FoCUS: Learning to Crawl Web Forums

Jingtian Jiang[†*] Nenghai Yu[†] Chin-Yew Lin[‡]

[†]MOE-MS Key Lab of MCC,
University of Science and Technology of China
No. 96 Jinzhai Road, Hefei, China 230026

[‡]Microsoft Research Asia
Building 2, No. 5 Dan Ling St, Haidian Dist.
Beijing, China 100190

silyt@mail.ustc.edu.cn ynh@ustc.edu.cn cyl@microsoft.com

ABSTRACT

In this paper, we present FoCUS (Forum Crawler Under Supervision), a supervised web-scale forum crawler. The goal of FoCUS is to only trawl 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 show how to learn accurate and effective regular expression patterns of implicit navigation paths from an automatically created training set using aggregated results from weak page type classifiers. Robust page type classifiers can be trained from as few as 5 annotated forums and applied to a large set of unseen forums. Our test results show that FoCUS achieved over 98% effectiveness and 97% coverage on a large set of test forums powered by over 150 different forum software packages.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *clustering, information filtering.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Forum Crawling, Page Classification, Page Type, URL Pattern.

1. INTRODUCTION

Internet forums are important platforms where users can request and exchange information with others. For example, the TripAdvisor 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 et al. [21], Yang et al. [20] and Song et al. [16] extracted structured data from forums. Glance et al. [9] tried to mine business intelligence from forum data. Zhang et al. [22] proposed algorithms to extract expertise network in forums. Gao et al. [8] identified question and answer pairs in forum threads. According to an article from eMarketer - Where Are Social Media Marketers Seeing the Most Success? - forums are still part of the global social media strategy of the Top 500 Companies, and they are still getting really high marketing success with forums[1].

To harvest knowledge from forums, their contents have to be downloaded first. Generic crawlers [5], 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 [6] [19]: (1) duplicate links & uninformative pages and (2) page-flipping links. A forum usually has many duplicate links which point to a common page but with different URLs, e.g., shortcut links pointing to latest posts or URLs for user experience functions such as "view by title". A generic crawler that blindly follows these links will trawl many duplicate pages that make it inefficient. A Forum typically has many uninformative pages such as login control to protect users' privacy. Following these links, a crawler will trawl many uninformative pages. Though there are standard-based methods such as specifying the "rel" attribute with "nofollow" value (i.e. "rel=nofollow")[2], Robots Exclusion Standard (robots.txt)[3], and Sitemap[4] [15], for forum operators to instruct web crawlers on how to crawl a site effectively, we found that over a set of 9 test forums more than 47% of the pages trawled by a generic crawler following these protocols are duplicate or uninformative. This number is a little higher than the 40% that Cai et al. [6] reported but both show the inefficiency of generic crawlers.

Besides duplicate links & uninformative pages, a long forum board or thread is usually divided into multiple pages which are linked by page-flipping links, for example, see Figure 2 (b) and (c). Generic crawlers process each page individually and ignore the relationship between such pages. These relationships should be preserved while crawling to facilitate downstream tasks such as page wrapping and content indexing [20]. For example, multiple pages belonging to a thread should be concatenated together in order to extract all posts of this thread as well as the reply-relationships between posts.

In addition to the above challenges, there is also the problem of entry URL discovery. A forum's entry URL points to its home page, which is the lowest common ancestor page of all threads. Our experiment in Section 5.3.2 shows that a crawler starting from an entry URL could achieve much higher performance than starting from other URLs. Previous works by Vidal et al. [18] and Cai et al. [6] assumed that an entry URL is given. But entry URL discovery is not a trivial problem. An entry URL is not necessary at the root URL level of a forum hosting site and its form varies from site to site. Without entry URLs, existing crawling methods such as Vidal et al. [18] and Cai et al. [6] are less effective.

*This work was done when Jingtian Jiang was an intern at Microsoft Research Asia.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

[1] http://www.emarketer.com/Article.aspx?R=1008211
[2] http://en.wikipedia.org/wiki/Nofollow
[3] http://www.robotstxt.org/
[4] http://sitemaps.org/protocol.php

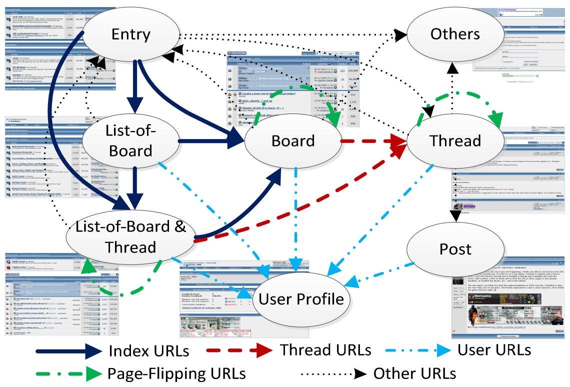


Figure 1. A typical link structure in forums (some links are ignored to show a clear view)

In this paper, we present FoCUS (Forum Crawler Under Supervision), a supervised web-scale forum crawler, to address these challenges. The goal of FoCUS is to trawl relevant content, i.e. user posts, from forums with minimal overhead. Forums exist in many different layouts or styles and powered by a variety of forum software packages, but they always have implicit navigation paths to lead users from entry pages to thread pages. Figure 1 illustrates a typical page and link structure in a forum. For example, a user can navigate from the entry page to a thread page through the following paths:

```
1. entry →board → thread
2. entry →list-of-board →board →thread
3. entry →list-of-board & thread →thread
4. entry →list-of-board & thread →board →thread
5. entry →list-of-board →list-of-board & thread →board
   →thread
6. entry →list-of-board →list-of-board & thread →thread
```

We call pages between the entry page and thread page which are on a breadth-first navigation path the *index page*. We represent these implicit paths as the following navigation path (EIT path):

```
entry page → index page → thread page
```

Links between an entry page and an index page or between two index pages are referred as *index URLs*. Links between an index page and a thread page are referred as *thread URLs*. Links connecting multiple pages of a board and multiple pages of a thread are referred as *page-flipping URLs*. A crawler starting from the entry page of a forum only needs to follow index URLs, thread URLs, and page-flipping URLs to traverse EIT path and achieve all thread pages. The challenge of forum crawling is then reduced to a URL type recognition problem. In this paper, we show how to learn regular expression patterns, i.e. ITF regexes, recognizing these three types of URLs from as few as 5 annotated forum packages and apply them to a large set of 160 unseen forums packages. Note that we specifically refer to "forum package" rather than "forum site". A forum software package such as vBulletin[5] can be deployed by many forum sites.

The major contributions of this paper are 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 5 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 scale evaluation of this type. In addition, we show that the 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 outperforms these crawlers in terms of effectiveness and coverage.
5. We design an effective forum entry URL discovery method. Entry URLs need to be specified to start crawling to get higher recall. But entry page discovery is not a trivial task since entry pages vary from forums to forums. Our evaluation shows that a naïve baseline can achieve only 76% recall and precision; while our method can achieve over 95% recall and precision.

The rest of this paper is organized as follows. Section 2 is a brief review of related work. In Section 3, we define terms used in this paper. We report our observations which motivate our method and describe the detail of the proposed method in Section 0. In Section 5, we report results of our experiments. In the last section, we draw conclusions and point out future directions of research.

