Optimizing Web Structures Using Web Mining Techniques

Jonathan Jeffrey[1], Peter Karski[1], Björn Lohrmann[1], Keivan Kianmehr[1], and Reda Alhajj[1,2]

[1] Computer Science Dept, University of Calgary, Calgary, Alberta, Canada
[2] Department of Computer Science, Global University, Beirut, Lebanon,
alhajj@ucalgary.ca

Abstract. With vibrant and rapidly growing web, website complexity is constantly increasing, making it more difficult for users to quickly locate the information they are looking for. This, on the other hand, becomes more and more important due to the widespread reliance on the many services available on the Internet nowadays. Web mining techniques have been successfully used for quite some time, for example in search engines like Google, to facilitate retrieval of relevant information. This paper takes a different approach, as we believe that not only search engines can facilitate the task of finding the information one is looking for, but also an optimization of a website's internal structure, which is based on previously recorded user behavior. In this paper, we will present a novel approach to identifying problematic structures in websites. This method compares user behavior, derived via web log mining techniques, to an analysis of the website's link structure obtained by applying the Weighted PageRank algorithm (see [19]). We will then show how to use these intermediate results in order to point out problematic website structures to the website owner.

Keywords: data mining, web mining, user behavior, search engines, PageRank.

1 Introduction

Website complexity is constantly increasing, making it more difficult for users to quickly locate the information they are looking for. This, on the other hand, becomes more and more important due to the widespread reliance on the many services available on the Internet nowadays. Analyzing the user's behavior inside the website structure, will provide insight on how to optimize the website's structure to improve usability.

Web Mining is the use of data mining methods to identify patterns and relationships amongst web resources. It is basically classified into web mining: web content, web usage and web structure mining; the last two are used to solve the website structure optimization problem. Web structure mining involves the crawling and analysis of web page content to identify all links existing within

the page, which will then be used to create a directed graph representing the structure of the site being mined. Each node within this graph signifies an individual page and each edge is a link between two pages. On the other hand, web usage mining requires the parsing of web server logs to identify individual user behavior. Specifically, the sites visited, total visits and total time spent looking at the page, also known as "think time", are considered. These values are parsed from original server logs, or could be taken from preprocessed logs as well. Furthermore, in this paper, we explain how to use the Weighted PageRank algorithm [2, 3, 19, 20] for web-structure mining to analyze the hyperlink structure of a website. Also, we demonstrate how to use web log mining to obtain data on the site user's specific navigational behavior. We then describe a scheme how to interpret and compare these intermediate results to measure the website's efficiency in terms of usability. Based on this, it shall be outlined how to make recommendations to website owners in order to assist them in improving their site's usability.

The rest of this paper is organized as follows. Section 2 is related work. Section 3 describes the proposed approach; we first present how web structure mining is utilized in the process of website optimization; then describe the participation of web usage mining to the process; then we discuss how the overall recommendation is conveyed to the user of the analyzed website. Section 4 reports test results that demonstrate the applicability and effectiveness of the proposed approach. Section 5 is summary and conclusions.

2 Related Work

As described in the literature, numerous approaches have been taken to analyze a website's structure and correlate these results with usability, e.g., [6, 7, 9–11, 15, 16]. For instance, the work described in [14] devised a spatial frequent itemset data mining algorithm to efficiently extract navigational structure from the hyperlink structure of a website. Navigational structure is defined as a set of links commonly shared by most of the pages in a website. The approach is based on a general purpose frequent itemset data mining algorithm, namely ECLAT [5]. ECLAT is used to mine only the hyperlinks inside a window with adaptive size, that slides along the diagonal of the website's adjacency matrix. They compared the results of their algorithm with results from a user-based usability evaluation. The evaluation method gave certain tasks to a user (like for example finding a specific piece of information on a website) and recorded the time needed to accomplish a task and failure ratios. The researchers found a correlation between the size of the navigational structure set and the overall usability of a website, specifically the more navigational structure a website has, the more usable it is as a general rule of thumb.

In [18], it is proposed to analyze the web log using data mining techniques to extract rules and predict which pages users will be going to be based on their prior behavior. It is then shown how to use this information to improve the website structure. By its use of data mining techniques, this approach is related

to our approach described in this paper, although the details of the method vary greatly, due to their use of frequent itemset data mining algorithms. The main difference between our approach and the method described in [18] is that they do not consider the time spent on a page by a visitor in order to measure the importance of that particular page. Their approach applies frequent itemset mining that discovers navigation preferences of the visitors based on the most frequent visited pages and the frequent navigational visiting patterns. However, we believe that in a particular frequent navigational pattern there might exist some pages which form an intermediate step on the way to the desirable page that a user is actually interested in. Therefore, the time spent on a page by a visitor has to be considered as an important measure to quantify the significancy of a page in a website structure.

