iRobot: An Intelligent Crawler for Web Forums

Rui Cai, Jiang-Ming Yang, Wei Lai, Yida Wang, and Lei Zhang
Microsoft Research, Asia
No. 49 Zhichun Road, Beijing, 100080, P.R. China
{ruicai, jmyang, weilai, v-yidwan, leizhang}@microsoft.com

ABSTRACT
We study in this paper the Web forum crawling problem, which is a very fundamental step in many Web applications, such as search engine and Web data mining. As a typical user-created content (UCC), Web forum has become an important resource on the Web due to its rich information contributed by millions of Internet users every day. However, Web forum crawling is not a trivial problem due to the in-depth link structures, the large amount of duplicate pages, as well as many invalid pages caused by login failure issues. In this paper, we propose and build a prototype of an intelligent forum crawler, iRobot, which has intelligence to understand the content and the structure of a forum site, and then decide how to choose traversal paths among different kinds of pages. To do this, we first randomly sample (download) a few pages from the target forum site, and introduce the page content layout as the characteristics to group those pre-sampled pages and re-construct the forum's sitemap. After that, we select an optimal crawling path which only traverses informative pages and skips invalid and duplicate ones. The extensive experimental results on several forums show the performance of our system in the following aspects: 1) Effectiveness – Compared to a generic crawler, iRobot significantly decreases the duplicate and invalid pages; 2) Efficiency – With a small cost of pre-sampling a few pages for learning the necessary knowledge, iRobot saves substantial network bandwidth and storage as it only fetches informative pages from a forum site; and 3) Long threads that are divided into multiple pages can be re-concatenated and archived as a whole thread, which is of great help for further indexing and data mining.

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 Crawler, Sitemap Construction, Traversal Path Selection.

1. INTRODUCTION
With the developing of Web 2.0, user-created content (UCC) now is becoming an important data resource on the Web. As a typical UCC, Web forum (also named bulletin or discussion board) is very popular almost all over the world for opening discussions. Every day, there are innumerable new posts created by millions of Internet users to talk about any conceivable topics and issues. Thus, forum data is actually a tremendous collection of human knowledge, and therefore is highly valuable for various Web applications. For example, commercial search engines such as Google, Yahoo!, and Baidu have begun to leverage information from forums to improve the quality of search results, especially for Q&A queries like "how to debug JavaScript". It is also noticed that some recent research efforts have tried to mine forum data to find out useful information such as business intelligence [15] and expertise [27]. However, whatever the application is, the fundamental step is to fetch data pages from various forum sites distributed on the whole Internet.

To download forum data effectively and efficiently, we should first understand the characteristics of most forum sites. In general, content of a forum is stored in a database. When a Web forum service receives a user request, it dynamically generates a response page based on some pre-defined templates. The whole forum site is usually connected as a very complex graph by many links among various pages. Due to these reasons, forum sites generally have the following common characteristics. First, duplicate pages (or content) with different Uniform Resource Locators (URLs) [2] will be generated by the service for different requests such as "view by date" or "view by title." Second, a long post divided into multiple pages usually results in a very deep navigation. Sometimes a user has to do tens of navigations if he/she wants to read the whole thread, and so does a crawler. Finally, due to privacy issue, some content such as user profiles is only available for registered users.

Generic Web crawlers [8], which adopt the breadth-first strategy [5], are usually inefficient in crawling Web forums. A Web crawler must make a tradeoff between "performance and cost" to balance the content quality and the costs of bandwidth and storage. A shallow (breadth-first) crawling cannot ensure to access all valuable content, whereas a deep (depth-first) crawling may fetch too many duplicate and invalid pages (usually caused by login failures). In the experiments we found out that using a breadth-first and depth-unlimited crawler, averagely there is more than 40% crawled forum pages are invalid or duplicate. Moreover, a generic crawler usually ignores the content relationships among pages and stores each page individually [8]; whereas a forum crawler should preserve such relationships to facilitate various data mining tasks. In brief, neither the breadth-first nor the depth-first strategy can simply satisfy the requirements of forum crawling. An ideal forum crawler should answer two questions: 1) what kind of pages should be crawled? Obviously, duplicate and invalid pages should be skipped to save network bandwidth and reduce redundancy; and 2) what kind of out links in a page should be followed, and how to follow these links? In nature, these two questions are coupled with each other. To verify whether a page is valuable, the crawler should find out where it comes from (i.e., following which links can reach this page); while to judge whether a link should be followed, the crawler must evaluate the informativeness of the target page.

Some related works have been reported in recent literature. Focused crawling, which attempts to download only Web pages that are relevant to some pre-defined topics, has been studied in [11], [20], and [25]. And in [18] and [21], the problem of deep Web

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 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

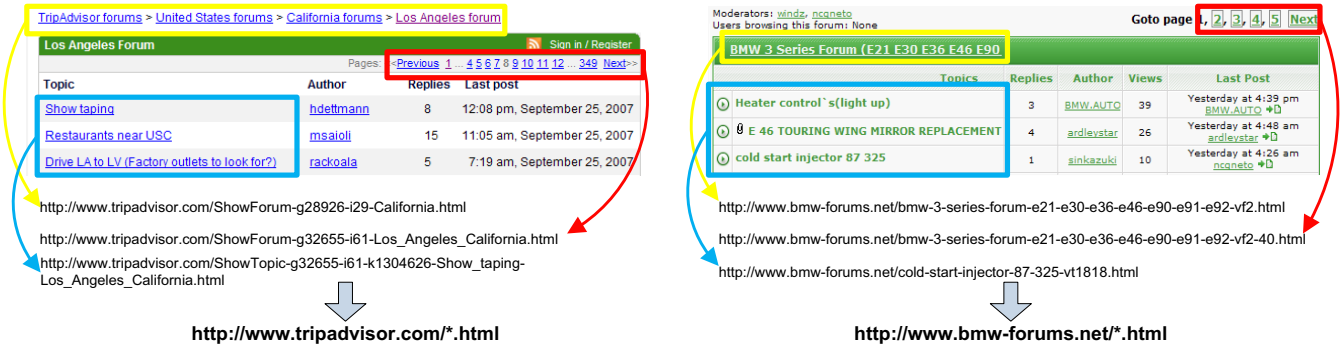

Figure 1. Two example Web forums with ambiguous URL formats. Links with the same functions are marked with color squares.

