CRAWL WEB FORUMS (FOCUS)

[1]Vipul Punjabi, [2]Sagarsing Pardeshi, [3]Dipali Pawar, [4]Nayna Chaudhari, [4]Ganesh Gavit
[1]Assistant Professor, [2345]Students of BE (Information Technology)
[1]Department of Information Technology
[1]R. C. Patel Institute of Technology, Shirpur, India

Abstract: ***Present Forum Crawler Under Supervision (FOCUS), a supervised web-scale forum crawler. The goal of FOCUS is to crawl relevant forum content from the web with minimal overhead. Forum threads contain information content that is the target of forum crawlers. Although forums have different layouts or styles and are powered by different forum software packages, they always have similar implicit navigation paths connected by specific URL types to lead users from entry pages to thread pages. Based on this observation, we reduce the web forum crawling problem to a URL- type recognition problem. And we show how to learn accurate and effective regular expression pat-terns of implicit navigation paths from automatically created training sets using aggregated results from weak page types. Web crawler is used for downloading in-formation from web. Web pages are changed without any notice. Web crawler frequently revisits websites to check updates.***

Index Terms – Entry Pages, Thread Pages, EIT Path, ITF Regex.

I. INTRODUCTION

Internet portent services where users can request and exchange forums also called web forums are in-formation with others. For example, the Trip Advisor Travel Board is a place where people can ask and share travel tips. Due to the richness of information in forums, researchers are increasingly interested in mining knowledge from them. Zhai and Liu , Yang, and Song et al. extracted structured data from forums. Gao et al. identified question and answer pairs in forum threads. Zhang et al. proposed methods to extract and rank product features for opinion mining from forum posts. Glance et al. tried to mine business intelligence from forum data. Zhang et al. proposed algorithms to extract expertise network in forums[1].

To harvest knowledge from forums, their content must be downloaded first. However, forum crawling is not a trivial problem. Generic crawlers, which adopt a breadth- first traversal strategy, are usually ineffective and inefficient for forum crawling. This is mainly due to two non-crawler friendly characteristics of forums:

1. Duplicate links and uninformative pages. 2. Page flipping links. A forum typically has many duplicate links that point to a common page but with different URLs, e.g., shortcut links pointing to the latest posts or URLs for user experience functions such as view by date or view by title. A forum also has many uninformative pages such as login control to protect user privacy or forum software specific FAQs[1][2]. Following these links, a crawler will crawl many uninformative pages. Though there are standard-based methods such as specifying the rel attribute with the no follow value (i.e., rel no follow), Robots Exclusion Standard (robots.txt), and Sitemap for forum operators to instruct web crawlers on how to crawl a site effectively, we found that over a set of nine test forums more than 47 percent of the pages crawled by a breadth-first crawler following these protocols were duplicates or uninformative. This number is a little higher than the 40 percent that Cai et al. reported but both show the inefficiency of generic crawlers. More information about this testing can be found in. Besides duplicate links and uninformative pages, a long forum board or thread is usually divided into multiple pages which are linked by page-flipping links. Generic crawlers process each page individually and ignore the relationships between such pages. These relationships should be pre-served while crawling to facilitate down- stream tasks such as page wrapping and content indexing. For example, multiple pages belonging to a thread should be concatenated together in order to extract all the posts in the thread as well as the reply-relationships between posts[2].

In addition to the above two challenges, there is also a problem of entry URL discovery. The entry URL of a forum points to its homepage, which is the lowest common ancestor page of all its threads. Our experiment Evaluation of Starting from Non-Entry URLs in that a crawler starting from an entry URL can achieve a much higher performance than starting from non-entry URLs[2].

The major contributions of system as follows:

[1]We reduce the forum crawling problem to a URL type recognition problem and implement a crawler, FOCUS, to demonstrate its applicability.

[2]We show how to automatically learn regular expression patterns (ITF regexes) that recognize the index URL, thread URL, and page-flipping URL using the page classifiers built from as few as five annotated forums.

[3]We evaluate FOCUS on a large set of 160 unseen forum packages that cover 668,683 forum sites. To the best of our knowledge, this is the largest evaluation of this type. In addition, we show that the learned patterns are effective and the resulting crawler is efficient.