The work described in [13] proposed two hyperlink analysis-based algorithms to find relevant pages for a given Web page. The work is different in nature from our work; however it applies web mining techniques. The first algorithm extends the citation analysis to web page hyperlink analysis. The citation analysis was first developed to classify core sets of articles, authors, or journals to different fields of study. In the context of the Web mining, the hyperlinks are considered as citations among the pages. The second algorithm makes use of linear algebra theories to extract more precise relationships among the Web pages to discover relevant pages. By using linear algebra, they integrate the topologic relationships among the pages into the process to identify deeper relations among pages for finding the relevant pages. The work described in [12] describes an expanded neighborhood of pages with the target to include more potentially relevant pages.

In the approach described in [19], the standard PageRank algorithm was modified by distributing rank amongst related pages with respect to their weighted importance, rather than treating all pages equally. This results in a more accurate representation of the importance of all pages within a website. We used the Weighted PageRank formula outlined in [19] to complement the web structure mining portion of our approach, with the hope of returning more accurate results than the standard PageRank algorithm.

In [21], the authors outline a method of preparing web logs for mining specific data on a per session basis. This way, an individual's browsing behavior can be recorded using the time and page data gathered. Preparations to the log file such as stripping entries left by robots are also discussed.

3 Overview

In order to achieve our goal of recommending changes to the link structure of a website, we have identified two main subproblems which must be initially solved. First, to determine which pages are important, as implied by the structure of the website. Second, to conclude which pages the users of this website consider to be important, based on the information amassed from the web log. Once we have solved these two subproblems, we now have methods in place which give us two different rankings of the same web pages. Our final task is implementing

a scheme to compare the results of the first two problems and make meaningful recommendations. In the subsections which follow, we will discuss the algorithms we will use for unraveling each of these tasks and the reasoning behind these algorithms.

3.1 Web Structure Mining

Web structure mining involves crawling through a series of related web pages (for example all pages inside a user defined subdomain), extracting meaningful data that identifies the page and use that data to give the page a rank based on given criteria. To begin with, a set or root of web pages is provided, an application called a crawler will traverse these pages and extract the needed information from them. The information we are interested in are the hyperlinks contained within the page.

Extracting this information can be done using regular expressions. There are challenges using regular expressions, because they assume that the code used within the web page follows all standards. Simple errors, such as some HTML tags not being closed or improperly formatted and non-HTML code such as CSS or javascript can throw off the parsing of the page and lead to inaccurate results.

Once the hyperlinks within the webpage have been extracted, the crawler will recursively continue crawling the web pages whose links were found in the current page after replicates have been removed, since crawling the same page twice is unnecessary. Any duplicate hyperlinks within the page will need to be removed, as well as any links that have already been processed or are already in the queue awaiting processing. After having crawled the complete website or a user defined part of it, depending on what the user specifies, the standard page rank $PR(p_i)$ of each page p_i can be computed as

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \tag{1}$$

where N is the total number of pages that have been crawled, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outgoing links on page p_j, and d is a damping factor, usually chosen around 0.85. The damping factor can be interpreted as the probability that a user follows the links on a page. It has been included due to the following observation: *Sometimes a user does not follow the links on a page p_k and just chooses to see a random page p_l by entering its address directly.* This should be considered when computing the page rank of p_l. Thus, in the above formula, $\frac{1-d}{N}$ can be seen as the influence of a random jump to page p_i on the page rank $PR(p_i)$.

In [19], an improved version of standard page rank is proposed. The weighted page rank algorithm (WPR) considers the fact that the page rank of a popular page should have a higher weight than the one of an unpopular page. The WPR value is computed as:

$$WPR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} PR(p_j) W^{in}_{(p_j,p_i)} W^{out}_{(p_j,p_i)} \tag{2}$$

Here, $W^{in}_{(p_j,p_i)}$ and $W^{out}_{(p_j,p_i)}$ are the weights of the link between documents p_j and p_i. They can be computed as:

$$W^{in}_{(p_j,p_i)} = \frac{I_{p_i}}{\sum_{p \in R(p_j)} I_p} \tag{3}$$

$$W^{out}_{(p_j,p_i)} = \frac{O_{p_i}}{\sum_{p \in R(p_j)} O_p} \tag{4}$$

where I_x is the number if links pointing to page x, O_x is the number of outgoing links in page x and $R(x)$ is the set of pages that are linked to from page x. Each element of the sum of page ranks is multiplied by it's respective weight. The result is that more important pages are given a higher page rank, unlike the original page rank algorithm that divides the rank of a page evenly amongst documents it links to. At the end, a more accurate result is achieved.