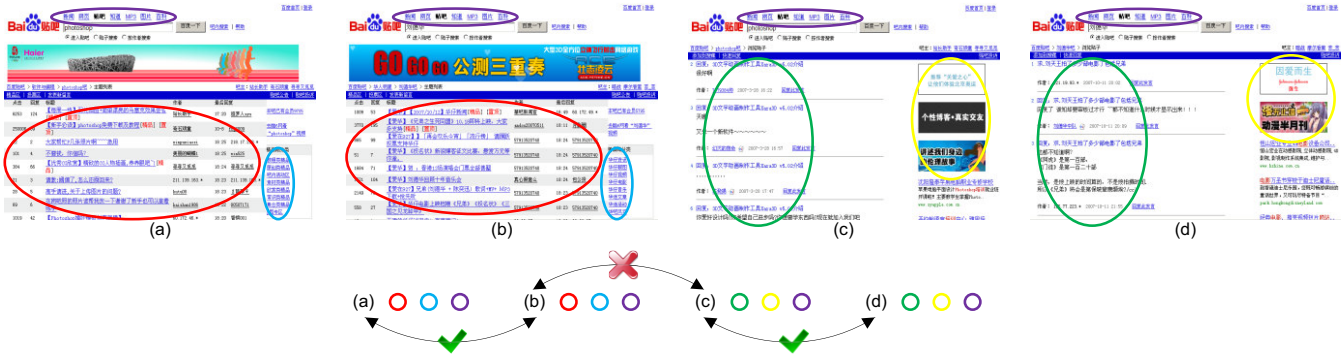

Figure 2. An example of salient repetitive regions (marked with ellipses in different colors) on four pages (a)−(d). Page (a) and (b) are list-of-thread; and (c) and (d) are post-of-thread. It is obvious that similar pages have similar repetitive regions while different pages contain different ones.

crawling is discussed, to automatically fetch hidden Web pages which are typically only accessible by submitting queries to a database. Another related topic is near-duplicate page detection [6][17][19], to remove redundant pages and reduce indexing overhead. And based on the ideas from [7], the Sitemaps Protocol [3] was recently proposed in industry to allow a webmaster to inform search engines about links that are available for crawling. We will provide a more detailed review in Section 2.

All these research studies may have partial answers to the aforementioned two questions in forum crawling. However, we argue that none of them can completely meet the requirements of forum crawling. For the first question, current research mostly focuses on fetching pages with particular semantic content, i.e., focused crawling; while topics in a forum are usually too diverse to be defined in a target list. There is also some work describing the target pages using Document Object Model (DOM) trees [25]; we argue that DOM trees are not robust enough in forum sites, as similar pages may have different numbers of advertisements, images, and even some complex sub-structure embedded in user posts. For the second question, most current works adopt URL patterns for navigation, i.e., first summarize the target links using a regular expression-like pattern and then follow any link matched with that pattern. However, automated URL pattern generation is very risky, as different forums may use multifarious strategies to formulate URLs [6]. Fig. 1 shows two example Web forums with very ambiguous URL formats. It's even hard for human being to understand how these links are formulated and write down appropriate patterns to distinguish links with different functions. Moreover, little work considers the problem of how to follow a link; while for forum crawling, it is usually desired to preserve the relationships among those pages from a same thread when following the corresponding links.

In this paper, after investigating a substantial number of Web forums, we found out two observations:

(1) **The repetitive regions in a forum page can robustly characterize the content layout of that page**. Forum pages are generated by pre-defined templates, and different templates are usually adopted to present different content such as *list-of-board*, *list-of-thread*, *post-of-thread*, *user profile*, etc. Moreover, valuable information in forum pages is mostly shown in some repetitive manners, as it is essentially data records stored in a database. In general, different templates contain different repetitive regions, as they are utilized to deliver different kinds of information. Fig. 2 illustrates such an example. In Fig. 2, there are four Web pages from http://post.baidu.com, the largest open discussion forum in China. Page (a) and (b) are pages of *list-of-thread*; while (c) and (d) are pages of *post-of-thread*. The salient repetitive regions in these pages are marked with colored ellipses in Fig. 2, and regions in a same color are with the same structures. From Fig. 2, it is clear that similar pages have similar repetitive regions which further have similar locations. Thus, a forum page can be well characterized by what kinds of repetitive regions it contains, and where these regions are located. In comparison with the whole DOM tree, the repetitive region-based representation is more robust. For example, the DOM tree structure of a *post-of-thread* page with only two posts is quite different with that of another page with twenty posts; while these two pages do have the same kind of repetitive region consisting of posts.

(2) **The location of a link on a page is important**. Such location information should be integrated with the URL patterns to decide which links should be followed and how to follow them. Let's revisit the examples in Fig. 1. Although it is hard to automatically infer appropriate patterns to distinguish links with different functions (such as page-flipping and link-to-thread) in these two pag-

es, the locations of these links are helpful. In general, links in a same repetitive region usually take on the same function, e.g., links of page-flipping are located repetitively on the top-right corner, marked with the red squares on these two pages. Thus, even for the worst case where the URL patterns cannot provide any useful information, we can still judge how to follow a link by verifying its location information.

Based on the above observations, in this paper we propose iRobot, an intelligent crawler for Web forums. The main idea of iRobot is to automatically rebuild the graphical architecture representation, i.e., the sitemap [23], of the target Web forum; and then select an optimal traversal path which only traverses informative pages and skip invalid and duplicate ones. Here, a sitemap is a directed graph consisting of a set of vertices and corresponding arcs, where each vertex represents a kind of pages in that forum and each arc denotes the link relation between two vertices. To reveal the sitemap, in iRobot, we first randomly crawl a few pages from the target site. Then, based on the first observation, all possible repetitive regions are discovered from these sampled pages, and are employed as features to group these pages into clusters according to their content layouts. Each cluster can be considered as a vertex in the sitemap. And based on the second observation, each arc in the sitemap is characterized by both the URL pattern and the location of related links, to provide more robust discrimination between links with different functions. In iRobot, we also propose an approach to automatically identifying vertices of informative pages from the sitemap, and then determine an optimal traversal path to across all the informative vertices with a minimum cost. Considering the possible risk of the automated traversing, in practice, iRobot also provides an interface for manual inspection, to improve the system performance with minimal human efforts.

iRobot has the following three advantages in comparison with a generic crawler: 1) *Effectiveness*. iRobot can intelligently skip most invalid and duplicate pages, while keep informative and unique ones. The experimental results show that iRobot can significantly reduce the ratios of invalidation and duplication; 2) *Efficiency*. First, iRobot only need a few pages to rebuild the sitemap. In the experiments, we found that 500 pages are usually enough to achieve an acceptable performance. Second, iRobot can save substantial network bandwidth and storage space as it only fetches informative pages in crawling; and 3) *Relationship-reserved Archiving*. When following links, iRobot can determine and record which pages of *list-of-thread* are from one board, and which pages of *post-of-thread* are from one thread. Thus, pages with such relationships can be consecutively achieved in repository, to facilitate further indexing and data mining tasks.

