Architecture for a Parallel Focused Crawler for Clickstream Analysis

Ali Selamat and Fatemeh Ahmadi-Abkenari

Software Engineering Research Group, UTM Knowledge Economy Research Alliance & Software Engineering Department, Faculty of Computer Science & Information Systems, Universiti Teknologi Malaysia (UTM), 81310 UTM Johor Baharu Campus, Johor, Malaysia
aselamat@utm.my, pkhoshnoud@yahoo.com

Abstract. The tremendous growth of the Web poses many challenges for all-purpose single-process crawlers including the presence of some irrelevant answers among search results and the coverage and scaling issues regarding the enormous dimension of the World Wide Web. Meanwhile, more enhanced and convincing algorithms are on demand to yield more precise and relevant search results in an appropriate amount of time. Due to the fact that employing the link based Web page importance metrics in search engines is not an absolute solution to identify the best answer set by the overall search system and because employing such metrics within a multi-processes crawler bears a considerable communication overhead on the overall system, employing a link independent Web page importance metric is required to govern the priority rule within the queue of fetched URLs. The aim of this paper is to propose a modest weighted architecture for a focused structured parallel crawler in which the credit assignment to the discovered URLs is performed upon a combined metric based on clickstream analysis and Web page text similarity analysis to the specified mapped topic(s).

Keywords: Clickstream analysis, Focused crawlers, Parallel crawlers, Web data management, Web page importance metrics.

1 Introduction

The dimension of the World Wide Web is being expanded by an unpredictable speed. As a result, search engines encounter many challenges such as yielding accurate and up-to-date results to the users, and responding in an appropriate timely manner. A centralized single-process crawler is a part of a search engine that traverses the Web graph and fetches any URLs from the initial or seed URLs, keeps them in a queue and then in an iterated manner - according to an importance metric - selects the first most important K URLs for further processing. A parallel crawler on the other hand is a multi-processes crawler in which upon partitioning the Web into different segments, each parallel process is responsible for one of the Web fractions. Since due to the enormous size of the Web, a single-process crawler is not capable of reaching an acceptable download rate, employing a parallel crawler within a search engine architecture is scalable. Besides, different parallel processes could run at geographic

N.T. Nguyen, C.-G. Kim, and A. Janiak (Eds.): ACIIDS 2011, LNAI 6591, pp. 27–35, 2011.

distant location to download the pages on different zones, so, upon applying the parallel crawler, the load on the overall network could be reduced [9]. The bottleneck in the performance of any crawler is applying an appropriate Web page importance metric.

In this paper we employ a clickstream-based metric as a heuristic; the hypothesis is the existence of a standard upon which the authorized crawlers have legal rights to access the server log files. We first review on the literature of focused crawlers and the existing Web page importance metrics by defining the drawbacks of each of them. Then, we briefly discuss our clickstream-based metric since it has been thoroughly discussed in a companion paper. Next, the application of the clickstream-based metric within the architecture of a focused parallel crawler will be presented. So, the objective of this paper is to propose an architecture for a focused-based parallel crawler in which prioritizing the crawl frontier is based on a function of clickstream analysis combined with a text analysis approach. The text analysis part of the derived formula helps the identification of authoritative Web content in newly uploaded pages and the pages in the dark part of the Web. Besides, the suggested architecture answers this question that in the absence of a central coordinator, how and in which order, the overall crawler reads the ordered queue of the parallel processes to achieve the most important discovered pages by parallel processes at the earliest time of result organizing.

2 Related Works

The related works on focused crawlers and linked based importance metrics are discussed as follows;

2.1 Focused Crawlers

There are two different classes of crawlers known as focused and unfocused. The purpose of unfocused crawlers is to attempt to search over the entire Web to construct their index. As a result, they confront the laborious job of creating, refreshing and maintaining a database of great dimensions. While a focused crawler limits its function upon a semantic Web zone by selectively seeking out the relevant pages to predefined topic taxonomy and avoiding irrelevant Web regions as an effort to eliminate the irrelevant items among the search results and maintaining a reasonable dimensions of the index. A focused crawler's notion of limiting the crawl boundary is fascinating because "a recognition that covering a single galaxy can be more practical and useful than trying to cover the entire universe" [6].

