IMPLICIT CLIENT SIDE USER PROFILING FOR IMPROVING RELEVANCY OF SEARCH RESULTS

Saniya Zahoor[1] and Dr. Mangesh Bedekar[2]

[1] M.E., IIIrd Sem, Computer Engineering Department, MAEER'S MIT, Pune, India
saniyazmalik@yahoo.com
[2] Associate Professor, Computer Engineering Department, MAEER'S MIT, Pune, India
mangesh.bedekar@gmail.com

ABSTRACT

The Web is being a pool of knowledge, where any user visits hundreds of pages for various purposes but keeping track of its relevance for him is a tedious job. An average browser just provides you by the details of your browsing history but has no way to determine what importance the page holds for the user. In this paper we propose a method which aims to generate user profiles automatically depending on the various web pages a user browses over a period of time and the user's interaction with them. This automatically generated user profile assigns weights to web pages proportional to the user interactions on the webpage and thus indicates relevancy of web pages to the user based on these weights.

KEYWORDS

User profiling, web personalisation, implicit user behaviour modelling, client side analysis

1. INTRODUCTION

The Internet has made information available to humankind in a quick, easy, publicly accessible manner which is within reach of one and all. Today it's the first resource any user turns to when he needs any form of information. A multitude of web pages exist on any topic and the number grows with each passing day. The task of finding relevant information from the corresponding web pages is a tedious task. Search engines are available for the same but the problem of ever increasing data reduces the efficiency of the search results performed by a user. Even if the user manages to find a relevant page corresponding to his search query, retaining and remembering the webpage, or storing it efficiently for future references is another task not very well achieved. Identifying the relevance of a web page to a user thus becomes a very challenging task. Interest indicators abound but doing so implicitly is a challenge for any search system.

In this paper we propose a method which studies the user's behaviour on different web pages from search results. The method capture various user interest indicator parameters like, the mouse movement, mouse scroll, time spent on webpage and user actions like save, print and bookmark the webpage, which all users do unknowingly and analyse it to determine the relevancy of the webpage for the user. Once the relevancy is determined, ranking of the pages is done and stored in a database which could be referred to, for any similar search query by the same user in the future.

Whenever a user gives a search query, the method scans the data stored in the database, with respect to the relevance of web pages, and gives the relevant search results from previous search queries, followed by other search results which would have come from the search engine normally.

We have proposed a technique of ranking web pages for a user according to its relevance to him by capturing six parameters all done on the client side. All this data is captured when the user accesses any webpage in the browser on his machine. This captured data is then analysed and relevance inferred. A result of testing our method proves that browsing for a user can be made more personalised and effective.

The main factors considered in this profiling are active time spent, pixels covered by the mouse pointer, vertical and horizontal scroll on a page and three others factors like save, bookmark and print. These six factors use statistical tools to allot weight to a given web page. We have made an extension to the formula already proposed in the paper [Teevan et. al., 2005]. We have also added factors like saving a web page, bookmarking a web page and printing a web page which are relevancy indicators too and explained the relevance factor in the calculations.

2. RELATED WORK

Many researchers have done work of allotting weights to the pages visited. This was primarily done on relating the number of pages a user visits and number of pages he finds relevant for a given search.

The method proposed by [1] ranks documents by summing over terms of interest the product of the term weight (wi) and the frequency with which that term appears in the document (tfi). When no relevance information is available, the term weight for term i is,

$$w_i = \log\left(\frac{N}{ni}\right)$$

Where N is the number of documents in the corpus, and ni is the number of documents in the corpus that contain the term i.

When relevance feedback information is available, two additional parameters are used to calculate the weight for each term. R is the number of documents for which relevance feedback has been provided. ri is the number of these documents that contain the term. The term weight in traditional feedback is modified as,

$$w_i = \log\left(\frac{(ri+0.5)(N-ni-R+ri+0.5)}{(ni-ri+0.5)(R-ri+0.5)}\right)$$

The methods proposed give relevancy of web pages to the user where both, feedback are un-available and whence relevance feedback is available.

In the earlier mentioned techniques, relevance of web pages to the user was gathered explicitly, by asking the user to rate the search results. Rating any web page explicitly after each search is a tedious task and is not in line with the usual behaviour of the user.