2. RELATED WORK

Vidal et al. [18] 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 pre-selected 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 to large- scale crawling. In contrast, FoCUS learns URL patterns across multiple sites and automatically finds forum entry page given a page from a forum. Experimental results show that FoCUS is effective in large scale forum crawling by leveraging crawling knowledge learned from a few annotated forum sites.

Guo et al. [10] and Li et al. [13] are similar to our work. However, Guo et al. did not mention how to discover and traverse URLs. Li et al. developed some heuristic rules to discover URLs, but their rules are too specific and can only be applied to specific forums powered by the particular forum software package in which the heuristics were conceived. Unfortunately, according to ForumMatrix [1], there are hundreds of different forum software packages on the internet. Please refer to [1] [2] [3] for more information about different forum software packages. In addition, many forums use their own customized software.

A recent and more comprehensive work on forum crawling is iRobot by Cai et al. [6]. iRobot aims to automatically learn a forum crawler with minimum human intervention by sampling forum pages, clustering them, selecting informative clusters via an informativeness 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. [19] 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". Important is determined by informativeness metric and coverage metric. Page-

[5] Please see www.forummatrix.org for a list of forum packages.

flipping links are determined using connectivity metric. By identifying and only following skeleton links and page-flipping links, they showed that iRobot can achieve good effectiveness and coverage. However, according to our evaluation, its sampling strategy and informativeness estimation is not always robust and its tree-like traversal path does not allow more than one path from a starting page node to a common ending page node. For example, as shown in Figure 1, there are 6 paths from entry to thread. But iRobot would only take the first path (entry → board → thread). iRobot learns URL location information to discover new URLs in crawling, but a URL location might become invalid when the page structure changes. Comparing with iRobot, we explicitly define EIT (entry-index-thread) paths. FoCUS leverages page layouts to identify index pages and thread pages. FoCUS learns precise URL string patterns instead of URL locations to discover new URLs. Thus it does not need to classify new pages in crawling and would not be affected by a change in page structures. The respective results between iRobot and FoCUS demonstrated that the EIT path and URL string patterns are more robust than the traversal path and URL location feature in iRobot.

Another related work is near-duplicate detection. Forum crawling also needs to remove duplicates. But content-based duplicate detection [11] [14] is not bandwidth-efficient, because it can only be carried out when pages have been downloaded. URL-based duplicate detection [4] [7] [12] is not helpful. It tries to mine rules of different URLs with similar text. But such methods still need to analyze logs from target sites or results of a previous crawl. In this paper, through detecting index, thread and page-flipping URLs, FoCUS could avoid duplicates without duplicate detection.

3. TERMINOLOGY

To facilitate presentation in the following sections, we first define some terms used in this paper.

Page Type. We classified forum pages into four page types:

Entry Page: A page that is the lowest common ancestor of all thread pages in a forum. See Figure 2 (a).

Index Page: A page that contains a table-like structure; each row in it contains information on URLs pointing to a board or a thread. See Figure 2 (b). In Figure 1, list-of-board page, list-of-board & thread page, board page are all index pages.

Thread Page: A page that contains a list of posts with user-generated content (UGC). See Figure 2 (c).

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

URL Type. There are four types of URLs:

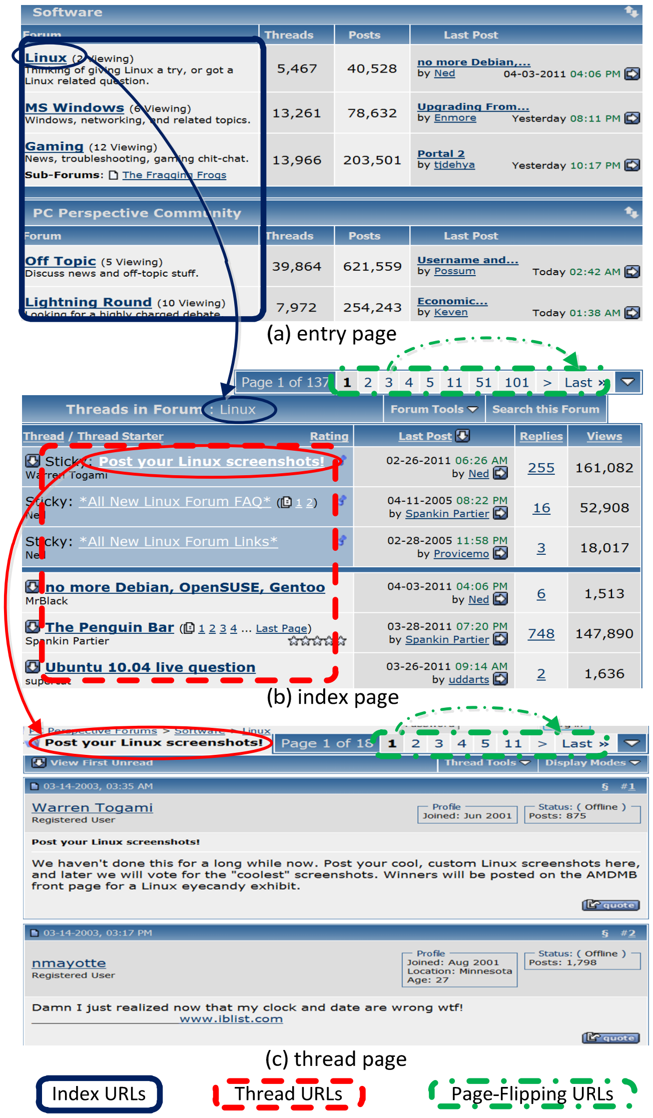
Index URL: A URL that is on an entry page or index page and points to an index page. Its anchor text shows the title of its destination board. Figure 2 (a) and (b) show an example.

Thread URL: A URL that is on an index page and points to a thread page. Its anchor text shows the title of its destination thread. Figure 2 (b) and (c) show an example.

Page-flipping URL: A URL that leads users to another page of a same board or a 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. See Figure 2 (b) and (c) for examples.

Other URL: A URL is not index, thread, or page-flipping URL.

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



Software

Forum	Threads	Posts	Last Post
Linux (2 Viewing) Thinking of giving Linux a try, or got a Linux related question.	5,467	40,528	no more Debian,... by Ned 04-03-2011 04:06 PM
MS Windows (6 Viewing) Windows, networking, and related topics.	13,261	78,632	Upgrading From... by Enmore Yesterday 08:11 PM
Gaming (12 Viewing) News, troubleshooting, gaming chit-chat. Sub-Forums: The Fragging Frogs	13,966	203,501	Portal 2 by tjdehya Yesterday 10:17 PM

PC Perspective Community

Forum	Threads	Posts	Last Post
Off Topic (5 Viewing) Discuss news and off-topic stuff.	39,864	621,559	Username and... by Possum Today 02:42 AM
Lightning Round (10 Viewing) Looking for a highly charged debate	7,972	254,243	Economic... by Keven Today 01:38 AM

(a) entry page

Page 1 of 137 1 2 3 4 5 11 51 101 > Last »