The user information demands specification in a focused crawler via exemplary Web documents instead of by keyword-based document. Therefore, a mapping process is performed by the system to highlight (a) topic(s) in the pre-existing topic tree which can be constructed based on human judgment [6]. The core elements of a traditional focused crawler are a classifier and a distiller. While the classifier checks the relevancy of each Web document's content to the topic taxonomy based on the naïve Bayesian algorithm, the distiller finds hub pages inside the relevant Web regions by utilizing a modified version of HITS algorithm. These two components, together determine the priority rule for the existing URLs in a priority based queue called crawl frontier [6], [13], [15], [16].

2.2 Link-Dependent Web Page Importance Metrics

As stated above, a centralized crawler or each parallel process within a parallel crawler retrieves URLs and keeps the links in a queue of URLs. Then, in the next step, due to the time and storage constraints, a crawler or a parallel process must decide which most important K URLs to process first according to one Web page importance metric. There are diverse Web page importance metrics, each views the importance of a page from a different perspective such as outgoing or incoming link enumeration, text analysis or location angles including Backlink count, PageRank, HITS, forward link count, location metric and content-query similarity checking metrics [10]. But the most well-known category is the link-based metrics. PageRank metric as a modification to Backlink count that simply counts the links to a page, calculates the weighted incoming links in an iterated manner and considers a damping factor which presents the probability of visiting the next page randomly [3],[10]. The *TimedPageRank* algorithm adds the temporal dimension to the PageRank by considering a function of time $f(t)$ ($0 \leq f(t) \leq 1$) in lieu of the damping factor. The notion of TimedPageRank is that a Web surfer at a page i has two options: 1) Randomly choosing an outgoing link with the probability of $f(t_i)$ and 2) Jumping to a random page without a link with the probability of $1\text{-}f(t_i)$. For a completely new page within a Web site, an average of the TimedPageRank of other pages in the Web site is used [19]. The HITS metric views Web page importance in its hub and authority scores. A Web page with high hub score is a page that points to Web pages with high authority scores and a Web page with high authority score is a page that has been pointed to by Web pages with high hub scores.

The HITS metric has some drawbacks including the issue of topic drift, its failure in detecting mutually reinforcing relationship between hosts, and its shortcoming to differentiate between the automatically generated links from the citation-based links within the Web environment. Due to the fact that pages to which a hub page points to are not definitely around the original topic, the problem of topic drift is formed. The second problem occurs when a set of documents in one host points to one document on another host. As a result, the hub score of pages on the first host and the authority score of the page on second host will be increased. But this kind of citation cannot be regarded as coming from different sources. Finally, Web authoring tools generate some links automatically that these links cannot be regarded as citation based links. Although the literature includes some modifications to HITS algorithm such as the research on detecting micro hubs, neglecting links with the same root, putting weights to links based on some text analysis approach or using a combination of anchor text with this metric, there is no evidence of a complete success of these attempts [2], [4], [5], [7],[8].

3 Proposed Clickstream Based Importance Metric

Since the area of Web usage mining considers the utilization of server log files, the usage of clickstream analysis is roughly ignored and little attention to the research has been allocated to it as a Web page importance metric despite its advantages. The literature includes the research on clickstream data for e-commerce objectives [12], [14]. In a companion paper, we proposed the clickstream analysis for Web page importance determination [1]. Besides, that paper has included the application of this proposed metric in a focused crawler. Furthermore, the application of this metric in a crawler with only a

parallel structure has been discussed in the companion paper [17]. Clickstream-based importance metric is computed according to the total duration of all visits per page during the observation span of time. In other words, the log ranking for a page (LR_p) is the total duration of server sessions per page (D_{sp}) as shown in Equation (1) [1];