A lot of literature is available which speaks of how interest indicators can be gathered implicitly. [2] have compared web retrieval systems with explicit v/s implicit feedback. In their experimentation they show that the implicit feedback systems can indeed replace explicit feedback systems with little or no effect to the users search behaviour or task completion. [3] proposed to combine some well known interest indicators to get better results for relevancy of web pages, they also propose several more implicit interest indicators. [4] proposed a system to identify users' interest based on his behaviour in the browser he uses to access the Internet. They also go on to show that change in users interests can be handled by the system.

Considering the improvements suggested by the systems as mentioned above, we proposed modifications to the existing mechanism to infer interests of users on web pages implicitly by considering all the activities the users performs in the browser when he accesses a web page and to infer relevancy from it.

3. MODIFICATIONS PROPOSED

In our method we have proposed a relevance factor to determine if a web page is relevant or not, and if relevant how much. For this we sort pages in order of their relevant factors and only the top few pages (based on a threshold, to be taken from the user) are said to be relevant. The activity by the user on the webpage can be inferred by the active time spent by the user, mouse movement and scrolling behaviour on the webpage as indicated in [5] [6] [7] [8].

The relevance factor, Rf is, calculated by,

$$R_f = \log\left(\frac{(\text{time})\times(\text{movement})}{\text{scroll}}\right)$$

Time is the active time spent by a user on the web page. Active time is defined as the total time for which the user was on that page. The moment he switches to another tab / another window (the focus of the current open tab is lost) the timer stops and resumes only if and when the user comes back to the same tab. Time is calculated in seconds.

Movement is the number of pixels a user has moved over on the web page through his mouse pointer. Mouse movement hence suggests the importance of a web page to the user. Scroll is the total area of the web page scrolled by a user (horizontal scroll as well as vertical scroll),

$$\text{Scroll} = (V + v) \times (H + h)$$

V = height of the page
v = vertical scroll
H = width of the page
h = horizontal scroll

The other factors included are bookmarking [9], saving and printing [10] the web page. These factors are indicators that the web page contained relevant information for the user so much so that he might want to refer it in future and hence made a copy of a reference to it on his browser (machine). However these factors have different relevancies based on their inherent characteristics as described ahead.

Bookmarking – This is the most efficient way of keeping a reference to the web page for future. Bookmarking is able to handle all the changes which might happen on the web page as only a reference of web page is stored. This factor is considered better than save and print as the other two factors do not handle any changes which might have happened on a web page over a period of time. Bookmark is also able to direct user to the web page in case the address changes over a period of time unlike saving and printing. Compared to the other two we allot a weight of ‘5 out of 10’.

Saving – Saving creates a copy of the web page on the user machine. Users prefer saving over printing as it saves paper and can be easily shared or edited. Though saving a page takes up memory space, it is still more preferable over printing. However saving cannot handle any changes made on that web page after it is saved. We allot a weight of ‘3 out of 10’.

Printing – This is another way how a user makes a permanent copy of the web page for future use. The user prints the web page from the browser if found important. However it is supposed to be a dead piece of information which doesn’t incorporate any form of changes made to the web page after it gets printed. It also can’t be trusted as there is no information about its source. Editing a printed document is difficult. Printing is also avoided by users as it consumes paper or a printer is not always available close by. We allot a weight of ‘2 out of 10’.

The moment the user saves and/or prints and/or bookmarks a web page the script running on the user’s machine (client side), captures these actions and allots the relevant weight to the factor. These factor otherwise have a default weight of ‘1’. The importance of the accessed web page is determined by the user’s action implicitly without ever asking him about the same explicitly.

We modify the relevance factor correspondingly to accommodate these user traits. The modified relevance factor is as mentioned below,

$$R_f = \log\left(\frac{(\text{time})\times(\text{movement})}{\text{scroll}}\right) * \text{save} * \text{print} * \text{bookmark}$$

4. DESIGN CONSIDERATIONS AND IMPLEMENTATION DETAILS

The proposed method was implemented in the manner as explained below.

The implementation starts with installation of WAMP (Windows, Apache, MySQL, PHP) server on the client’s machine and addition of a particular database with a table containing specific columns namely the URL, active time, mouse movement, scroll, save, print and bookmark. Here active time stores the value in seconds, mouse movement stores the value in pixels, scroll saves the value in pixels square and save print and bookmark stores the values of corresponding weights if the action occurs or it stores zero.

Once the server has been installed with the required database and table, every time the user logs into a browser he needs to switch on the server after which the method starts getting implemented as the data starts getting stored in the database after which its analysis starts.