The rest of this paper is organized as follows. In Section 2, a brief review of related work is presented. In Section 3, we formally define some concepts used in our work. The overview of the proposed system is introduced in Section 4, and the module details are described in Section 5. Experiment results are reported in Section 6. And in the last section, we draw conclusions and point out future research directions.

2. RELATED WORK

To the best of our knowledge, little existing work in literatures has systemically investigated the problem of forum crawling. However, there are still some previous works that should be reviewed, as our approaches were motivated by them.

The most related work is focused crawling. Focused crawling was first proposed by Chakrabarti et al. [11] to attempt to only retrieve Web pages that are relevant to some pre-defined topics or labeled examples. The target descriptions in focused crawling are quite different in various applications. In the user-centric crawling [20], the targets are mined from user queries to guide the refreshing schedule of a generic search engine; while in [25] the target is described by the DOM tree of a manually selected sample page, and the crawling is limited to a specified Web site. Forum crawler is also a kind of targeted crawler as it selectively downloads informative pages having user-created content. However, the existing methods for target descriptions are not suitable for forum crawling. First, semantic topics in forums are too diverse to be simply characterized with a list of terms. The DOM tree-based method is also unreliable. DOM tree-based method adopts tree edit distance and a pre-defined threshold to determine whether two pages are similar [13][22]. However, DOM trees in forum sites are usually very complex, and it is difficult to adaptively select an appropriate threshold for every forum. Moreover, current focused crawlers still cannot handle the problems caused by invalid and duplicate pages in forum crawling.

Deep Web (or hidden Web) crawling [18][21] is also a related research topic. Content of deep Web is usually stored in databases, and is dynamically presented to users when requests come in. Most forum sites are a kind of deep Web as they also consist of dynamic pages generated based on database records. However, the focuses of a forum crawler and a deep Web crawler are different. A deep Web crawler focuses on how to prepare appropriate queries to retrieve hidden pages; while a forum crawler is interested in how to find out valuable links to follow given that most forum pages have explicit in-links.

Another related work is near-duplicate detection. The objective of near-duplicate detection is to remove redundant Web documents, to facilitate further indexing and archiving tasks. This is also one of the requirements of forum crawling. Most current work focuses on content-based duplicate detection [17][19], in which each Web document is first characterized with some fingerprints such as Shingles [9] or SimHash [19], and then any pair of documents with a small L_2 or hamming distance are considered as duplicates. However, content-based de-dup can only be carried out offline after the Web pages have been downloaded. It does benefit the following steps of indexing and archiving, but cannot help reduce the waste of bandwidth in crawling. There is also some recent work that discusses URL-based duplicate detection, which tries to mine rules of different URLs with similar text (DUST) [6]. However, such rules are very risky in practice, as URL formulations are too multifarious to generate robust rules. Furthermore, such method still needs to analyze logs from the target Web server or some previous crawling.

An industry-related work is the Sitemaps Protocol [3], which is actually an XML file that lists the URLs, as well as additional information such as update frequencies, for a Web site. However, it is hard to stably maintain such a protocol file for Web forums as their content continually changes. Moreover, the Sitemaps Protocol can only support no more than 50,000 URLs, which is inadequate for most forum sites.

There is also one recent study talking about the problem of forum crawling [16]. Unfortunately, it is based on heuristic rules and can only deals with forums with some specific organization structures. While in reality there are hundreds of forum structures implemented by either Website developers or different Internet forum software [1]. Thus it is difficult to define universal heuristics for general forum crawling.

3. PROBLEM DEFINITION

To make a clear presentation and facilitate the following discussions, we first define some concepts used in this paper.

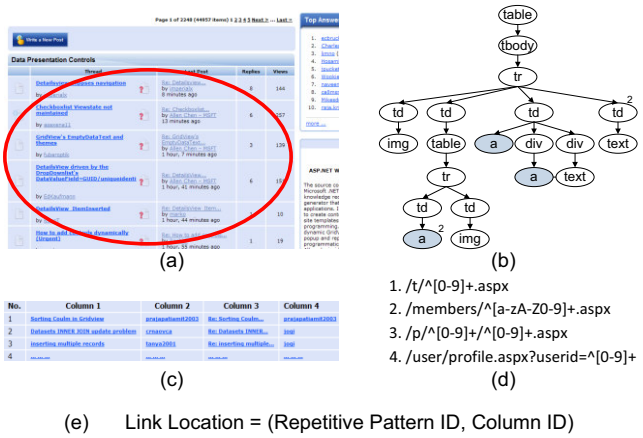

Figure 3. An illustration of the definitions of (a) Repetitive Region, (b) Repetitive Pattern, (c) Link Table, (d) URL Pattern, and (e) Link Location.

Repetitive Region. A repetitive region on a Web page is a block area containing multiple data records in a uniform formation. Fig.3 (a) gives an example of a repetitive region, marked with a red ellipse, on a *list-of-thread* page from the ASP.NET forum (http://forums.asp.net). This region consists of a list of thread records, and each record contains fields like title and author of each thread. As we have mentioned before, repetitive regions are very common on forum pages, and most valuable information such as posts, navigation bars, advertisements, etc. is shown in repetitive regions.