We chose to use WPR instead of the standard page rank, since it has proven to yield slightly better results in experiments (see [19]). The output of this processing stage is a list of $[p_i, WPR(p_i)]$ pairs, sorted in descending order by the WPR values.

3.2 Web Log Mining

We have decided to base the rankings which users give to web pages on two parameters: 1) frequency (number of visits), and 2) time (total time spent by all users at a web page).

Preprocessing: We need to take steps to "clean" the web log to minimize interference from robots before using it to generate the actual output of this stage. One approach we consider useful for this has been proposed in [21], where it is proposed to discard those sessions that match the following access patternts that are likely to be robots characteristics:

- Visiting around midnight, during light traffic load periods in order to avoid time latency.
- Using `HEAD` instead of `GET` as the access method to verify the validity of a hyperlink; the Head method performs faster in this case as it does not retrieve the web document itself.
- Doing breadth search rather than depth search; robots do not navigate down to low-level pages because they do not need to access detailed and specific topics
- Ignoring graphical content; robots are not interested in images and graphic files because their goal is to retrieve information and possibly to create and update their databases.

Based on the cleaned logfile, we then identify sessions, which in turn are used to compute the total number of visits v_i and the total time spent by users t_i for each page. It shall be noted that we must ensure that the number of user sessions we extract from the web log are of sufficient size to give us a realistic ranking of popular pages.

Computing log rank values: As already mentioned, the first parameter to consider in the process is the number of times a particular page was visited by our group of users. The fact that a user has visited a page may lead us to believe that they consider it an important page. While this is true in many cases, there is also the possibility that the page was just an intermediate step or "hop" on the way to the page which the user is actually interested in. A very high number of visits from page $A \rightarrow B$ and a relatively low total time spent at page B seem to imply that the page B is used primarily (or even exclusively) as a "hopping" point. In this case, a viable recommendation may be to change the link structure of the website so that the user is able to navigate directly from $A \rightarrow C$, without having to make a stop in-between at page B. Therefore, we need a way to give lesser weight to the visit counter and a higher weight to the total time in our ranking scheme.

Assuming that v_i is the number of visitors for a page i and t_i is the total time spent by all visitors on this page, the *log rank* value l_i shall be defined as:

$$l_i = 0.4v_i + 0.6t_i \tag{5}$$

Taking a weighted sum of the visits and time will result in a value that represents the importance of a page relative to the others. Pages that are frequently visited and accessed for long periods of time will have a larger log rank than pages with an insignificant number of visits and think time. Rather than giving time and visits equal importance as discussed above, the difference is quantified through a constant, in this case being a 60/40 split, respectively. Depending on the content of the web site being analyzed, these constants can be changed to account for the specific audience or purpose of the site. For example, a website with pages normally filled with large amounts of visual/textual information will have on average longer think times than pages sparsely filled with content. A balance between having fewer pages with large amounts of content, versus many pages with little content must be identified and quantified in the log rank function in order to be accurate. Finally, the output of this stage, is a list of pages sorted by their log rank value l_i.

3.3 Analysis

This is the last stage of processing and yields results directly for the users, in form of recommendations on how to change their website. The required input is:

- A list of pages with their page rank values $p_i \in R^+$, which is sorted by the page rank values.
- A list of websites with their ranking values $l_i \in R^+$ from the weblog mining, which is sorted by the ranking values.

Preprocessing: First, we need to preprocess the page rank and log ranking values. This step consists of normalizing them to a common index of integers. This is done by taking the list of page ranks and sorting them in descending

order. We then chose to assign an index to each page rank value, the largest page rank receiving index zero, and the smallest page rank receiving the highest index value. In the event that several pages share the same page rank value, they would receive the same index value. The same process is then repeated for the log rank values.

This step changes the log and page rank values from two different distributions into a simple linear distribution, which facilitates appropriate comparison. We validated this step during the evaluation of our method (see Section 4), where the log and page rank values had significantly different distributions, which yielded non-sensical analysis results.

