 1/9 = 23/9 = 2.5556 : The normalized weight w_3 = 0.8257

K_4= 'news'

W_4 = 0 + 0 + 7/9 + 6/9 + 0 + 4/9 + 3/9 + 2/9 + 0 = 22/9 = 2.4444 : The normalized weight w_4 = 0.8091

All of keywords are evaluated .($K_i \in K$)

iii) The weight changing process:

Let, $K_{ij} \in D_j$ Keyword i is in a document or web page D_j. If a user download D_j, u = 0.2, so, the changed keyword weight is like the following process.

K_{11} = 'mashup' w' = 0.9821 * (1 + 0.2) = 1.1784. Here, w' means the changed weight.

K_{21} = 'identity', w' = 0.6608 * (1 + 0.2) = 0.7929

K_{31} = 'registration', w' = 0.5 * (1+0.2) = 0.6

K_{41} = 'conference', w' = 0.5 * (1+0.2) = 0.6

If a user click a D_2, u = 0.1, so, the changed keyword weight is like the following process.

K_{12}='mashup', w' = 1.1784 * (1+0.1) = 1.2964

K_{22}= 'mashups', w' = 0.8686*(1+0.1) = 0.9555

K_{32}='days', w' = 0.6607*(1+0.1)=0.7268

K_{42}='apis', w' = 0.7914*(1+0.1)=0.8705

K_{52}='popular', w' = 0.4722*(1+0.1)=0.5195

K_{62}='matrix, w' = 0.4722*(1+0.1)=0.5195

...

The remainder processes are the similar process, so we omitted here the changed process in detail.

4. Conclusion

In this paper, we proposed the web information searching system. Each keyword weight is updated according to user's behaviors on web pages. From the simulation result, our system can be a useful tool for saving surfing time and effort to find users' preferred web pages.

However, the values of variable u users' behaviors, i.e., u = 0 for no-action, u = 0.1 for clicking, and u = 0.2 for downloading were obtained from the exhaustive empirical experiment. The values might not be optimal for all users. As one of further works of our system, it is needed to develop an automatic algorithm for determining the values for each user.

References

[1] Baeza-Yates, R., Davis, E.: Web Page Ranking using Link Attributes. International World Wide Web Conference, pp.328-329, 2004

[2] Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword Proximity Search on XML Graphs. Proc. International Conference on Data Engineering , pp.367-378, 2003

[3] Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: A System for Keyword-based Search over Relational Databases. Proc. International Conference on Data Engineering, pp.5-16, 2002

[4] Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword Searching and Browsing in Database using BANKS. Proc. International Conference on Data Engineering, pp.5-16, 2002

[5] Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Database. Proc. The 2005 ACM SIGMOD International Conference on Management of Data, pp.527-538, 2005

[6] Guo, L., Shao, F., Botev, C., Shanmugasundaram: XRANK: Ranked Keyword Search over XML Documents. SIGMOD, 2003