Repetitive Pattern. A repetitive pattern is an abstract representation of all the records in a repetitive region; and a repetitive region can be treated as an instance of a repetitive pattern on the related page. More specifically, in this paper, a repetitive pattern is described with a tree structure which is basically an extended DOM trees with regular expression-like sign for each node, following the idea in [28]. Fig. 3 (b) shows the repetitive pattern generated based on the repetitive region in Fig. 3 (a). In our system, every repetitive pattern discovered from a forum site is indexed with a unique ID.

Link Table. Data records in a repetitive region can be extracted by aligning the DOM tree with corresponding repetitive pattern, and stored in a table-like data structure [26][28]. In this paper, as we are just interested in links on a page, only link-related fields are retained to construct a link table for each repetitive region. Fig. 3 (c) illustrates the link table generated by the repetitive pattern in Fig. 3 (b), where link related nodes (<a>) are shown in shadow.

URL Pattern. A URL pattern is a regular expression string which provides a generalized representation to a group of URLs. As URLs are usually constructed in a hierarchical manner, following the generic specification in [2], the URL pattern is actually a concatenation of sub-strings where each sub-string is a regular expression for the corresponding hierarchical level. Fig. 3 (d) shows four URL patterns for links from the four columns in Fig. 3 (c). Here, the domain name is ignored as we only interested in internal links of the target site.

Link Location. Link location is defined to characterize where a link is located in a page. In this paper, we only consider links in link tables, as most valuable navigational links in forum pages are in repetitive manners. Since repetitive regions already have certain location information, in this paper the location of a link is described using the combination of its column ID in the corresponding link table, and the ID of the repetitive pattern which generates the link table, as shown in Fig. 3 (e).

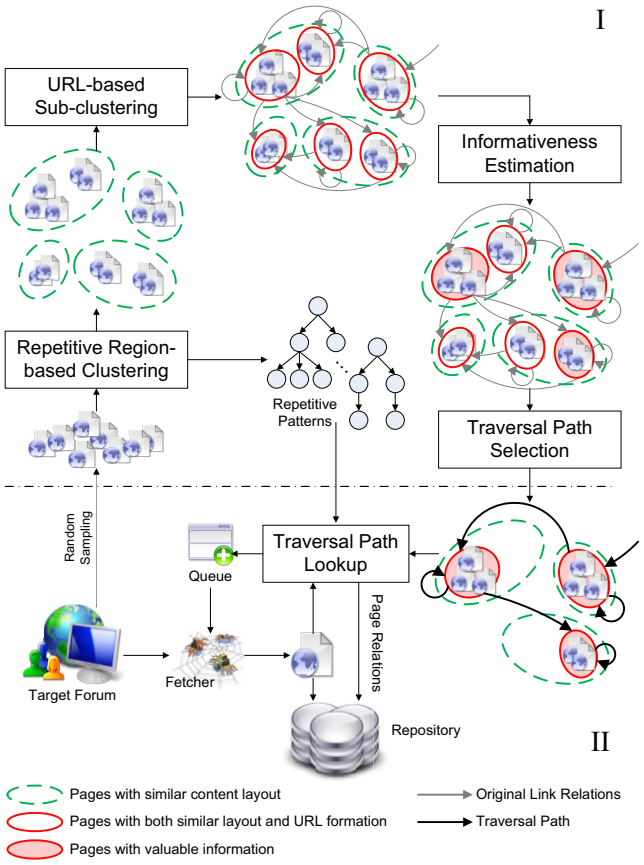

Figure 4. The flowchart of the proposed system, which consists of two parts: (I) offline sitemap reconstructing and traversal path selection; and (II) online crawling.

4. SYSTEM OVERVIEW

The flowchart of iRobot is illustrated in Fig. 4, which mainly consist of two parts: (I) offline sitemap reconstructing and traversal path selection; and (II) online crawling.

The goal of the offline part is to mine useful knowledge based on a few sampled pages, and guide the following massive crawling. The sampling quality is the foundation of the whole mining process. To keep the sampled pages as diverse as possible, in implementation, we adopt a double-ended queue and fetch URLs randomly from the front or end. In this way, the sampling process actually adopts a combined strategy of breadth-first and depth-first, and can retrieve pages at deep levels within a few steps. Then, through the *repetitive region-based clustering* module, pages with similar layout are grouped into clusters, as illustrated with green dash ellipses in Fig. 4, according to which kinds of repetitive patterns they hold. The byproduct of this module is a list of repetitive patterns occurring in pages from the target forum. After that, according to their URL formats, pages in each layout cluster are further grouped into subsets by the *URL-based sub-clustering* module. Thus, each subset contains pages with both uniform page layout and URL format, marked with red ellipses in Fig. 4, and is taken as a vertex in the sitemap graph. In this module, arcs among various vertices are also rebuilt, where each arc is characterized by both the URL pattern and link location of the corresponding links. The third module is *informativeness estimation*, which is in charge of selecting valuable vertices with infor-

mative pages on the sitemap, and throwing away useless vertices with invalid or duplicate pages. The valuable vertices are labeled with shadowed red ellipses in Fig. 4. The last module in the offline part is *traversal path selection*, whose function is to find out an optimal traversal path to traverse the selected vertices and step aside discarded ones. The selected paths are finally shown with dark arrows in Fig. 4; while original arcs are gray arrows. The details of these modules are introduced in Section 5.

Once the offline part is ready, the online crawling is carried out as follows. After a page is fetched, it is first sent to the *traversal path lookup* module. The other two inputs of the module are the traversal paths and repetitive patterns. In this module, the input page is aligned with all the repetitive patterns and classified into one of the vertices on the sitemap; and corresponding link tables are simultaneously constructed. Then, for each link in each link table, the module decides whether it should be added to the crawling queue, by looking up the list of traversal paths. Moreover, the module also outputs the relationships between the input page and those links to be followed, e.g., are they from the same thread? Such relationship information is then passed to the repository to judge whether these pages should be stored together. In practice, the *traversal path lookup* module is also responsible for failure detection. The module considers the mined knowledge as out-of-date if for a long time the input pages cannot be classified into any of the vertices on the sitemap. For example, this may be caused by template update in the forum site. Once this happens, the module will restart the offline flow to create new knowledge for the target.

5. MODULE DETAILS

In this section, we provide detailed introduction to those primary modules in iRobot, including: 1) repetitive region-based clustering, 2) URL-based sub-clustering, 3) informativeness estimation, and 4) traversal path selection. The operations in the module of traversal path lookup have overlaps with other modules, and will be discussed in related subsections.