[4]We compare FOCUS with a baseline generic breadth-first crawler, a structure- driven crawler, and a state of-the-art crawler iRobot and show that FOCUS out- performs these crawlers in terms of effectiveness and coverage[2].

[5]We design an effective forum entry URL discovery method. To ensure high coverage, we show that a forum crawler should start crawling forum pages from forum entry URLs. Our evaluation shows that a entry link discovery baseline can achieve only 76 percent recall and precision; while our method can achieve over 99 percent recall and precision.

[6]We show that, though the proposed approach is targeted at forum crawling, the implicit EIT like path also apply to other User Generated Content (UGC) sites, such as community QA sites and blog sites.

A recent and more comprehensive work on forum crawling is iRobot by Cai et al. iRobot aims to automatically learn a forum crawler with minimum human intervention by sampling pages, clustering them, selecting informative clusters via an in formativeness measure, and finding a traversal path by a spanning tree algorithm. However, the traversal path selection procedure requires human inspection. Follow up work by Wang et al. proposed an algorithm to address the traversal path selection problem. They introduced the concept of skeleton link and page-flipping link. Skeleton links are the most important links supporting the structure of a forum site. Importance is determined by in

formativeness and coverage metrics. Page-flipping links are deter-mined using connectivity metric. By identifying and only following skeleton links and page-flipping links, they showed that iRobot can achieve effectiveness and coverage[4].

II. LITERATURE SURVEY

Vidal et al. proposed a method for learning regular expression patterns of URLs that lead a crawler from an entry page to target pages. Target pages were found through comparing DOM trees of pages with a preselected sample target page. It is very effective but it only works for the specific site from which the sample page is drawn. The same process has to be repeated every time for a new site. Therefore, it is not suitable for large-scale crawling. In contrast, FOCUS learns URL patterns across multiple sites and automatically finds a forums entry page given a page from the forum. Experimental results show that FOCUS is effective at large-scale forum crawling by leveraging crawling knowledge learned from a few annotated forum sites[3].

Guo et al. and Li et al. are similar to our work. However, Guo et al. did not mention how to discover and traverse URLs. However, their rules are too specific and can only be applied to specific forums powered by the particular software package in which the heuristics were conceived. Unfortunately, according to Forum Matrix, there is hundreds of different forum software packages used on the Internet. For more information about forum software packages. In addition, many forums use their own customized software[4].

III. METHODOLOGY

In this we discussed about our proposed scheme and how to implement it. We illustrate all the method in separate module with detailed description such as synopsis of anticipated scheme, ITF Regexes Learning, Online Crawling and Entry URL Discovery.

3.1 Overview Proposed Scheme

We present architectural for our anticipated scheme. It consists of two major parts: the learning part and the online crawling part. The learning part first learns ITF regexes of a given forum from automatically constructed URL training examples. The online crawling part then applies learned ITF regexes to crawl all threads efficiently[6].

3.2 Page Type

We classified forum pages into page types.

Entry Page:-

The homepage of a forum, which contains a list of boards and is also the lowest common ancestor of all threads.

Index Page:-

A page of a board in a forum, which usually contains a table-like structure; each row in it contains information of a board or a thread.

Thread Page:-

A page of a thread in a forum that contains a list of posts with user generated content belonging to the same discussion.

Other Page:-

A page that is not an entry page, index page, or thread page.

3.3 URL Type

There are three types of URL:-

Index URL

A URL that is on an entry page or index page and points to an index page. Its anchor text shows the title of its destination board.

Thread URL

A URL that is on an index page and points to a thread page. Its anchor text is the title of its destination thread.

Page-flipping URL

A URL that leads users to another page of the same board or the same thread. Correctly dealing with page-flipping URLs enables a crawler to download all threads in a large board or all posts in a long thread.

3.4 EIT Path

An entry-index-thread path is a navigation path from an entry page through a sequence of index pages (via index URLs and index page-flipping URLs) to thread pages (via thread URLs and thread page-flipping URLs).

3.5 ITF Regex

An index-thread-page-flipping regex is a regular expression that can be used to recognize index, thread, or page-flipping URLs. ITF regex is what FOCUS aims to learn and applies directly in online crawling. The learned ITF regexes are site specific, and there are four ITF regexes in a site: one for recognizing index URLs, one for thread URLs, one for index page-flipping URLs, and one for thread page-flipping URLs[7]. Gives an example. A perfect crawler starts from a forum entry URL and only follows URLs that match ITF regexes to crawl all forum threads. The paths that it traverses are EIT paths.