Threads in Forum : Linux Forum Tools Search this Forum

Thread / Thread Starter	Rating	Last Post	Replies	Views
Sticky: Post your Linux screenshots! Warren Togami		02-26-2011 06:26 AM by Ned	255	161,082
Sticky: *All New Linux Forum FAQ* (1 2) Ned		04-11-2005 08:22 PM by Spankin Partier	16	52,908
Sticky: *All New Linux Forum Links* Ned		02-28-2005 11:58 PM by Provicemo	3	18,017
no more Debian, OpenSUSE, Gentoo MrBlack		04-03-2011 04:06 PM by Ned	6	1,513
The Penguin Bar (1 2 3 4 ... Last Page) Spankin Partier	☆☆☆☆☆	03-28-2011 07:20 PM by Spankin Partier	748	147,890
Ubuntu 10.04 live question supercat		03-26-2011 09:14 AM by uddarts	2	1,636

(b) index page

PC Perspective Forums > Software > Linux
Post your Linux screenshots! Page 1 of 18 1 2 3 4 5 11 > Last »
View First Unread Thread Tools Display Modes

03-14-2003, 03:35 AM § #1
Warren Togami
Registered User
Profile Joined: Jun 2001
Status: (Offline) Posts: 875
Post your Linux screenshots!
We haven't done this for a long while now. Post your cool, custom Linux screenshots here, and later we will vote for the "coolest" screenshots. Winners will be posted on the AMDMB front page for a Linux eyecandy exhibit.
quote

03-14-2003, 03:17 PM § #2
nmayotte
Registered User
Profile Joined: Aug 2001 Location: Minnesota Age: 27
Status: (Offline) Posts: 1,798
Damn I just realized now that my clock and date are wrong wtf!
www.iblist.com
quote

(c) thread page

Figure 2. An instance of EIT path: entry → board → thread

ITF Regex. An ITF (index-thread-page-flipping) regex is a regular expression that used to recognize index, thread, or page-flipping URLs on EIT path. ITF regexes is what FoCUS aims to learn and applies directly in online crawling.

4. FoCUS – A Supervised Forum Crawler

In this section, we first motivate and give an overview of our approach. The remaining sections deep dive into each module.

4.1 Observations

In order to crawl forum threads effectively and efficiently, we investigated about 40 forums (not used in testing) and found the following characteristics in almost all of them:

a. **Navigation Path**: Despite differences in layout and style, forums always have similar implicit navigation paths leading users from their entry pages to thread pages. In general web crawling, Vidal et al. [18] learned "navigation patterns" leading to target pages (thread pages in our case). iRobot also adopted similar idea but applied page sampling and clustering techniques to find target pages (Cai et al. [6]). It used informativeness and coverage metrics to find navigation paths (or traversal paths in Wang et al. [19]). We explicitly defined EIT path that specifies what types of URL and page that a crawler should follow to reach thread pages.

b. **URL Layout**: URL layout information such as the location of a URL on a page and its anchor text length is an important indicator of its function. URLs of the same function usually appear at the same location. For example, in Figure 2 (a) and (b), index URLs appear in the left rectangles. In addition, index URLs and thread URLs usually have longer anchor texts that provide board or thread titles (see Figure 2 (a) and (b)).

c. **Page Layout**: Index pages from different forums share similar layout. The same applies to thread pages. However, an index page usually has very different page layout from a thread page. An index page tends to have many narrow records giving information about boards or threads. A thread page typically has a few large records that contain user posts. iRobot used this feature to cluster similar pages together and apply its informativeness metric to decide whether a set of pages should be crawled. FoCUS learns page type classifiers directly from a set of annotated pages based on this characteristic. This is the only part where manual annotation is required for FoCUS.

Inspired by these observations, we developed FoCUS. The main idea behind FoCUS is that index URL, thread URL, and page-flipping URLs can be detected based on their layout characteristics and destination pages; and forum pages can be classified by their layouts. This knowledge about URLs and pages and forum structures can be learned from a few annotated forums and then applied to unseen forums. Our experimental results in Section 5 confirm the effectiveness of our approach.

4.2 System Overview

Figure 3 shows the overall architecture of FoCUS. It consists of two major parts: the learning part and the online crawling part. The learning part learns ITF regexes of a given forum from automatically constructed URL examples. The online crawling part applies learned ITF regexes to crawl all threads efficiently.

Given any page of a forum, FoCUS first finds its entry URL using *Entry URL Discovery* module. Then, it uses the *Index/Thread URL Detection* module to detect index URLs and thread URLs on the entry page; the detected index URLs and thread URLs are saved to the URL training set. Next, the destination pages of the detected index URLs are feed to this module again to detect more index URLs and thread URLs until no more index URL detected. After that, the *Page-Flipping URL Detection* module tries to find page-flipping URLs in both index pages and thread pages and saves them to the training set. Finally, the *ITF Regexes Learning* module learns a set of ITF regexes from the URL training set.

FoCUS performs online crawling as follows: it first pushes the entry URL into a URL queue; next it fetches a URL from the queue and downloads its page, and then pushes the outgoing URLs that are matched with any learned ITF regex into the URL queue. This step is repeated until the URL queue is empty.

4.3 Learning ITF Regexes

To learn ITF regexes, FoCUS adopts a two-step supervised training procedure. The first step is training set construction. The second step is regex learning.

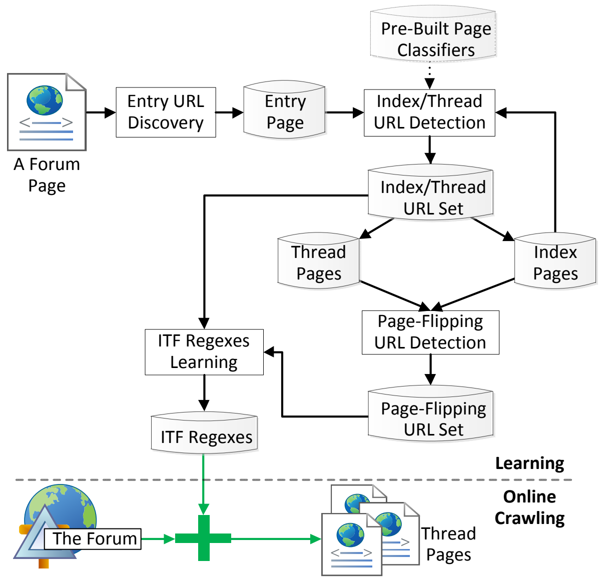


Figure 3. Overall architecture of FoCUS

4.3.1 Constructing URL Training Set

The goal of training set construction is to automatically create sets of highly precise index URL, thread URL, and page-flipping URL string samples for regex learning. We use a similar procedure to construct index URL and thread URL training sets since they have very similar properties except the types of their destination pages; we present this part first. Page-flipping URL strings have their own specific properties which are different from properties of index URL and thread URL strings; we present this part later.

4.3.1.1 Index and Thread URL String Training Sets

Recall that an index URL is a URL that is on an entry page or index page; and its destination page is another index page; while a thread URL is a URL that is on an index page; and its destination page is a thread page. We also note that the only way to distinguish index URLs from thread URLs is the type of their destination pages. Therefore, we need a method to decide page type of a destination page.

As we mentioned in Section 4.1, the index page and thread page have their own typical layouts. Usually, an index page has many narrow records, relatively long anchor text and short plain text; while a thread page has a few large records, usually user posts or merchant advertisements. Each post has a very long text block and relatively short anchor text. An index page or a thread page always has a timestamp field in each record, but the timestamp order in the two types of pages are reversed: the timestamps are typically in descending order in an index page while they are in ascending order in a thread page. In addition, each record in an index page or a thread page usually has a link pointing to a user profile page (See Figure 2 for example).