Greasemonkey is a Mozilla Firefox extension that allows users to install scripts that make on-the-fly changes to web page content after or before the page is loaded in the browser (also known as augmented browsing).The changes made to the web pages are executed every time the page is viewed, making them effectively permanent for the user running the script. Greasemonkey can be used for customizing page appearance, adding new functions to web pages (for example, embedding price comparisons within shopping sites), fixing rendering bugs, combining data from multiple web pages, and numerous other purposes. With the help of greasemonkey we installed out script in all the web pages a user visits through Mozilla Firefox so that we could capture all the factors needed for the methods implementation.

Mozilla Firefox Browser is free and open source, its features include tabbed browsing, spell checking, incremental find, live bookmarking, smart bookmarks, a download manager, private browsing, Functions can be added through extensions, created by third-party developers, of which there is a wide selection, a feature that has attracted many of Firefox's users. One of the main reasons why Mozilla Firefox was used was because of its unique add-on Greasemonkey.

JavaScript (JS) is an interpreted computer programming language. It was implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. Javascript can be imbedded in the html page and have pre-defined function which help in capturing various data from the web page. Javascript along with tools of PHP and AJAX capturing the data needed for the method and directs it to the database where it gets stored.

WAMP is a form of mini-server that can run on almost any Windows Operating System. WAMP includes Apache 2, PHP 5 and MySQL (phpMyAdmin and SQLitemanager are installed to manage your databases) preinstalled. The WAMP server needs to be hosted every time the script is run. The javascript directs the data captured to the PHP page which stores the data in the corresponding table of the database in the WAMP server.

5. SCRIPT DESCRIPTIONS

The important script written in greasemonkey ensured that the moment the page loads the script gets implemented in all the web page. Hence this assures that the script runs in all the pages a user visits.

The scripts are modularised and handle user events. As soon as the webpage is completely loaded, a script starts calculating active time. Another script handles mouse movements. It returns the total mouse movement on the page. A third script handles any scrolls on the page. It handles both vertical and horizontal scroll. The counter decrements if there is any backward or upward scroll, the counter decrements. This gives the total scroll on the web page.

Three more user events, namely save, print and bookmark have been defined whenever the user saves prints or bookmarks a page. This script handles what all actions are performed by the user on the webpage.

The method as explained above was implemented on a Firefox browser using tools like html, PHP, JavaScript, JQuery, AJAX on Greasemonkey (an add-on available on Mozilla Firefox which allows to customize the way a web page displays or behaves, by using small bits of JavaScript) and hosted by WAMP server. The database was stored and retrieved from MYSQL.

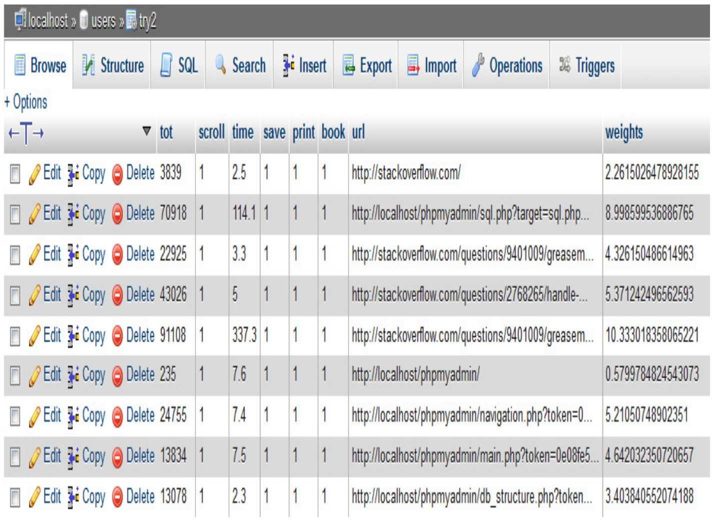
Scripts are written to calculate the active time a user spends on a given page. The mouse movement of a user was calculated by another script. Mouse movement was calculated using JQuery where the number of pixels (height * width) the mouse hovers on was calculated. The moment the document gets loaded a function keeps track for any sort of mouse movement.

Page Scroll was also calculated the same way using JQuery. Scroll refers to horizontal and vertical scrolling of the web page. The script also takes care that if the user scrolls downward and then upward the script nullifies the value of downward scroll with the upward scroll. This takes care of the factor that the content at some portion of the page wasn’t useful for the user because of which he came back to the initial position. The same goes for horizontal, forward and backward scroll.

Another script took care of keyboard inputs for page down, page up, end and home buttons. The scroll action from these buttons is also calculated. Keyboard Shortcuts for bookmarking, saving and printing were captured the same way where a function is triggered as soon as a keyboard stroke is recognized.