5.1 Repetitive Region-based Clustering

This module tries to automatically group forum pages with a similar content layout (i.e. those may be generated by a same template) through measuring their distances in the feature space constructed by repetitive patterns. Thus, it needs to discover all possible repetitive patterns by investigating the sampled pages, and then for each page generates a description in the feature space. Actually, these two steps can be carried out simultaneously, as shown in Fig. 5.

The main procedure in Fig. 5 is to generate a pattern $\boldsymbol{p}^*$ for every repetitive region r in every page s, and investigate whether $\boldsymbol{p}^*$ can be matched with any existing pattern $\boldsymbol{p}^i$ in the list $\mathbb{P}$. If matched, $\boldsymbol{p}^i$ is updated with $\boldsymbol{p}^*$; otherwise $\boldsymbol{p}^*$ is added to $\mathbb{P}$ as a new pattern. Meanwhile, the feature description $\boldsymbol{f}$ is created for every s by recording n^i, the number of times $\boldsymbol{p}^i$ occurs in s, considering that in a page there may be several regions which are generated by the same pattern. The core algorithms behind this procedure are repetitive region detection and tree alignment, which are the foundations of the *RepetitiveRegionDetection*, *AlignRecordsInRegion*, and *TreeAlignmentCost* in Fig. 5. These algorithms have been well studied recently; for more details please refer to [26][28].

Here we would like to address some special considerations in our system. In practice, it was found that different patterns have different abilities in distinguishing forum pages. More specifically, it was observed that: 1) patterns are important if they have large rendering sizes on screen, as users always pay attention to salient blocks on a page and ignore small ones; and 2) popular patterns

Repetitive-Pattern-based-Feature-Extraction

input: a set of sampled Web pages $\mathbb{S}$ and a threshold ϵ
output: 1) a list of discovered repetitive patterns $\mathbb{P}$; and
2) a set of feature descriptions $\mathbb{F}$ for pages in $\mathbb{S}$
begin
  $\mathbb{P} = \phi$; $\mathbb{F} = \phi$;
  foreach $s \in \mathbb{S}$ **do**
    $\mathbb{R} = RepetitiveRegionDetection(s)$;
    foreach $r \in \mathbb{R}$ **do**
      create an empty $\boldsymbol{p}^* = \{p^*_{tree}, p^*_{area}, p^*_{support}\}$
      $p^*_{tree} = AlignRecordsInRegion(r)$;
      p^*_{area} = area of r / screen area; $p^*_{support} = 1$;
      foreach $\boldsymbol{p}^i \in \mathbb{P}$ **do**
        if $TreeAlignmentCost(p^*_{tree}, p^i_{tree}) < \epsilon$
          then $\boldsymbol{p}^i = Update(\boldsymbol{p}^i, \boldsymbol{p}^*)$ and **break;**
      end
      if cannot align $\boldsymbol{p}^*$ with any $\boldsymbol{p}^i \in \mathbb{P}$ **then** $\mathbb{P} \leftarrow \boldsymbol{p}^*$;
    end
    $\boldsymbol{f} = [n^1, n^2, \ldots, n^{\|\mathbb{P}\|}]$, n^i = number of $\boldsymbol{p}^i$ occurs in s;
    $\mathbb{F} \leftarrow \boldsymbol{f}$;
  end
end

Figure 5. The procedure used to discover repetitive patterns and generate feature descriptions for sampled pages.

are unimportant. Actually, patterns appearing on all the pages are removed before clustering. Thus, we added two parameters, p^i_{area} and $p^i_{support}$, to describe a pattern besides the tree structure p^i_{tree}. In implementation, the rendering information is from our previous technology of Vision-based Page Segmentation (VIPS) [10][24]. At last, p^i_{area} is the average area ratio of all the repetitive regions generated by p^i; and $p^i_{support}$ is the number of pages having such repetitive regions. All these p^i_{area} are re-normalized to sum to one, and are taken as weights of patterns in distance measure. Moreover, inspired by the well-known "term frequency–inverse document frequency (TF×IDF)" in text retrieval [4], the page feature $\boldsymbol{f}$ is further revised by integrating $p^i_{support}$, to punish those popular patterns, as:

$$f_i = (n^i / \sum_{j=1}^{\|\mathbb{P}\|} n^j) \times \log_{10}(\|\mathbb{S}\| / p^i_{support}) \quad (1)$$

The distance between two pages s_a and s_b is finally defined as:

$$dist(s_a, s_b) = \sqrt{\sum_{i=1}^{\|\mathbb{P}\|} p^i_{area} \times (f^i_a - f^i_b)^2} \quad (2)$$

As there is no prior knowledge about how many kinds of pages may exist in the target forum, in clustering, we just utilize the single linkage algorithm [12] to agglomerate these pages in a hierarchical way. The agglomeration is stopped when the minimum distance between pages of each cluster is larger than a pre-defined threshold.

In online crawling, these obtained clusters are also used to classify a new page in the traversal path lookup module. That is, the feature $\boldsymbol{f}$ is extracted for the new page and then compared with all the cluster centers one by one. If the minimum distance is less than the threshold used in clustering, the new page is classified into the corresponding cluster; otherwise the classification is failed and this page is discarded by the crawler.

5.2 URL-based Sub-clustering

Pages in a same layout cluster may still have different URL formats. This is usually caused by duplicate or invalid pages. Un-

doubtedly, duplicates are with different URLs but have almost the same semantic content and page layout. For invalid pages, these different URL addresses should originally point to various valid pages for logged users; however, these addresses are re-directed to the same login portal for a crawler as it is just a "guest" to that forum. Thus, to better rebuild the real topology structure of the target forum, in this module, each layout cluster are further split into subsets by grouping those pages with similar URL formats.