Inspired by such characteristic, we propose features representing page layouts as shown in Table 1 and build page classifiers using Support Vector Machine (SVM) [17] to decide page type. SVM^{light} Version 6.02[6] with a default linear kernel setting is used to build our classifiers. One index page classifier and one thread page classifier are built using the same feature set. FoCUS does not need strong page type classifiers, as we will explain later.

[6] Please see svmlight.joachims.org for details about SVM^{light}.

Table 1. Main features used in index/thread page classification

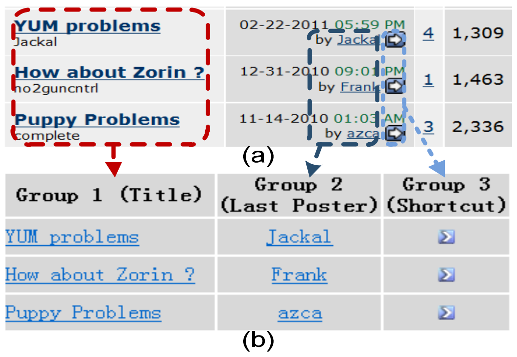
Feature Name	Value Type	Short Description
Record Count	Float	Number of records
Max Anchor Text Length	Float	The max length in characters of anchor text per record
Max Height	Float	The max value of height of each record
Max Text Length	Float	The max length in characters of plain text per record
Average Text Length	Float	The average length in characters of plain text among all records
Has Timestamp	Boolean	Whether each record has a timestamp
Time Order	Float	The order of timestamps in the records if the timestamps exist
Has User Link	Boolean	Whether each record has a link pointing to a user profile page

In our experiment over 160 forum sites (10 pages each of index, thread, and other page) each powered by a different forum software package, our classifiers achieved 96% recall and 97% precision for index page and 97% recall and 98% precision for thread page with different amount of training data (see Table 2).

After showing how to detect index and thread pages, we next describe how to find index URLs and thread URLs on an entry page or index page. Recall again the definition of index (or thread) URL that its anchor text is the board (or thread) title of its destination page. This is illustrated in Figure 2 (a) and (b) where index (or thread) URLs usually have relatively long anchor text, are grouped according to their functions, and placed at the same "table" column position. See also Figure 4 (a). Leveraging such structured layout information, we group URLs based on their locations and treat URLs as a group instead of as individual URLs. We then assume that the URL group with the longest total anchor text is a candidate group of index (or thread) URLs. This simple group anchor-text-length-based discriminative method does not need a length threshold to decide the type of a URL and can take care of index (or thread) URL with short anchor text.

According to [21] and [16], URLs that appear in an HTML table-like structure can be extracted by aligning DOM trees and stored in a link-table. This technique has been well studied in recent years, so we will not discuss it here. In this paper, we adopted the partial tree alignment method in [16]. Figure 4 shows an example of the table-like structure in an index page and its corresponding link-table. Figure 4 (a) is a screenshot of an index page, in which each row contains the thread title, the last poster, a shortcut to the last post, the number of posts, and the number of views. After aligning the DOM tree in Figure 4 (a), a link-table is generated as shown in Figure 4 (b). The text and images with their URLs are aligned and stored in the link-table while the plain text (number of posts and views) is discarded. In Figure 4, thread titles with their URLs are aligned to group 1, the most recent posters with their URLs are aligned to group 2, and the shortcuts are aligned to group 3. Based on our assumption of index or thread URL groups, we will select group 1 as the candidate for the index or thread URL group because it has the longest anchor text.

Note that we cannot determine the type of the candidate group so far. We need to check the destination page type of the URLs in the candidate group. A majority voting method is adopted to determine the type since classification results on individual page might be erroneous. By utilizing aggregated classification results, FoCUS does not need strong page classifiers. In summary, we create index and thread URL training strings using Algorithm 1.



YUM problems Jackal	02-22-2011 05:59 PM by Jackal	4	1,309
How about Zorin ? no2guncntrl	12-31-2010 09:01 PM by Frank	1	1,463
Puppy Problems complete	11-14-2010 01:03 AM by azca	3	2,336

Group 1 (Title)	Group 2 (Last Poster)	Group 3 (Shortcut)
YUM problems	Jackal	➡
How about Zorin ?	Frank	➡
Puppy Problems	azca	➡

Figure 4. An example of URL grouping (a) an HTML "table" containing thread information (b) grouped URLs

Algorithm 1. Index/thread URL detection

```
Input:    sp: an entry page or index page
Output:   it_group: a group of index/thread URLs
 1: let it_group be □;
 2: url_groups = Collect URL groups by aligning DOM tree of sp;
 3: foreach ug in url_groups do
 4:     ug.AnchorLen = Total anchor text length in ug;
 5: end foreach
 6: it_group = arg max( ug.AnchorLen ) in url_groups;
 7: it_group.DstPageType = Majority type of destination pages;
 8: if it_group.DstPageType is INDEX_PAGE
 9:     it_group.UrlType = INDEX_URL;
10: else if it_group.DstPageType is THREAD_PAGE
11:     it_group.UrlType = THREAD_URL;
12: else
13:     it_group = □;
14: end if
15: return it_group;
```

Algorithm 2. Page-flipping URL detection

```
Input:    sp: an index page or thread page
Output:   pf_group: a group of page-flipping URLs
 1: let pf_group be □;
 2: url_groups = Collect URL groups by aligning DOM tree of sp;
 3: foreach ug in url_groups do
 4:   if the anchor text of ug are digit strings //property 1)
 5:     pages = Download( URLs in ug );
 6:     if ug appears at same location in pages as in sp//property 2)
          and pages have the similar layout to sp //property 3)
 7:       pf_group = ug;
 8:       pf_group.UrlType = PAGE_FLIPPING_URL;
 9:       break;
10:     end if
11:   end if
12: end foreach
13: return pf_group;
```

4.3.1.2 Page-Flipping URL String Training Set

Page-flipping URLs point to index pages or thread pages but they are very different from index URLs or thread URLs. For example, URLs in the rounded rectangles in top-right corner of Figure 2 (b) and (c) are page-flipping URLs. They have following properties:

1) Their anchor text is either a sequence of digits such as 1, 2, 3, or special text such as "last";
2) They appear at the same location in the DOM tree of their destination pages as in their source page;

3) Their destination pages have similar layout with their source page. We use tree similarity to determine whether the layouts of two pages are similar or not;

Our page-flipping URL detection module works based on above properties. The detail is shown as Algorithm 2. In our experiment over 160 forum sites (10 pages each of index and thread page), our method achieved 93% recall and 99% precision, proving this simple method is quite effective. The found page-flipping URLs are saved as training examples.

4.3.2 Learning ITF Regexes

We have shown how to create index URL, thread URL, and page-flipping URL string training sets; next we explain how to learn ITF regexes from these training sets.