Once all these user initiated events are captured, this data along with the URL of the web page is stored into the database. The relevance factor script is then invoked using Gresemonkey which monitors the users' behaviour on the web page. In this way all the necessary data values required for the method are captured and stored.

Since the databases are stored on WAMP server, as shown in figure-1, which is hosted on a client's machine, the database is accessible only to the client who may password protect it, which can prevent any form of invasion of privacy.



	tot	scroll	time	save	print	book	url	weights
Edit Copy Delete	3839	1	2.5	1	1	1	http://stackoverflow.com/	2.2615026478928155
Edit Copy Delete	70918	1	114.1	1	1	1	http://localhost/phpmyadmin/sql.php?target=sql.php...	8.998599536886765
Edit Copy Delete	22925	1	3.3	1	1	1	http://stackoverflow.com/questions/9401009/greasem...	4.326150486614963
Edit Copy Delete	43026	1	5	1	1	1	http://stackoverflow.com/questions/2768265/handle-...	5.371242496562593
Edit Copy Delete	91108	1	337.3	1	1	1	http://stackoverflow.com/questions/9401009/greasem...	10.333018358065221
Edit Copy Delete	235	1	7.6	1	1	1	http://localhost/phpmyadmin/	0.5799784824543073
Edit Copy Delete	24755	1	7.4	1	1	1	http://localhost/phpmyadmin/navigation.php?token=0...	5.21050748902351
Edit Copy Delete	13834	1	7.5	1	1	1	http://localhost/phpmyadmin/main.php?token=0e08fe5...	4.642032350720657
Edit Copy Delete	13078	1	2.3	1	1	1	http://localhost/phpmyadmin/db_structure.php?token...	3.403840552074188

Figure 1 – Screen shot of the table storing the data about user's activity on a web page.

6. DISCUSSION

Whenever a user gets the results for any search query, we can segregate that search query appearing on a page into different segments by using Greasemonkey scripts. These search result segments of a page are thus available to us, which can then be evaluated separately. Depending upon the amount of mouse movement which appears on a segment for the session of the query and detected amount of the mouse click on that segment, we can infer if the given search result was relevant to the user or not.

If the user scrolled till, say result number 5, out of the 10 results appearing on that web page, we can conclude that only the fifth or the top 5 results were relevant to the user. Furthermore, depending on how many segments had mouse hovering or clicks in it, we can conclude which all results among these were relevant to the user and how much. Using our method we can further re-rank search results on a search result page.

7. THE EVALUATION FRAMEWORK

The moment a user visits any web page the script captures it's URL along with the required six factors namely - time, scroll, movement, save, print and bookmark and stores it in the database as soon as the web page is unloaded. The formula proposed above for calculating the relevance factors would be applied to different pages and pages will be sorted according to their relevance factor. This entry is stored for each session. Moreover these entries can be analysed for each day, week, month etc. and can help in depicting the change in a user's browsing pattern and interests over a period of time. Pages which have not been accessed over a long span of time (based on the threshold value) would be automatically truncated in order to increase the efficiency of the method and reduce the sample space evaluation.

8. RESULTS OBTAINED

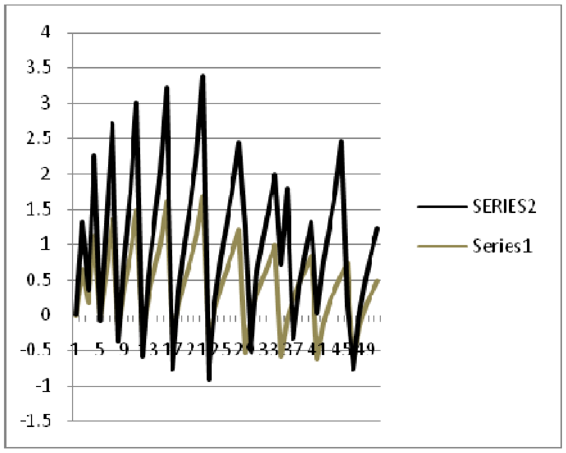


Figure 2 Comparison of performance measures

As depicted in Figure 2, the graph was obtained when we plotted the model initially proposed in [1] as series 1 and our model after the modification as series 2. We observed that the weight (on the y axis) obtained after our modification were lying in a larger range as compared to the model initially proposed.