$$LR_p = D_{sp} \tag{1}$$

According to the link independent nature of clickstream-based metric, in a crawler based of this metric, the need for link enumeration will be removed. As a result, in a parallel crawler, the communication overhead, together with parallel processes that inform each other of the existence of links will be eliminated [1]. Also upon employing the clickstream importance metric, the calculated importance of each page is precise and independent from the downloaded segments of the Web. In our approach, by employing a clickstream-based metric within the crawler, we will go beyond noticing the page importance in its connection pattern. Instead, the page credit computation is performed according to a function of a simple textual log files in lieu of working with matrices of high dimensions. Because the clickstream analysis has no relation with the page content, we combine it with a text analysis approach to make a robust decision on priority determination for the items stored in a crawl frontier.

4 Architecture of a Crawler Based on Clickstream Importance Metric

Within a crawler with a focused structure let's consider G as a Web Graph with physically dispersed nodes, C as a tree-shaped topic directory, c as each topic node, $D(c)$ as example documents associated with topic c, c^* as the highlighted mapped node, p as a Web page and $R_{c^*}(p)$ is a measure of content relevancy page p to c^*. Since it is not very easy for users to issue an effective search request, the user enters the interface and imports the documents (Web pages) of his/her interest to the search system. We call this set as D_u. Upon analyzing the imported documents, the system highlights (a) node(s) in the existing topic taxonomy tree through the mapping process. Figure 1 depicts the mapping process from the corpus to (a) node(s) in topic taxonomy tree.

Let's call the mapped highlighted node, c^*. The mapping process will be done using an *Inverted List* with special heed to the terms in title, italic and bold faced and headings in the imported documents by considering the weighting scheme of *TF-IDF*. So given a set of user imported documents of $D=\{d_1, \ldots, d_n\}$ with a unique identifier for each document, there is a vocabulary V containing all the distinct terms in the semantic region in which the focused crawler is specialized, such as the field of computer science. The Inverted list version is $<id_i, w_j, [o_1,\ldots, o_k]>$ in which the id_i is the document unique identifier, the w_j is the weight of each term j and the rest is the offset of the term j in the document i [14]. In order to use less memory space for the Inverted List, a method of compression is used to represent the document unique identifier since it is the most space consuming section of an Inverted List. The method of compression could be *Elias Delta coding* in which the representation is shorter for large integers such as documents numbers in comparison to other compression methods like Unary coding or Elias Gamma coding [14]. Furthermore the system will add any distinct terms from the corpus which is not included in the vocabulary into a temporary vocabulary with the exact version of the above Inverted List. Then a

Semantic Checker will be used to determine the semantically related terms from this temporary vocabulary with the terms in the main vocabulary of *V*. Then the not semantically related words with high importance rate will be used by the system to edit the topic taxonomy. After the detection of (a) node(s) in the topic taxonomy tree, the pre-existing examples (Web pages) associated with the node(s) will be added to the user imported examples to form the crawling list or *CL* as shown in Equation (2) [1];

$$CL = D_u + D(c^*) \tag{2}$$

The *CL* list in our approach is considered as the second level seed URL and should be divided among parallel processes by a Web partitioning function. Due to the fact that all of the pages of a Web site are not the descriptive in nature and usually one or a few of them could illustrate the user's topics of interest, the probability that the Web pages inside the D_u and $D(c^*)$ belonging to different Web sites is high. Therefore with regard to the discussed advantages of the site-hash-based approach, we choose the site-hash-based partitioning function to divide the second level seed URLs among parallel processes. As a result, the selected partitioning function could distribute the crawling list among parallel processes in a balanced manner.