The output of this step consists of two lists, the page rank and log rank indexes along with their respective page. At this step, the lists are still independent of each other.

Performing Analysis: After the preprocessing step, we can now compute the following value d_i for each page:

$$d_i = index(l_i) - index(p_i) \tag{6}$$

This calculates the difference in rank between the two rank values. Ideally, we will find $d_i = 0$, because there should be little deviation in the ranking of the page rank and log rank values. Finally, the pages will be sorted in ascending order according to their d_i values. At the top and the bottom of the list, we can distinguish the following cases:

1. The page has got a high page rank index and a low log rank index (d_i very low).
2. The page has got a low page rank index and a high log rank index (d_i very high).

In case 1, the software performing the analysis should recommend the user to put the site into a place, where it is harder to reach, in favor of pages that might require to be reachable more easily. This includes but is not limited to:

- Removing links to that page, especially on those pages with high page rank.
- Linking to the page from places with low page rank value instead.

In case 2, the software should recommend modifying the link structure in a fashion that makes the page easier to reach. This means, for example, adding links to that page, especially on suitable pages with high page rank.

The intuition for a very high or very low deviation generally being undesirable, is the following: One could interpret a high page rank value as a site being easily reachable from other (important) pages, whereas a low page rank value thus could be interpreted as an indicator, that the page is hard to reach. On the other hand, a high log rank value testifies that a page is popular, whereas low log rank values indicate unpopular pages. Therefore, in the first case with a very low d_i, the site is easy to reach, but only few people actually want to see

it; and in the second case with a very high d_i, the page is very hard to reach for visitors, but comparably many people want the information on it and have to spend time looking for it. Thus, it is natural, that according to this scheme, an ideally positioned page has a value $d_i \approx 0$.

The Relinking Process: The aforementioned "relinking" process has to be carried out manually by the website owner since he/she has to consider the content structure of the page; thus it is out of the scope of this algorithm to propose concrete relinking in terms of "Link page A to B" or "Remove the link on page A". The outlined algorithm merely represents a support in determining possibly misplaced pages and in deciding where to add or remove links (page rank values can be helpful here).

To assist him/her in the process, after this stage, the website owner should be presented with the following information:

- The sorted list of pages (called *UNLINK* list), with $d_i < 0$ and $d_i^* > \epsilon_1$, where ϵ_1 is a user defined threshold.
- The sorted list of pages (called *LINK-TO* list), with $d_i > 0$ and $d_i^* > \epsilon_2$, where ϵ_2 is another user defined threshold.
- For each webpage in the above lists, provide the set of pages that link to it (incoming links) and the set of pages that are being linked to from it (outgoing links).
- The page rank and log rank value for each webpage that has been analyzed, including but not limited to those in the *UNLINK* and the *LINK-TO* list.

This information should be sufficient to detect and resolve design issues in a website's structure that affect usability. The ranking approach is supportive in that it helps the owner to focus on the important issues. To guide the process of relinking or altering the structure, page rank and log rank values are provided.

4 Evaluation

We tested our algorithm on a medium sized website ($\approx$ 631 pages) obtained from [17], which provides reference for HiFi devices. Its structure is mostly wider than deep, as for example when it lists the manufacturers of documented devices. Since this website has been provided for experiments with data mining techniques, it already came with a log file that had been parsed into sessions. Performing the analysis on the site yielded the distribution of deviation values d_i as shown in Figure 1.

As can be seen from Figure 1, we have a relatively low number of pages with a deviation far from the ideal value. The majority of the pages fall within a small margin of ± 200, which is still acceptable. Some pages like for example /dr-660/index.html (lowest d_i value) showed a large discrepancy between user popularity and reachability, since it was linked to from one of the central pages, but hardly received any hits. Other pages like /manufacturers/korg/s-3/index.html (second highest d_i value) appear to have been very popular with

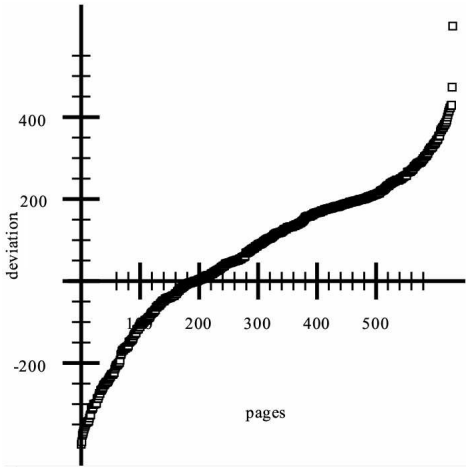

Fig. 1. Plot of d_i (sorted by d_i)

the site users, but are relatively hard to reach since they are hidden "deep" in the website's structure. A viable change in this case would be to provide a link to it on the pages at or close to the website's document root (for example in a "Favorites" or "Recommendations" section), since this is where the users start browsing. Further investigation of the highest and lowest values, showed the same tendency and thus revealed locations where relinking seemed necessary after manual investigation from our side.