The foundation of the URL clustering is to measure the similarity between any two URL addresses. Fortunately, a URL is a relatively semi-structured string, which can be treated as a sequence of substrings separated by a forward slash "/". Moreover, considering that most forum URLs contain a query part to encode additional identification information, the last substring is again divided into two parts by a question mark "?"; and the part after the "?" is then decomposed into a series of <*key*, *value*> pairs according to the signs of equality "=" and ampersand "&" [2]. In this paper, the substrings before "?" are called *paths*; and those after "?" are *parameters*. Then, two URL strings are considered to be similar and are grouped together if they satisfy: 1) they have both the same number and the same order of *paths*; and 2) they have the same *parameters* of *keys*. It should be indicated that digital sequences are taken as a special kind of string, and are considered to be the same in the above two conditions. After the clustering, each cluster is finally represented by a URL pattern, which is essentially the concatenation of a sequence of regular expressions generated for every segment of *paths* and *parameters*.

To provide a more straightforward explanation, we list below two examples from the ASP.NET forum. They are both content layout clusters which can be further divided into multiple subsets with different URL patterns.

Cluster A is an example caused by duplicate pages. It is a group of pages which can be categorized into the following two URL patterns. However, given an ID, the two URLs will return a same *list-of-board* page in that forum.

- "/default.aspx?GroupID=^[0-9]+"
- "/default.aspx?ForumGroupID=^[0-9]+"

Cluster B is another example caused by invalid pages. It is a group of login pages returned by various links with the following patterns, which can only be accessed by registered users.

- "/user/ Profile.aspx?UserID=^[0-9]+"
- "/members/^[a-zA-Z0-9]+.aspx"
- "/AddPost.aspx?ReplyToPostID=^[0-9]+&Quote=False"

After the URL-based clustering, each obtained subset represents a kind of pages with the consistent content layout and URL format, and thus can be taken as a vertex on the sitemap. Then, the second responsibility of this module is to connect various vertices with arcs. Here, an arc is established if in the source vertex there is a page having a link pointing to another page in the target vertex. As aforementioned, each arc is also characterized by the location of the corresponding links besides the URL pattern. Therefore, two arcs will be created for two links with the same URL pattern but from different locations, which means different link tables or even different columns of a same link table. Accordingly, there may be more than one arc from one vertex to another.

5.3 Informativeness Estimation

As shown in Section 5.2, not all the vertices on a sitemap are valuable. Thus, an intelligent crawler must be able to decide by itself what kinds of pages (vertices) that are worth to download.

By examining a lot of forums, we found that most valuable pages (and page clusters) satisfy the following three assumptions:

1. A valuable page often belongs to a big "family" where there are many "brother" pages in a similar style (both the page layout and the URL pattern). In other words, pages in a large cluster are with a high probability to be valuable.
2. A valuable page is usually with a relatively large file size. As most informative pages in a forum are user-created content, their sizes are doubtless larger than those valueless pages which only show prompt information.
3. As a complement of the 2nd assumption, a more precise clue is the semantic diversity of the pages in a cluster. Basically, prompt pages such as the login portal are with fixed content; while pages of user created content are more diverse in semantics, although their layouts look alike.

Based on the three assumptions, in our system, a quantitative estimation of the informativeness of a vertex V_i is defined as:

$$Infor(V_i) = \frac{N_i}{N} \times \frac{S_i^{avg}}{S^{avg}} \times \left(1 - \frac{N_i^{dup}}{N_i}\right) \quad (3)$$

where N_i is the number of pages in V_i, and N is the total number of sampled pages; S_i^{avg} is the average page size in bytes of V_i, and S^{avg} is the average of all the sampled pages; and N_i^{dup} is the number of near-duplicates in V_i. The three components in (3) are designed for the three assumptions, respectively.

One thing should be addressed here is how to calculate the number of near-duplicates N_i^{dup} in (3). Following the ideas in [9][6], in our system, the semantic content of each sampled page is characterized with the histogram of its shingles. Here, a shingle is the hash value of any *n* consecutive words (*n*=5 in implementation) in a document. Then, two pages are considered to be near-duplicates if the L_2 distance of their histograms is small enough. However, computing such pair-wise distances is time consuming with a time complexity of $O(N^2)$. To accelerate it, in our system, we index all the histograms using the locality sensitive hashing (LSH) [14], which is designed to efficiently find nearest neighbors in a high-dimensional feature space. In this way, the process of near-duplicate detection can be finished in $O(N)$ time.

Although the cost of the near-duplicate detection is somewhat higher than the measurements of page numbers and average page size, in practice, it does improve the estimation accuracy significantly. For example, there are around 50% pages sampled from the ASP.NET forum are the portal of login. This leads to a high score if only using the first two components in (3). However, the score can be significantly punished by the third component based on the near-duplicate detection.

5.4 Traversal Path Selection

Traversing on a complex graph is not a trivial problem. Automated approaches usually take risks and thereby lower the crawling quality; while the most reliable decisions are still made by human beings. Actually, the re-constructed sitemap has provided a well presented mid-level result, based on which people only need to examine every arc and decide whether to follow, and how to follow it. However, this still requires considerable manual effort in practice since there are usually tens of arcs in a sitemap. Thus, in our system, we proposed a semi-automated method which only needs a few human inspections.

In the proposed method, the first step is to clean the sitemap by automatically removing most useless vertices and arcs, following the heuristics below, one by one:

1. Vertices with low quantity of information are dropped. For example, in the ASP.NET forum, it is not worth to label any arc from or to the vertices of the login portal. In implementation, we discard all the vertices whose $Infor(V_i)$ is less

than $tr = Infor_{avg} - Infor_{std}$, where $Infor_{avg}$ and $Infor_{std}$ are the mean and standard deviation of all the $Infor(V_i)$ on the sitemap, respectively.

2. For a layout cluster containing several vertices, reserving one representative one is enough, as the others are prone to be duplicates. For example, keeping one subset in the **Cluster A** in Section 5.2 is enough for retrieving all the *list-of-board* pages in the ASP.NET forum. In implementation, we just keep the largest vertex for each layout cluster.
3. For each vertex, arcs of self-linking are removed except those whose anchor text is a digital string or some particular strings such as "next". These remaining self-linking arcs are mostly links of page flipping, which should be followed with their relationships preserved for further indexing and archiving tasks, as shown in Fig. 4.