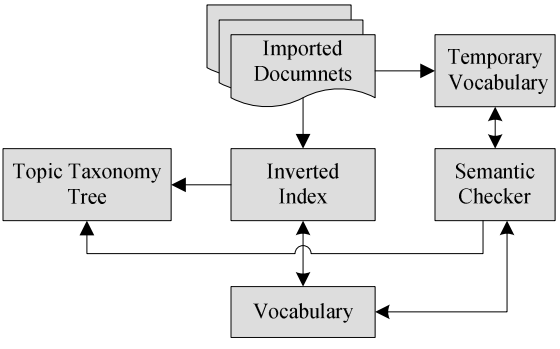

Fig. 1. Mapping process of user imported documents to (a) node(s) in topic taxonomy tree

As depicted in Fig. 2, in the proposed architecture for a focused parallel crawler, each parallel process has four elements namely *distiller*, *classifier*, *crawler* and *coordinator*. The crawler element is responsible for the process of fetching any unvisited URLs from the allocated second seed URLs in an iterated manner and populates them in the crawl frontier. The crawl frontier is a priority-based list corresponding to the Best-First crawling. Also a time-stamp list of visited URLs or *crawl history* is maintained by the overall crawler to keep those URLs pages that have been fetched, as a way to decrease the overlaps among different parallel processes. The history can be maintained in disk for post crawling evaluation or in memory for fast look-up [14]. Besides, a section of *duplication detector* is maintained by each parallel process to prevent the occurrence of duplicate URLs in the crawl frontier through maintaining a separate hash-table. Figure 2 only shows one of the parallel processes because of the space limitation. In the next step, there should be an algorithm which guides the priority rule inside the crawl frontier to choose the most important URLs for further processing.

For having a robust decision on the importance of each existing page in the crawl frontier, the distiller and classifier sections together govern the priority rule. The distiller element is responsible to calculate the page importance based on the clickstream analysis. We call the result as *LR(p)* short for ranking of page *p* based on log file contents. Intermittently the classifier measures the importance of page based on the content relevancy of page *p* to the mapped topic(s) of *c**. The result of this part will be reflected in the computed value of $R_{c*}(p)$ as $0 \le R_{c*}(p) \le 1$. The $R_{c*}(p)$ exactly like that in the traditional focused crawler is based on the naïve Bayesian algorithm as shown in Equation (3) [15];

$$R_{c*}(p) = \sum_{c*} \Pr(c* \mid p) \tag{3}$$

Now a combined importance metric is applied for credit assignments to the items in the frontier by the coordinator section of each parallel process. As discussed earlier, the presence of a central coordinator causes an inevitable load on the network due to the unavoidable communication between this component and parallel processes. Besides, the existence of a central section within a system causes maintenance complication with regard to the dependency of the whole system to this element on the upgrade occasions. Hence in our proposed architecture, we eliminate the presence of a central coordinator and instead, there is a smaller section inside each parallel process to harmonize the result of each parallel process with the results of other parallel processes.

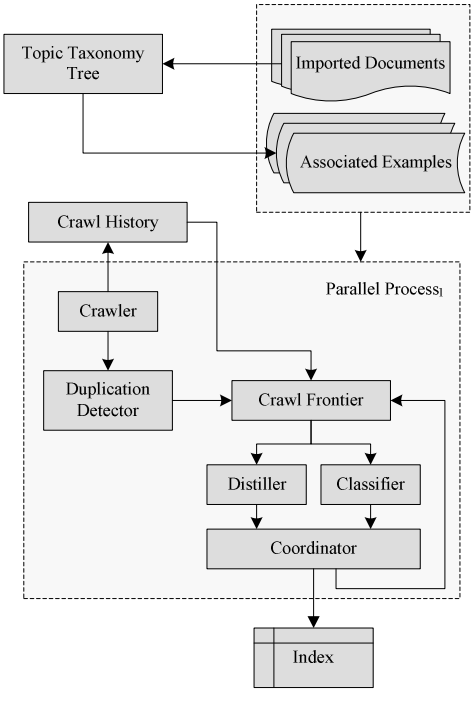

Fig. 2. Architecture of a focused parallel crawler

Therefore the coordinator of each parallel process is responsible for combining the two measures of *LR(p)* and $R_{c*}(p)$ to yield an importance measure of *I(p)*. As *LR(p)* is from the second type and it has a colossal value, to leverage it with the $R_{c*}(p)$, it is converted into hours. Moreover due to the fact that some pages with high *LR(p)* scores may contain news and be of low descriptive nature for user information demands, therefore in the computation of *I(p)*, an emphasis factor of *E* empowers the $R_{c*}(p)$ to make it more compelling than *LR(p)*. So the *I(p)* is computed as shown in Equation (4);