Vidal et al. [18] applied URL string generalization, but we do not use their method because it is too strict and is easily affected by negative URL examples. It requires very clean precise URL examples. However, FoCUS cannot guarantee this since its training sets are all created automatically. For example, given URL examples as follows (the top 4 URLs are positive while the bottom 2 URLs are negative):

http://www.gardenstew.com/about20152.html
http://www.gardenstew.com/about18382.html
http://www.gardenstew.com/about19741.html
http://www.gardenstew.com/about20142.html
http://www.gardenstew.com/user34.html
http://www.gardenstew.com/post180803.html

It creates a URL regular expression pattern as follows: http://www.gardenstew.com/\w+\d+.html; while the target pattern is http://www.gardenstew.com/about\d+.html. Instead, we apply the method introduced by Koppula et al. [12] which is better able to deal with negative examples. We briefly describe their technique below. For more details, please refer to their paper.

Starting with the generic pattern ‘*’, their algorithm finds more specific patterns matching a set of URLs. Then each specific pattern is further refined into more specific patterns. Patterns are refined recursively until no more patterns could be refined. Given the previous example, ‘*’ is refined to one specific pattern http://www.gardenstew.com/\w+\d+.html that matches all URLs. Then this pattern is refined to three more specific patterns:

1. http://www.gardenstew.com/about\d+.html
2. http://www.gardenstew.com/user\d+.html
3. http://www.gardenstew.com/post\d+.html

Each pattern matches a subset of URLs. These patterns are refined recursively until no more specific patterns could be generated. These patterns are final output as they cannot be refined further.

We made one modification of the technique: a refined pattern is retained only if the number of its matching URLs is greater than an empirically determined threshold. This is designed to reduce patterns with very low coverage since we expect a correct pattern should cover many URLs. The threshold is set to 0.2 times the total count of URLs. It was determined based on 5 training forums. For the above example, only the first pattern is retained.

In practice, this method might include session ID[7] in learned URL patterns. We issue multiple requests of a URL over a span of time to detect any embedded session ID and exclude it from URLs.

[7] http://en.wikipedia.org/wiki/Session_ID

4.4 Online Crawling

Given a forum, FoCUS first learns a set of ITF regexes following the procedure described in the previous sections. Then it performs online crawling using a breadth-first strategy. It first pushes the entry URL into a URL queue; next it fetches a URL from the URL queue and downloads its page; and then it pushes the outgoing URLs that are matched with any learned regex into the URL queue. FoCUS repeats this step until the URL queue is empty or other conditions are satisfied.

What makes FoCUS efficient in online crawling is that it only needs to apply the learned ITF regexes on outgoing URLs in newly downloaded pages. FoCUS does not need to group outgoing URLs, classify pages, detect page-flipping URLs, or learn regexes again for that forum. Such time consuming operations are only performed during its learning phase.

4.5 Entry URL Discovery

In the previous sections, we explained how FoCUS learns ITF regexes that can be used in online crawling to determine what URLs to follow and what URLs to ignore. However, an entry page needs to be specified to start the crawling process. To the best of our knowledge, all previous methods assumed a forum entry page is given. In practice, especially in web-scale crawling, manual forum entry page annotation is not practical. Forum entry page discovery is not a trivial task since entry pages vary from forums to forums. Our experiment shows that a naïve baseline method can achieve only about 76% recall and precision. To make FoCUS more practical and scalable in web-scale crawling, we design a simple yet effective forum entry URL discovery method based on some techniques introduced in previous sections.

We observe that (1) almost every page contains a link to lead users back to the entry page of a forum; (2) an entry page has most index URLs since it leads users to all forum thread pages. Based on the index URL detection module and the index page classifier described the Section 4.3.1.1, we proposed a method for finding the entry page for a forum given a URL pointing to any page of the forum. The method's detail is shown in Algorithm 3. Our evaluation on 160 forums shows that this method can achieve 98% precision and 95% recall.

Algorithm 3. Entry URL discovery

```
Input:   page: a forum page from a forum
Output:  entry_url: Entry URL of this forum
 1: cand_urls = Extract outgoing URLs in page; // candidate URLs
 2: sel_urls = Randomly select a few URLs from cand_urls;
 3: foreach u in sel_urls do
 4:     page = Download( u ); // every page has an entry URL
 5:     cand_urls = cand_urls ∩ {outgoing URLs in page};
 6: end foreach
 7: let entry_url be empty, count be 0;
 8: foreach u in cand_urls do
 9:     page = Download( u );
10:     index_urls = Detect index URLs in page;
11:     if count < |index_urls|
12:         count = |index_urls|;
13:         entry_url = u; // entry page has most index URLs
14:     end if
15: end foreach
16: return entry_url;
```

5. EXPERIMENTS

To carry out meaningful evaluations that are good indicators of web-scale forum crawling, we selected 200 different forum software packages from ForumMatrix [1], Forum Software [2], and Big-Boards [3]. For each software package, we found a forum site powered by it. In total, we have 200 forums powered by 200 different software packages. Among them, we selected 40 forums as our training set and leave the remaining 160 for testing.

These 200 software packages cover a large number of forum sites. The 40 training packages are deployed by 59,432 forum sites and the 160 test packages are deployed by 668,683 forum sites. To the best of our knowledge, this is the most comprehensive investigation of forum crawling in terms of forum site coverage to date. In addition, we wrote scripts to find out how many threads, posts, and users are in these forums. In total, we estimated that these packages cover about 2.7 billion threads generated by over 986 million users.

It should be noted that according to our statistics, on all forum sites that we have found, the top 10 most frequent software packages are deployed by 17% of all forum sites and cover about 9% of all threads.

In the evaluation of page type classification, index/thread URL detection, and entry page discovery, we selected the training/test pages and checked results manually as the data set is not very large (no more than 5,000 pages). But in the evaluation of online crawling, it's impossible to manually verify whether all the crawled pages are thread page or not. Instead, we wrote a set of URL based rules to detect index and thread pages for each forum. We used these rule sets to check the page type of crawled pages. To ensure that our rule sets can be used as an alternative to manual evaluation, we used each rule set to annotate 200 pages randomly selected from its corresponding forum. We found that these rule sets achieved at least 99% precision and 100% recall.

5.1 Efficiency of FoCUS

We evaluated the efficiency of FoCUS in terms of the number of pages crawled and the time spent during its learning phase.

Similar to structure-driven crawler and iRobot, FoCUS fetches some pages during its learning phase. We evaluated how many pages FoCUS needed to visit to get satisfactory learning results. Among the 160 test forums used in our evaluation, the maximum number of pages FoCUS visited is 2,663, the minimum number is 252, and the average number is 442. We found that the structure-driven crawler needed to visit at least 3,000 pages to find navigation patterns, which is larger than Vidal et al. [18] reported; iRobot [6] needed to sample at least 2,000 pages to find complete traversal paths.

To estimate the time spent on a forum during the learning phase, we ran FoCUS on a machine with two 4-core 2.20 GHz CPUs, 32 GB memory, and 64-bit Windows Server 2008 SP1 OS. The maximum time spent on a forum is 3,416 seconds, the minimum time was 25 seconds, and the average was 466 seconds. According to our experience, even a skilled human would spend about 1,200 seconds on average to write URL rules for a forum.