After that, the second step is to find out an optimal traversal path on the sitemap, which traverses all the survived vertices with minimum cost. Essentially, the problem is to find a directed spanning tree which is a sub-graph of the sitemap and contains all the vertices. Similar to the Prim's algorithm [12] in building minimum spanning tree, it's possible to construct such a directed spanning tree iteratively starting from the root vertex V_0 (the portal page of the forum) of the sitemap. That is, repeatedly add the arc with the minimum cost from all the arcs whose starting point is a vertex in the spanning tree but the ending point is not yet included, until all the vertices are included in the spanning tree. Since V_0 is the root of the sitemap, this process is guaranteed to include each vertex on the sitemap; since in each loop this process only includes a vertex not yet visited, it is guaranteed to end up with a tree. And the remaining problem is how to define the cost of every arc in the sitemap. To answer this question, we first summarize the human behavior in browsing a Web forum, which an ideal traversal path should refer to. That is, for pages at different levels, the one at shallow level is first visited; for pages at the same level, the one with more information is first browsed. Therefore, in our system, the cost of an arc is defined as the depth of its target vertex; and if two arcs point to vertices with the same depth, the one pointing to the vertex with the larger $Infor(V_i)$ is first selected. Here, the depth of a vertex is approximated by averaging the crawling levels of all the sampled pages belonging to it. Finally, the traversal path is the combination of the directed spanning tree and the arcs of self-linking selected by the 3rd heuristic above.

In this way, the traversal path is determined in a coarse-to-fine way, where each step can be easily inspected and revised by human. Such inspections need minor manual effort as the automated strategy can provide most suggestions for each step. For example, the above cleaning process dramatically reduces the numbers of vertices and arcs on the sitemap. In experiments we will show that the automated strategy can achieve very promising decisions for traversing, which just need a few manual corrections.

In our system, the traversal path is actually stored as a lookup table, where each line looks like:

$$IsFollow\left(id^{URL} \middle| id^{Vertex}, id^{Location}\right) = \theta,$$
$$where\ \theta = \begin{cases} 0 & don't\ follow \\ 1 & follow \\ 2 & follow\ \&\ keep\ relationship \end{cases} \quad (4)$$

In online crawling, a downloaded page is first classified into one of the vertex (id^{Vertex}) on the sitemap, as introduced in Section 5.2. Then, for an out link from that page, the traversal path lookup module further finds out its URL pattern (id^{URL}) and location ($id^{Location}$) on that page, and finally determine how to follow it by looking up this table.

Table 1. Web Forums in the Experiments

Web Forums	Description	URLs
http://www.biketo.com/bbs/	A bike forum (in Chinese)	17,965
http://forums.asp.net/	A commerce technical forum (in English)	16,116
http://post.baidu.com/	The largest Chinese community (in Chinese)	11,850
http://www.douban.com/	A community about book, music, and movie (in Chinese)	19,450
http://bbs.cqzg.cn/	A general forum (in Chinese)	19,809
http://www.tripadvisor.com/ForumHome	A message boards where travelers ask and answer questions (in English)	16,233
http://bbs.hoopchina.com/	A basketball forum (in Chinese)	18,996

6. EXPERIMENTS AND RESULTS

In this Section, we present the experimental results of the proposed system, including the characteristic analysis of our system, and some comparisons with a generic crawler on both effectiveness and efficiency.

6.1 Experiment Setup

To evaluate the performance of our system on various situations, seven different forums were selected in the experiments, as listed in Table.1. In these forums, "Biketo" and "CQZG" are powered by a popular forum tool, Discuz!, in China (but with different software versions); while the others are all self-developed forums with very diverse page layouts and site organizations. It should be noted that "Tripadvisor" is a travel tutorial site with an embedded forum, which contains only static links with ambiguous URL formats, as shown in Fig. 1.

To set up a consistent data collection for further evaluation and comparison, we first mirrored the above seven sites with a brute-force, breadth-first, and depth-unlimited crawler. For each site, the crawler starts from the homepage and follows any links belonging to that domain; and a unique URL address will be followed only once. For each forum we kept only the first 20,000 pages, which is enough to cover all kinds of pages in that forum. After removing error pages due to network problems, the number of unique URLs for each forum is listed in Table 1. The further crawling experiments, with either iRobot or the generic crawler, are all simulated on this data collection.

Moreover, for quantitative evaluation, we also labeled the ground truth for each page in the dataset. The ground truth includes whether a page is invalid, and whether it has duplicates in the corresponding forum. The invalid pages were manually selected for each forum. These pages are mainly with content like "login failed" and "content cannot be accessed". And to label duplicate pages, we built a tool for near-duplicate detection based on shingles and LSH, as introduced in Section 5.3. Here, it should be noticed that in the experiments we only investigate the duplication in valid pages. Invalid pages were not labeled as duplicates although they do have very similar content. At last, the pages beside duplicate and invalid pages are considered as valuable pages (for a set of duplicate pages, only one of them is considered as valuable and the others are treated as duplicate ones).

6.2 Characteristics of Our System

The characteristic analysis here includes: 1) the accuracy of the automated traversal path selection; 2) the crawling quality; 3) how many sampled pages is enough to support the system; and 4) the accuracy of the preserved page relationships.

First, as introduced in Section 4, for each site we randomly sampled 500 pages, based on which the corresponding sitemap was re-constructed and the traversal paths were generated, following the details in Section 5. We will show later that 500 pages are enough to provide a promising crawling quality. Fig. 6 illustrates such an example of the rebuilt sitemap and the selected traversal

Table 2. The comparison of the auto-generated and the manually selected traversal paths

Forums	Biketo	Asp	Baidu	Douban	CQZG	Tripadvisor	Hoopchina
auto-generated	18	9	48	40	34	123	18
manually-selected	17	6	26	37	25	100	17
intersection	15	5	23	33	24	78	17

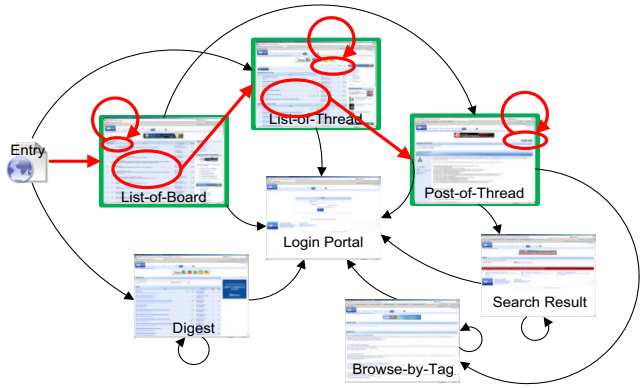

Figure 6. An example of the re-constructed sitemap and the selected traversal paths for the ASP.NET forum. This graph has been simplified for a clear view.

paths of the ASP.NET forum. In Fig. 6, the selected pages (vertices) are labeled with green squares, traversal paths are shown with red arcs, and the locations of the corresponding links are marked with red ellipses. It's clear that without a well selected path, a crawler may download lots of pages of the login portal, as almost every page in this forum has some links that require user login.