$$I(p) = LR(p)/3600 + E \times R_{c*}(p) \tag{4}$$

Upon the computation of *I(p)* by the coordinator section for each Web page in the crawl frontier, the queue of each parallel process is reordered based on the descending value of *I(p)*. Now each parallel process knows the importance of each page in its crawl frontier. The question raised in the absence of a central coordinator is that in which order the overall crawler fetches the ordered queue of each parallel process. To address this issue, a matter of communication transfer is vital among parallel processes to substitute the role of the central coordinator and to yield an organized and integrated result by the overall crawler. To achieve this objective, let's consider *K* as the size of the crawl frontiers or the number of Web pages in the queue of each parallel process in a way that it is the same for all parallel processes. As part of communication, parallel processes should inform each other of the importance of the pages in their queue. If during the communication, the average of *I(p)* for all the *K* URLs is transferred, it means that a queue with few number of very high important pages and many low important pages may have the same average as a queue with many medium important pages. Missing the very high important pages contributes to inaccuracy and misjudgment of the overall crawler. To prevent this problem, the communication among parallel processes in our approach, consists of sending the average of *I(p)* for a fraction of the pages or the *K/L* size of the frontier in lieu of transmitting the average of *I(p)* for *K* URLs. Therefore if *m* is the number of parallel processes in the parallel crawler and *n* is the size of each information nugget which is due to transfer and *L* is the number of frontier divisions, the notification overhead is calculated as shown in Equation (5);

$$Notification\ overhead = n \times m \times (m-1) \times L \tag{5}$$

As depicted in figure 3 the format of each nugget is in a way that the first position from the left is a flag bit. It is set to one if the average of *I(p)* belongs to the last *K/L* set of that parallel process' queue and zero if vice versa. The rest of the nuggets consist of the correspondent parallel process identification number, the *K/L* identification number or the start position of *L* within the queue and the last part is the average of *I(p)* for the *K/L* set respectively. Equation (6) shows the formulation for *n* since it is a dependent variable to *m* and *L* in which α is the size of I_{avg} value in bits.

$$n = 1 + \alpha + Log_2^{mL} \tag{6}$$

Upon receiving the notification nuggets from other parallel processes, the coordinator section of each parallel process compares the *I(p)* average in the received nuggets with that of itself. The coordinator of the parallel process with the best *I(p)* average sends the URLs of the corresponding *K/L* set to the index section and produces a new nugget

for the next *K/L* set. The production of the nuggets by parallel processes is done in a synchronous manner in each *t* seconds intervals. When the time of *t* arrives, only the parallel process which sends a set of URLs to the index is eligible to produce a new nugget and this nugget will be compared to the old received ones. A parallel process that sends its last set of URLs to the index - with the flag bit set to one - does not produce any new notification nugget and the comparison process will be performed among the other parallel processes. This is a normal run of the procedure and any other variation of this rule produces an erroneous condition. Figure 4 depicts an example of a parallel crawler with four parallel processes in which *L*=4. For the matter of simplicity it only depicts the nuggets received by parallel process$_1$ and the first transmitted nugget from parallel process$_1$ in the first iteration of notification transmission.

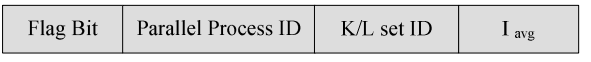

Fig. 3. The structure of the notification nuggets

According to Equation (5) to decrease the notification overhead, the number of parallel processes should not be high. Google employs a few numbers of parallel processes of power two [19]. Moreover the size of each nugget should be maintained as small as possible. Also considering the *L* as a big number causes more numbers of notification nugget transfer among parallel processes and as a result more overhead on the overall system is produced. Upon considering few numbers of parallel processes and an appropriate measure for *L*, the size of each notification nugget will be influenced according to Equation (6).