Despite a certain "noise" (meaning pages that are classified as misplaced, but cannot be really relinked), our method has succeeded to identify problematic locations in the website's structure.

5 Summary and Conclusions

In this paper, we explained how to use the Weighted PageRank algorithm for web-structure mining to analyze the hyperlink structure of a website. Further, we demonstrated how to use web log mining to obtain data on the site user's specific navigational behavior. Our approach then showed how to combine these values in order to measure a website's usability. We successfully validated our method using the data set provided under [17], which shows that this is a simple but viable approach to solve the given problem. In our opinion, a similar method should be used as part of a larger set of tools, when it comes to usability optimization of websites.

References

1. S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. *Proc. of the International Conference on World Wide Web*, pp.280-290,

2003.
2. A. Altman and M. Tennenholtz. Ranking systems: the pagerank axioms. *Proc. of ACM Conference International on Electronic commerce*, pp.1-8, 2005.
3. M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Transactions on Internet Technology*, 5(1):92–128, 2005.
4. P. Boldi, M. Santini, and S. Vigna. Pagerank as a function of the damping factor. *Proc. of the International Conference on World Wide Web*, pp.557-566, 2005.
5. C. Borgelt. Efficient implementations of apriori and eclat, Nov. 14 2003.
6. A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Transactions on Internet Technology*, 5(1):231–297, 2005.
7. J. T. Bradley, D. V. de Jager, W. J. Knottenbelt, and A. Trifunovic. Hypergraph partitioning for faster parallel pagerank computation. *Proc. of Formal Techniques for Computer Systems and Business Processes, European Performance Engineering Workshop*, pp.155-171, 2005.
8. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. *Proc. of the International Conference on World Wide Web*, 1998.
9. Y.-Y. Chen, Q. Gan, and T. Suel. I/o-efficient techniques for computing pagerank. *Proc. of ACM International Conference on Information and knowledge management*, pp.549-557, 2002.
10. P.-A. Chirita, J. Diederich, and W. Nejdl. Mailrank: using ranking for spam detection. *Proc. of ACM International Conference on Information and knowledge management*, pp.373-380, 2005.
11. J. Cho, S. Roy, and R. E. Adams. Page quality: in search of an unbiased web ranking. *Proc. of ACM SIGMOD*, pp.551-562, 2005.
12. J. Dean and M. Henzinger. Finding related pages in the world wide web. *Proc. of the International Conference on World Wide Web*, 1999.
13. J. Hou and Y. Zhang. Effectively finding relevant web pages from linkage information. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):940–951, 2003.
14. C.H. Li and C.K. Chui. Web structure mining for usability analysis. Proc. of *IEEE/WIC/ACM International Conference on Web Intelligence*, pp.309-312, 2005.
15. X.-M. Jiang, G.-R. Xue, W.-G. Song, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Exploiting pagerank at different block level. *Proc. of the International Conference on Web Information Systems Engineering*, pp.241-252, 2004.
16. P. Massa and C. Hayes. Page-rerank: Using trusted links to re-rank authority. *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, pp.614–617, 2005.
17. U. of Washington Artificial Intelligence Research. Music machines website. `http://www.cs.washington.edu/ai/adaptive-data/`.
18. I. V. Renáta Iváncsy. Frequent pattern mining in web log data. *Journal of Applied Sciences at Budapest Tech*, 3(1):77–90, 2006.
19. W. Xing and A. A. Ghorbani. Weighted pagerank algorithm. *Proc. of Annual Conference on Communication Networks and Services Research*, pp.305-314. IEEE Computer Society, 2004.
20. W. Xing and A. A. Ghorbani. Weighted pagerank algorithm. *Proc. of Annual Conference on Communication Networks and Services Research*, pp.305-314, 2004.
21. J. X. Yu, Y. Ou, C. Zhang, and S. Zhang. Identifying interesting customers through web log classification. *IEEE Intelligent Systems*, 20(3):55–59, 2005.