Constructing URL Training Sets:-

The goal of URL training sets construction is to automatically construct the sets of highly precise index URL, thread URL, and page-flipping URL strings for ITF regexes learning. We use a comparable process to construct index URL and thread URL training sets since they have very comparable properties with the exception of the types of their destination pages.

Learning ITF Regexes:-
This sub-module, we have shown how to construct index URL, thread URL, and page-flipping URL string training set. We also elucidate how to learn ITF regexes from these training sets[7]. Vidal et al. applied URL string generalization. For ex-ample, given URLs as follows (the top four URLs are encouraging while the bottom two URLs are pessimistic).

3.6 Online Crawling

We perform online crawling using a breadth-first strategy (actually, it is easy to adopt other strategies). FOCUS first pushes the entry URL into a URL queue; next it fetches a URL from the URL queue and finally downloads its page; and then it pushes the outgoing URLs which are coordinated with any learned regex into the URL queue. FOCUS repeats this step until the URL queue is empty or other conditions are satisfied. FOCUS only needs to apply the learned ITF regexes on innovative outgoing URLs in newly downloaded pages to making the more pro client for online crawling. FOCUS does not need to group outgoing URLs, classify pages, recognize page-flipping URLs, or learn regexes again for that forum[7][8].

4 Crawling Using Algorithm

Detecting Forum Software :
1. Fetch the entry page of forum.
2. Check for predefined signatures in the fetched page.
3. Decide forum software based on signatures found in the fetched page.

Generating Index URL regex :
1. Select predefined Index URL pattern of detected forum software.
2. Generate regular expressions that matches selected Index URL pattern.

Generating Thread URL regex :
1. Select predefined Thread URL pattern of detected forum software.
2. Generate regular expressions that matches selected Thread URL pattern.

Filtering URLs :
1. Check if the newly obtained URL matches with the Index or Thread URL regex.
2. Ignore URL's that does not matches Index or Thread URL regex.
3. Add all newly obtained thread and index URL's only if they are not already in the list.

Crawling Index Pages :
1. Fetch all index URL's from the queue sequentially.
2. Extract newly obtained index URL's from fetched index pages and add it in queue.

Crawling Thread Pages :
1. Fetch all thread URL's from the queue sequentially.
2. Extract newly obtained thread URL's from fetched thread pages and add it in queue.
3. Fetch thread URL and add it's content to database.

Crawling Flipping URL's :
1. Using predefined signatures of detected forum software extract flipping URLs.
2. Fetch all extracted URLs and extract contents and add it's content to database.

5 Crawling Application

As mentioned, the crawling application we consider in this paper is a breadth-first crawl starting out at a set of seed URLs, in our case the URLs of the main pages of several hundred US Universities, which are initially sent to the crawl manager. The application then parses each downloaded page for hyper-links, checks whether these URLs have already been encountered before, and if not, sends them to the manager in batches of a few hundred or thousand. The downloaded les are then forwarded to a storage manager for compression and storage in a repository. The crawling application is implemented in many languages using STL and the Red-Black tree implementation in the kazlib library. (The application consists of two threads each using a Red-Black tree data structure; this required use of two different implementations since the current implementation in STL is not thread-safe).

The data structure and performance aspects of the application will be discussed. We note however the following important two points: First, since each page contains on average about hyperlinks, the set of en-countered (but not necessarily downloaded) URLs will grow very quickly beyond the size of main memory, even after eliminating duplicates. Thus, after down-loading million pages, the size of the set of encountered URLs will be well above 100 million. Second, at this point, any hyperlink parsed from a newly downloaded page and sent to the manager will only be downloaded several days or weeks later. Thus, there is no reason for the manager to immediately insert new requests into its dynamic data structures[9].