The weights in model proposed in [1] was in the range of [-0.6, 1.75] whereas the weights obtained from our model after the modification were in the range of [-0.85, 3.4]. Hence, we could conclude that the weights after the modification were more discrete and well-spaced. Since the values of the weights were not very close to each other as earlier, the pages could be ranked in a better way.

9. INFERENCES

The model proposed in [1] did not handle user actions on the web page. However, as a human trait the actions of the user in the browser on the web page reveal the relevance of the web page to him. These user actions are implicit in nature and happen naturally which do not obstruct his flow of actions, if otherwise asked explicitly [11]. The proposed extension handles common user types like – a user who would bookmark a web page for future reference more often than saving or printing, a user who would save the web page for further reference, a user who would print the web page for future reference and a combination thereof.

10. FUTURE WORK

Since the framework can rank pages according to the user's relevance, this method will be used in giving relevant search results to the user. User profile thus generated will be used to give relevant search to a user combined with the ranking returned by a web search engine. For each search term the pages browsed by the user are recorded and ranked according to his profile. Once a concrete database gets created over a period of time, as soon as the user searches any term he will get a list of pages visited by him for similar search done earlier which would be ranked according to their relevance, followed by the search results of the default search engine. In case the search is entirely new to his profile he would just get the search results of the default search engines, the framework will learn this new search query.

The relevance of the corresponding webpage can be further increased if the user copies text from the web page on to the clipboard, it indicates the relevancy of the contents of the webpage and hence the relevancy of the webpage itself as suggested in [12].

REFERENCES

[1] [Teevan et. al., 2005] Jaime Teevan, Susan Dumais, and Eric Horvitz, Personalizing search via automated analysis of interests and activities, In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05). ACM, New York, NY, USA, Pp. 449-456.

[2][Ruthven, 2002] White, R., Ruthven, I. and Jose, J.M., The use of implicit evidence for relevance feedback in web retrieval, Proceedings of the Twenty-Fourth European Colloquium on Information Retrieval Research (ECIR ‘02). Lecture Notes in Computer Science. Glasgow. 2002, Pp.93-109.

[3] [Shapira, 2006] B. Shapira, M. Taieb-Maimon, and A. Moskowitz. Study of the usefulness of known and new implicit indicators and their optimal combination for accurate inference of users interests. Proc. SAC '06, Pp.1118–1119.

[4] [Li, 2008] Fang Li, Yihong Li, Yanchen Wu, Kai Zhou, Feng Li, Xinguang Wang. Discovery of a User Interests on the Internet, In Proceedings of the IEEE/WIC/ACM, International Conference on Web Intelligence and Intelligent Agent Technology [C], 2008, Pp.359-362.

[5] [Kellar, 2004] Kellar, M., Watters, C., Duffy, J., & Shepherd, M. (2004). Effect of task on time spent reading as an implicit measure of interest Proceedings of the 67th Annual Meeting of the American Society for Information Science, 41, Pp.168–175.

[6] [Nagy, 2009] Istvan K. Nagy and Csaba Gaspar-Papanek, User Behaviour Analysis Based on Time Spent on Web Pages, Web Mining Applications in E-commerce and E-Services, Studies in Computational Intelligence, 2009, Volume 172 / 2009, Springer, Pp. 117-136.

[7] [Roman, 2011] Pablo Enrique Roman Asenjo, Web User Behavior Analysis, Doctoral Thesis, March 2011.

[8] [Nyman, 2013] Mathias Nyman, Navigation Behavior Analysis and User Profiling Based on Automatically Collected Website Data, Master’s Thesis, School of Science, Aalto University, Finland, February 1, 2013.

[9] [Claypool, 2001] Mark Claypool, Phong Le, Makoto Wased, David Brown, "Implicit interest indicators", Proceedings of the 6th international conference on Intelligent user interfaces, January 14-17, 2001, Santa Fe, New Mexico, USA. Pp. 33-40.

[10] [Kim, 2005] Kim, H. and Chan, P. K. Implicit indicator for interesting web pages, International Conference on Web Information Systems and Technologies, Miami, 2005, Pp.270-277.

[11] [Reber, 1989] Implicit Learning and Tacit Knowledge Journal of Experimental Psychology: Arthur S. Reber (1989) General 1989, Vol. 118, No. 3, Pp. 219-235.

[12] [Holub, 2010] Michal Holub, Maria Bielikova, Estimation of user interest in visited web page, Proceedings of the 19th international conference on World Wide Web, WWW '10, April 26-30, 2010, Raleigh, North Carolina, USA, Pp. 1111-1112.