5.2 Evaluations of FoCUS Modules

5.2.1 Evaluation of Index/Thread URL Detection

To build page classifiers, we manually selected 5 index pages, 5 thread pages, and 5 other pages from each of the 40 forums and extracted the features listed in Table 1. For testing, we manually

Table 2. Result of page type classification and URL detection

#Train Forum	Index Page %				Thread Page %				Index/Thread URL Detection %			
	Precision		Recall		Precision		Recall		Precision		Recall	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
5	97.51	0.83	96.98	1.33	98.24	0.55	98.12	1.15	99.02	0.17	98.14	0.21
10	97.05	0.69	97.47	1.56	98.28	0.27	98.04	1.23	99.01	0.15	98.13	0.18
20	97.23	0.20	96.91	1.38	98.43	0.44	97.96	1.49	99.01	0.15	98.08	0.17
30	97.34	0.18	96.18	0.56	98.66	0.26	98.00	1.18	99.00	0.10	98.10	0.12
40	97.44	N/A	96.38	N/A	99.04	N/A	97.49	N/A	99.03	N/A	98.12	N/A

Table 3. Results of entry page discovery

Method	Precision %		Recall %	
	Average	Std. Dev.	Average	Std. Dev.
Baseline	76.38	1.74	76.38	1.74
FoCUS	98.08	0.85	95.81	0.59

selected 10 index pages, 10 thread pages and 10 other pages from each of the 160 forums. This is called 10-Page/160 test set. We then ran the index/thread URL detection module described in Section 4.3.1.1 on the 10-Page/160 test set and manually checked whether the detected URLs are correct. Note that we computed the results at the page level not at the individual URL level since we applied a majority voting procedure.

To further examine how many annotated pages FoCUS needs to achieve a good performance, we conducted similar experiments but with more training forums (10, 20, 30, and 40) and applied cross validation. Table 2 shows the results. We find that our classifiers achieved over 96% recall and precision at all cases with tight standard deviation. It is particularly encouraging to see that FoCUS can achieve over 98% precision and recall in index/thread URL detection with only as few as 5 annotated forums.

5.2.2 Evaluation of Page-Flipping URL Detection

We applied the module described in Section 4.3.1.2 on the 10-Page/160 test set and manually checked whether it found the correct page-flipping URLs. The method achieved 99% precision and 93% recall. The failure is mainly due to JavaScript-based page-flipping URLs or HTML DOM tree alignment error.

5.2.3 Evaluation of Entry URL Discovery

As far as we know, all prior works in forum crawling assume that an entry page is given. However, finding forum entry page is not trivial. To demonstrate this, we compare our entry page detection method with a heuristic baseline. The heuristic baseline tries to find the following keywords ending with '/' in a URL: *forum*, *board*, *community*, *bbs,* and *discus*. If a keyword is found, the path from the URL host to this keyword is extracted as its entry page URL; if not, the URL host is extracted as its entry page URL.

For each forum in the test set, we randomly sampled a page and fed it to this module. Then we manually checked if its output was indeed its entry page. In order to see whether FoCUS and the baseline are robust, we repeated this procedure 10 times with different sample pages. The result is shown in Table 3. The baseline had 76% precision and recall. On the contrary, FoCUS achieved 98% precision and 95% recall. The low standard deviation also indicates that it is not sensitive to sample pages. There are two main failure cases: 1) forums are no longer in operation and 2) JavaScript generated URLs which we do not handle currently.

Table 4. Forums used in online crawling evaluation

ID	Forum	Software	#Threads
1	http://forums.afterdawn.com/	Customized	535,383
2	http://forums.asp.net/	CommunityServer	66,966
3	http://forum.xda-developers.com/	vBulletin	299,073
4	http://bbs.cqzg.cn/	Discuz!	428,555
5	http://forums.crackberry.com/	vBulletin V2	525,381
6	http://forums.gentoo.org/	phpBB V2	681,813
7	http://lkcn.net/bbs/	IP.Board	180,692
8	http://www.techreport.com/forums/	phpBB	65,083
9	http://www.redandwhitekop.com/forum/	SMF	138,963

5.3 Evaluation of Online Crawling

We have shown in the previous sections that FoCUS is efficient in learning ITF regexes and is effective in detection of index URL, thread URL, page-flipping URL, and forum entry URL. In this section, we compare FoCUS with other existing methods in terms of effectiveness and coverage (defined later).

We selected 9 forums (Table 4) from among the 160 test forums for this comparison study. 8 of the 9 forums are popular software packages used by many forum sites (plus one customized package used by afterdawn.com). These software packages cover 388,245 forums. This is about 53% of forums powered by the 200 packages studied in this paper, and about 15% of all forums we have found.

We now define effectiveness and coverage metric. Effectiveness measures the percentage of thread pages among all pages crawled:

$$Effectiveness = \frac{\#Crawled\ threads}{\#Crawled\ pages} \times 100\%$$

Coverage measures the percentage of crawled thread pages of a forum crawled to all retrievable thread pages of the forum:

$$Coverage = \frac{\#Crawled\ threads}{\#Threads\ in\ all} \times 100\%$$

Ideally, we would like to have 100% effectiveness and 100% coverage when all threads of a forum are crawled and only thread pages are crawled. A crawler can have high effectiveness but low coverage or low effectiveness and high coverage. For example, a crawler can only trawl 10% of all thread pages, i.e. 10% coverage, with 100% effectiveness; or a crawler needs to trawl 10 times the thread pages, i.e. 10% effectiveness to reach 100% coverage.

In order to make a fair comparison, we have mirrored the 160 test forums by a brute-force crawler. We also checked the forums to find out the number of threads in each forum. All the following crawling experiments were simulated on the mirrored data set.

5.3.1 Evaluation of a Generic Crawler

To show the hardness of the challenges in forum crawling, we implemented a generic breadth-first crawler following the protocols of "nofollow" and robots.txt. This crawler also recorded the URLs with the attribute "rel=nofollow" or which were disallowed by robots.txt but did not visit them.

Figure 5 shows the ratio of thread URLs, uninformative & duplicate URLs, URLs disallowed by robots.txt and URLs with "rel=nofollow". We can see that "nofollow" is only effective on 3 forums while robots.txt is effective on 6 forums. Neither nofollow nor robots.txt is effective on 2 forums as they are not used on the 2 forums. Even though they help a generic crawler avoid a lot of pages on 7 forums, the crawler still visited many uninformative & duplicate pages. To show that clearly, the effectiveness of this crawler is shown in Figure 8 and coverage is shown in Figure 9.

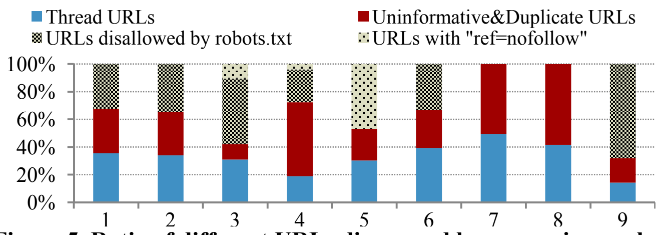


Figure 5. Ratio of different URLs discovered by a generic crawler

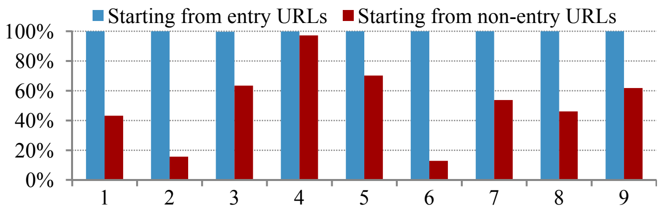


Figure 6. Coverage comparison between starting from entry URL and non-entry URL

The coverage on all forums is almost 100%, but the average effectiveness is around 53%. The best effectiveness is about 74% on "xda-developers (3)". It's because this forum better maintained robots.txt than other forums. This showed that "nofollow" and robots.txt did help forum crawling, but not enough. We can conclude that a generic crawler is less effective and not scalable for forum crawling, and its performance depends on how well the "nofollow" and robots.txt is maintained.