6 Algorithms

6.1 Index URL and Thread URL Detection

Input: sp: an entry or index page
Output: it group: a group of index/thread URLs
Let it group be p:data _
1. url groups=Collect URL groups by aligning HTML DOM tree of sp;
2. for each ug in url groups do
3. uganchor len=Total anchor text length in ug;
5. end for each
6. if group=argmax(ug.anchor len)in url groups;
7. if group.DstPageType=Majority page type of the destination pages of URLs in ug;
8. if group.DstPageType is INDEX PAGE
9. if group.Urltype=INDEX URL;
10. else if if group.DstPageType is THREAD PAGE

11.if group.Urltype=Thread URL;
12.else
13.if group=D;
14.end if
15.return if group;

6.2 Page-Flipping URL Detection

Input: sp: an index page or thread page .
Output: if group: a group of page-flipping URLs
let pf group be

1. url groups=Collect URL groups by aligning HTML DOM tree of sp;
2. for each ug in url groups do
3. if the anchor texts of ug are digit strings
4. pages=Download(URLs in ug);
5. if pages=have the similar layout to sp and ug ap-pears at same location of pages as in sp
6. pf group=ug;
7. break;
8. end if
9. end if
10. end for each
11. if pf group is
12. for each url in outgoing URLs in sp
13. P=Download(url);
14. pf url=Extract URL in p at the same location as url in sp;
16. if pf url exists and pf url.anchor==url.anchor and pf url.UrlString|=url.UrlString
17. Add url and and url into pf group;
18. break;
19. end if
20. end for each
21. end if
22. pf group Url Type=PAGE FLIPPING URL;
23. return pf group;

IV. RESULTS

Here we developed a FOCUS based forum crawler. We designed methods to generate regular expressions of index URLs, thread URLs and page-flipping URLs of forum. Experimental results on widely used forum software packages confirm that our system can effectively crawl forums. Our system automatically detects forum software of the website using predefined FOCUS and selects predefined URL patterns of detected forum software for generating regular expressions of Index URLs, Thread Urls and page-flipping URLs. By following only URLs that matches with the generated regexes we can ensure that it only follows index and thread pages once. Flipping URLs are also extracted using predefined signatures.

IV. ACKNOWLEDGMENT.

We thank Prof. Mr. V. D. Punjabi sir (Assistance Prof. of Information Technology from RCPIT, Shirpur) for his continual expert guidance and encouragement throughout the project.

REFERENCES

[1] R. Cai, J.-M. Yang, W. Lai, Y. Wang, and L. Zhang, iRobot: An Intelligent Crawler for Web Forums, Proc. 17th Intl Conf. World Wide Web, pp. 447-456, 2008.
[2] A. Dasgupta, R. Kumar, and A. Sasturkar, De-Duping URLs via Rewrite Rules, Proc. 14th ACM SIGKDD Intl Conf. Knowledge Discovery and Data Mining, pp. 186-194, 2008.
[3] C. Gao, L. Wang, C.-Y. Lin, and Y.-I. Song, Find-ing Question- Answer Pairs from Online Forums, Proc. 31st Ann. Intl ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 467-474, 2008.
[4] N. Glance, M. Hurst, K. Nigam, M. Siegler, R. Stockton, and T. Tomokiyo, Deriving Marketing Intelligence from Online Discussion, Proc. 11th ACM SIGKDD Intl Conf. Knowledge Discovery and Data Mining, pp. 419-428, 2005.
[5] Y. Guo, K. Li, K. Zhang, and G. Zhang, Board Fo-rum Crawling: A Web Crawling Method for Web Forum, Proc. IEEE/WIC/ACM Intl Conf. Web In-telligence, pp. 475-478, 2006.
[6] M. Henzinger, Finding Near-Duplicate Web Pages: A Large- Scale Evaluation of Algorithms, Proc. 29th Ann. Intl ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 284-291, 2006.
[7] H.S. Koppula, K.P. Leela, A. Agarwal, K.P. Chi-trapura, S. Garg, and A. Sasturkar, Learning URL Patterns for Webpage De-Duplication, Proc. Third ACM Conf. Web Search and Data Mining, pp. 381-390, 2010.
[8] K. Li, X.Q. Cheng, Y. Guo, and K. Zhang, Crawl-ing Dynamic Web Pages in WWW Forums, Com-puter Eng., vol. 33, no. 6, pp. 80-82, 2007.
[9] G.S. Manku, A. Jain, and A.D. Sarma, Detecting Near-Duplicates for Web Crawling, Proc. 16th Intl Conf. World Wide Web, pp. 141-150, 2007.