5 Conclusion

In this paper we propose a crawl frontier prioritizing metric based on clickstream analysis and text analysis within an introduced architecture for a focused structured parallel crawler. The reasons to choose this framework are the limited topic specific search boundary of a focused crawler and its ability to yield more precise answers to the users' information demand and the optimized download rate of a parallel crawler in comparison to a centralized crawler. Besides, in our approach, parallel processes collaborate with each other in the absence of a central coordinator section in order to minimize the inevitable communication overhead and to make parallel processes to operate more independently.

Acknowledgment

The authors wish to thank Ministry of Higher Education Malaysia (MOHE) and Universiti Teknologi Malaysia (UTM), for funding the related research.

References

1. Ahmadi-Abkenari, F., Selamat, A.: Application of Clickstream Analysis in a Tailored Focused Web Crawler. Journal of Communications of SIWN,The Systemic and Informatics World Network (2010)
2. Bharat, K., Henzinger, M.R.: Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In: Proceeding of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 104–111 (1998)
3. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks 30(1-7), 107–117 (1998)
4. Chakrabarti, S.: Mining the Web. In: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, San Francisco (2003)
5. Chackrabarti, S.: Integrating Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information Extraction. In: Proceeding of the 13th international World Wide Web Conference (WWW 2001), pp. 211–220 (2001)
6. Chackrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: Mining the Link Structure of the World Wide Web. IEEE Computer 32(8), 60–67 (1999)
7. Chackrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., Kleinberg, J.: Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In: Proceeding of the 7th international World Wide Web Conference, WWW 2007 (1998)
8. Chakrabarti, S., Van den Berg, M., Dom, B.: Focused Crawling: A New Approach to Topic Specific Web Resource Discovery. Computer Networks 31(11-16), 1623–1640 (1999)
9. Cho, J., Garcia-Molina, H.: Parallel Crawlers. In: Procceding of 11th International Conference on World Wide Web. ACM Press, New York (2002)
10. Cho, J., Garcia-Molina, H., Page, L.: Efficient Crawling through URL Ordering. In: Procceeding of 7th international Conference on World Wide Web (1998)
11. Diligenti, M., Coetzee, F.M., Lawrence, S., Giles, C.L., Gori, M.: Focused Crawling using Context Graph. In: Procceeding of the 26th VLDB Conference, Cairo, Egypt, pp. 527–534 (2000)
12. Giudici, P.: Applied Data Mining, Web Clickstream Analysis. ch.8, pp. 229–253. Wiley Press, Chichester (2003) ISBN: 0-470-84678-X
13. Kleinberg, J.: Authoritative Sources in a Hyperlinked Environment. Journal of the ACM 46(5), 604–632 (1999)
14. Liu, B.: Web Data Mining, Information Retrieval and Web Search. ch.6, pp. 183–215. Springer Press, Heidelberg (2007) ISBN: 3-540-37881-2
15. McCallum, A., Nigam, K.: A Comparison of Event Models for Naïve Baes Text Classification. In: Procceeding of the AAAI-1998 Workshop on Learning for Text Categorization (1998)
16. Menczer, F., Pant, G., Srinivasan, P.: Topical Web Crawlers: Evaluating Adaptive Algorithms. ACM Transactions on Internet Technology 4(4), 378–419 (2004)
17. Selamat, A., Ahmadi-Abkenari, F.: Application of Clickstream Analysis as Web Page Importance Metric in Parallel Crawlers. In: Procceeding of the International Symposium on Information Technology (ITSIM 2010), Kuala Lumpur, Malaysia (2010)
18. Srivastava, A.N., Sahami, M.: Text Mining, Classification, Custering and Applications. CRC Press, Boca Raton (2009)