5.3.2 Evaluation of Starting from Non-Entry URLs

As we discussed earlier, a crawler starting from the entry URL cab achieve higher coverage than when starting from other URLs. We also did an experiment to verify this. We used the generic crawler in Section 5.3.1, but set it only follow the index URL, thread URL and page-flipping URL. So it is an ideal forum crawler with 100% effectiveness. It starts from the entry URL and a randomly selected non-entry URL respectively. The crawler stopped when no more pages could be retrieved. We also repeat this experiment with different non-entry URLs.

The results are shown in Figure 6 (we did not show the effectiveness as it was 100%). When starting from entry URL, the coverage is all close to 100%. But when starting from a non-entry URL, the coverage decreased significantly. The average coverage is about 52%. The coverage on "cqzg (4)" is very high. This is because there are many cross-board URLs in that forum. That is, a page of one board contains URLs pointing to other boards. But in other forums, there are fewer cross-board URLs. Thus the crawler could only find the threads in the board to which the starting URL belongs. This experiment showed that an entry URL is critical for forum crawling, and automatic entry URL discovery is necessary to make a crawler scalable.

5.3.3 Evaluation of Structure-Driven Crawler

Although the structure-driven crawler [18] is not a forum crawler, it could be applied to forums. To make it a more meaningful comparison, we adapted it to find page-flipping URL patterns in order to increase its coverage.

The results of it are shown in Figure 7. We observe that it performed well on "cqzg (4)" and "techreport (8)". On these 2 forums, it found the exact patterns of index URL, thread URL, and page-flipping URL. However, it did not perform well on other forums, and for example, suffered from low effectiveness. This is primarily due to the absence of good and specific URL similarity functions in structure-driven crawler. Thus it did not find precise URL patterns. Therefore, it performed similarly to the behavior of a generic crawler and got good coverage on the forums except “afterdawn (1)”. This is because it does not find the pattern of page-flipping URL on this forum. From the results, we can conclude that a structure-driven crawler with small domain adaptation is not enough to be used as an effective forum crawler. We compare FoCUS with a more competent forum crawler, iRobot, and a generic crawler in the next section.

5.3.4 Online Crawling Comparison

In this section, we report the result of comparison between a generic crawler described in Section 5.3.1, iRobot (we re-implemented iRobot for our evaluations) and FoCUS. We let iRobot and FoCUS crawl each forum until no more pages could be retrieved. After that we count how many threads and other pages were crawled, respectively.

5.3.4.1 Crawling Effectiveness Comparison

Figure 8 shows the effectiveness comparison. FoCUS achieved almost 100% effectiveness on all forums. This confirms the effectiveness and robustness of FoCUS. The generic crawler is much less effective as we discussed in Section 5.3.1. As a comparison forum crawler, iRobot’s effectiveness was about 90%. It obtained high effectiveness on 6 out of 9 forums. But, it did not perform well on “xda-developers (3)”, “lkcn (7)”, and “techreport (8)”. After an error analysis, we found iRobot’s ineffectiveness was mainly due to that its random sampling strategy sampled many useless and noisy pages. For “xda-developers (3)” and “techreport (8)”, they have many redundant URLs, which made it hard to select the best traversal path. As to “lkcn (7)”, iRobot crawled many user profiles. Because this forum did not impose login control or robots.txt, many user profile pages were sampled and retained by iRobot’s informativeness estimation.

Comparing to iRobot, FoCUS learns the EIT path in forums and ITF regexes directly. Therefore, FoCUS was not affected by noisy pages and performed better. This indicates that given the fixed bandwidth and storage, FoCUS could crawl much more valuable content than iRobot.

5.3.4.2 Crawling Coverage Comparison

According to Figure 9, FoCUS had better coverage than iRobot. The average coverage of FoCUS was 99%, which is very close to a generic crawler, comparing to iRobot’s 86%. After an error analysis, we found that iRobot’s low coverage on “cqzg (4)” was due to the fact its forum pages had two layout structures. iRobot learned only one structure from sampled pages. This led to iRobot’s loss of many thread pages. For “crackberry (5)” and “Gentoo (6)”, iRobot suffered from changed page-flipping URL locations across pages. iRobot’s tree-like traversal path also deceased its coverage on “lkcn (7)” and “cqzg (4)”. In these 2 forums, some boards have many sub-boards. Recall that iRobot’s tree-like traversal path selection does not allow more than one path from an entry page node to a thread page node. Thus iRobot missed the thread URLs in these sub-boards.

In contrast, FoCUS crawled almost all the threads in forums since it learned EIT path and ITF regexes directly. In online crawling, FoCUS found all boards and threads whether they appeared in repetitive regions or not. The results on coverage also demonstrated our methods of index/thread URL and page-flipping URL detection are very effective.

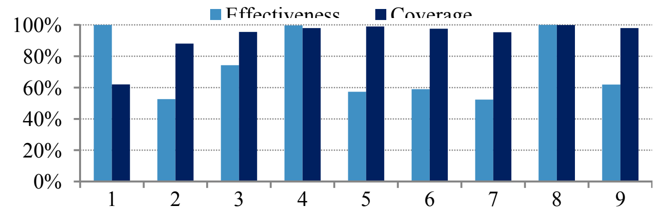


Figure 7. Effectiveness and coverage of structure-driven crawler

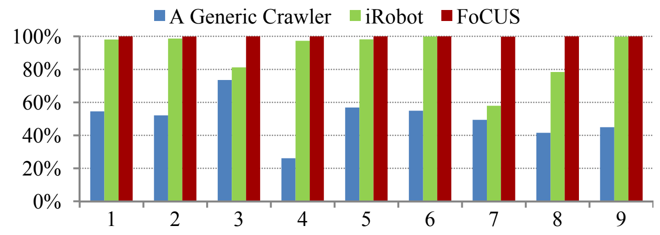


Figure 8. Effectiveness comparison between a generic crawler, iRobot and FoCUS

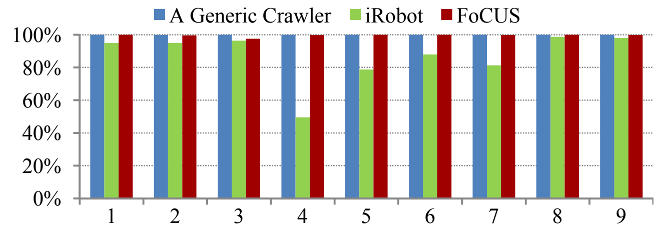


Figure 9. Coverage comparison between a generic crawler, iRobot and FoCUS

5.3.5 Large Scale Online Crawling

All previous works evaluated their methods on only a few forums. In this paper, to find out how FoCUS would perform in real online crawling, we evaluated FoCUS on 160 test forums which represents 160 different forum software packages. After learning ITF regexes, we found that FoCUS failed on 2 forums. One forum was no longer in operation and the other used JavaScript to generate index URLs. We tested FoCUS on the remaining 158 forums.

The effectiveness of 156 out of the 158 forums was greater than 98%. The effectiveness of the remaining 2 forums was about 91%. The coverage of 154 out of the 158 forums was greater than 97%. The coverage of the remaining 4 forums ranged from 4% to 58%. For these forums, FoCUS failed to find the page-flipping URLs since they either used JavaScript or they were too small.

The smallest forum in the 158 test forums had only 261 threads and the largest one had over 2 billion threads. To verify that FoCUS performs well across forums of different sizes, we calculated its micro-average and macro-average of effectiveness and coverage. As shown in Table 5, the micro-average and macro-average are both high and they are very close. This indicates that FoCUS performed well on small forums and large forums and is practical in web-scale crawling. To the best of our knowledge, it’s the first time such a large-scale test has been reported.

Table 5. Micro/Macro-average of effectiveness and coverage

	Effectiveness %	Coverage %
Micro-Average	99.96	99.16
Macro-Average	99.85	97.46

5.4 Apply to cQA Sites and Blog Sites

Though we proposed the concept of recognizing EIT path by learning ITF regexes to solve the forum crawling problem, the concept of EIT path and learning ITF regexes also applies to some other sites, such as community Question & Answer sites (cQA) and blog sites. In order to verify, we conducted a small experiment to test FoCUS on some cQA sites and blog sites. The numbers of questions or blogs in these sites vary from 93 to more than 1.6 million. The results show FoCUS achieved nearly 100% effectiveness and coverage on all sites. This demonstrated that our EIT path also applies to at least some cQA sites and blog sites and achieve the same performance as on forum sites.

6. CONCLUSION

In this paper, we proposed and implemented FoCUS, a supervised forum crawler. We reduced the forum crawling problem to a URL type recognition problem and showed how to leverage implicit navigation paths of forums, i.e. entry-index-thread (EIT) path, and designed methods to learn ITF regexes explicitly. Experimental results on 160 forum sites each powered by a different forum software package confirm that FoCUS could effectively learn knowledge of EIT path and ITF regexes from as few as 5 annotated forums. We also showed that FoCUS can effectively apply learned forum crawling knowledge on 160 unseen forums to automatically collect index URL, thread URL, and page-flipping URL string training sets and learn the ITF regexes from the training sets. These learned regexes could be applied directly in online crawling. Training and testing on the basis of forum package makes our experiments manageable and our results applicable to many forum sites. Moreover, FoCUS can start from any page of a forum, while all previous works expect an entry page is given. Our test results on 9 unseen forums show that FoCUS is indeed very effective and efficient and outperforms the state-of-the-art forum crawler, iRobot. The results on 160 forums show that FoCUS can apply the learned knowledge to a large set of unseen forums and still achieve a very good performance. Though, the method introduced in this paper is targeted at forum crawling, the implicit EIT-like path also apply to other sites, such as community Q&A sites, blog sites, and so on.

In the future, we would like to handle forums which use JavaScript, include incremental crawling, and discover new threads and refresh crawled threads in a timely manner. The initial results of applying FoCUS-like crawler to other social media are very promising. We would like to conduct more comprehensive experiments to further verify our approach and improve upon it.

7. ACKNOWLEDGMENT

This work was supported in part by Fundamental Research Funds for the Central Universities (No. WK2100230002), National Natural Science Foundation of China (No. 60933013), and National Science and Technology Major Project (No. 2010ZX03004-003).

8. REFERENCES

[1] ForumMatrix. http://www.forummatrix.org/index.php

[2] Forum Software Reviews. http://www.forum-software.org/forum-reviews

[3] Message Boards Statistics. http://www.big-boards.com/statistics/

[4] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the DUST: different URLs with similar text. In *Proc. of 16th WWW*, pages 111-120, 2007.

[5] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998.

[6] R. Cai, J.-M. Yang, W. Lai, Y. Wang, and L. Zhang. iRobot: An Intelligent Crawler for Web Forums. In *Proc. of 17th WWW*, pages 447-456, 2008.

[7] A. Dasgupta, R. Kumar, and A. Sasturkar. De-duping URLs via rewrite rules. In *Proc. of 14th KDD*, pages 186-194, 2008.

[8] C. Gao, L. Wang, C.-Y. Lin, and Y.-I. Song. Finding Question-Answer Pairs from Online Forums. In *Proc. of 31st SIGIR*, pages 467-474, 2008.

[9] N. Glance, M. Hurst, K. Nigam, M. Siegler, R. Stockton, and T. Tomokiyo. Deriving Marketing Intelligence from Online Discussion. In *Proc. 11th SIGKDD*, pages 419-428, 2005.

[10] Y. Guo, K. Li, K. Zhang, and G. Zhang. Board Forum Crawling: a Web Crawling Method for Web Forum. In *Proc. of 2006 IEEE/WIC/ACM WI*, pages 475-478, 2006.

[11] M. Henzinger. Finding near-duplicate Web pages: a large-scale evaluation of algorithms. In *Proc. of 29th SIGIR*, pages 284-291, 2006.

[12] H. S. Koppula, K.P. Leela, A. Agarwal, K.P. Chitrapura, S. Garg and A. Sasturkar. Learning URL Patterns for Webpage De-duplication. In *Proc. of 3rd WSDM*, pages 381-390, 2010.

[13] K. Li, X.Q. Cheng, Y. Guo, and K. Zhang. Crawling Dynamic Web Pages in WWW Forums. *Computer Engineering*, 33(6): 80-82, 2007.

[14] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for Web crawling. In *Proc. of 16th WWW*, pages 141-150, 2007.

[15] U. Schonfeld , N. Shivakumar. Sitemaps: above and beyond the crawl of duty. In *Proc. of the 18th WWW*, pages 991-1000, 2009.

[16] X.Y. Song, J. Liu, Y.B. Cao, and C.-Y. Lin. Automatic Extraction of Web Data Records Containing User-Generated Content. In *Proc. of 19th CIKM*, pages 39-48, 2010.

[17] V. N. Vapnik. The Nature of Statistical Learning Theory. Springer, 1995.

[18] M. L. A. Vidal, A. S. Silva, E. S. Moura, and J. M. B. Cavalcanti. Structure-driven Crawler Generation by Example. In *Proc. of 29th SIGIR*, pages 292-299, 2006.

[19] Y. Wang, J.-M. Yang, W. Lai, R. Cai, L. Zhang, and W.-Y. Ma. Exploring Traversal Strategy for Web Forum Crawling. In *Proc. of 31st SIGIR*, pages 459-466, 2008.

[20] J.-M. Yang, R. Cai, Y. Wang, J. Zhu, L. Zhang, and W.-Y. Ma. Incorporating Site-Level Knowledge to Extract Structured Data from Web Forums. In *Proc. of 18th WWW*, pages 181-190, 2009.

[21] Y. Zhai and B. Liu. Structured Data Extraction from the Web based on Partial Tree Alignment. IEEE Trans. Knowl. Data Eng., 18(12):1614−1628, 2006.

[22] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise Networks in Online Communities: Structure and Algorithms. In *Proc. of 16th WWW*, pages 221-230, 2007.