Moreover, we also manually checked the auto-generated traversal paths by removing those redundant arcs and adding those missed. The numbers of the auto-selected arcs, the manually selected arcs, and their intersections are listed in Table 2. From Table 2, it is noticeable that the number of intersection is very close to the number of manually selected arcs, which means most correct arcs have been covered by the auto-generated traversal paths. The average recall is above 85%. However, the precision is only around 70% which means that there are some redundant arcs in the auto-generated traversal paths. These redundant arcs are usually caused by useless vertices on the sitemap. The worst case is Tripadvisor, which is actually not a regular Web forum. There are lots of noise pages in this site and the generated sitemap is with too many vertices. It is hard for the system to decide which vertices are informative and thus most of them are reserved. Therefore, the informativeness estimation in Section 5.3 and the 1st heuristic in Section 5.4 are still need to be improved. However, as the human effort is mainly focused on removing useless vertices and arcs, such extra cost is still acceptable in practice.

After that, to evaluate the crawling quality of the propose system, we define three criteria: invalid ratio R_{inv}, duplicate ratio R_{dup}, and coverage ratio R_{cov}, as:

$$R_{inv} = \frac{N_{inv}}{N_{crawl}} \times 100\%; \quad R_{dup} = \frac{N_{dup}}{N_{crawl}} \times 100\%; \quad R_{cov} = \frac{N_{crawl} - N_{inv} - N_{dup}}{N_{val}} \times 100\% \tag{5}$$

where N_{inv} and N_{dup} are the numbers of invalids and duplicates in the total N_{crawl} pages retrieved by iRobot in the simulated crawling on a mirrored site, and N_{val} is the number of valuable pages in the ground truth of that site. Similarly, for each mirrored site, we can also calculate its invalid ratio and duplicate ratio according to the ground truth. Fig. 7 and Fig. 8 illustrated the comparison of iRobot and the mirrored sites, for invalid ratio and duplicate ratio, respectively. From Fig. 7, it's clear that iRobot can effectively

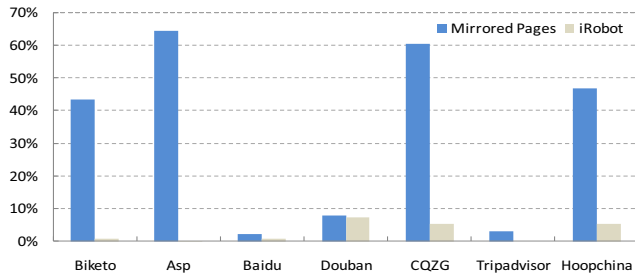

Figure 7. Invalid ratio comparison between the mirrored pages and the pages crawled by iRobot.

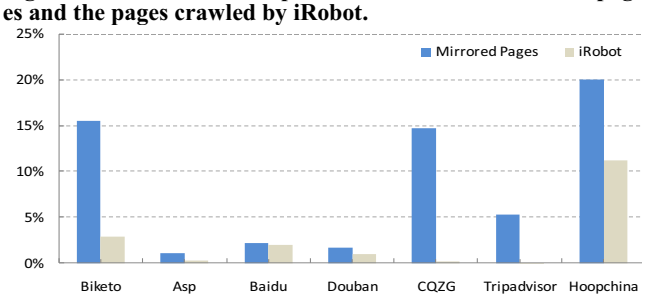

Figure 8. Duplicate ratio comparison between the mirrored pages and the pages crawled by iRobot.

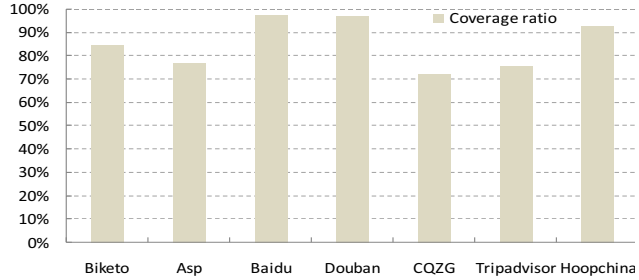

Figure 9. The coverage of valuable pages retrieved by iRobot.

reduce the invalid ratio in the original mirrors. Averagely, only 2.9% pages retrieved by iRobot are invalid; while for mirrored pages the number is 32.7%. And iRobot also works well in decreasing duplicates in crawling. As shown in Fig. 8, for iRobot, there are only around 2.5% pages that are duplicates on average; while about 8.7% are duplicates in the mirrored pages. Moreover, it is worth noting that iRobot can significantly decrease both invalid and duplicate ratios while keeping a high coverage ratio. As shown in Fig. 9, more than 85% valuable pages can be visited by iRobot.

From Fig. 7, 8, and 9, we can find that the invalid and duplicate ratios of various forums are quite different. Some forums are with both higher invalid and duplicate ratios, such as Biketo, CQZG, and Hoopchina. Most forums from non-commercial organizations are of this style. In contrast, forums supported by commercial companies are better designed and with less invalidation and duplication, such as Baidu and Douban. And forums with restricted access policies are often with higher invalid ratio, such as the ASP.NET forum. Fortunately, in general iRobot can achieve promising performance on all these forums. We also noticed that there are still some problems that should be improved. First, we should better balance the tradeoff between guaranteeing coverage and removing invalids. Take Biketo as an example, its coverage is somewhat harmed although the invalid ratio is significantly decreased. Second, we need more evidences to remove duplicates. In the current system, the duplicate detection is mainly based on the 2nd heuristic in Section 5.4, which may be too simple to handle some complex situations.

Table 3. The number of correctly preserved page relationships

Forums	Mirrored	Crawled by iRobot	Correctly Covered
Biketo	1584	1313	1293
Asp	600	536	536
Baidu	–	–	–
Douban	62	60	37
CQZG	1393	1384	1311
Tripadvisor	326	272	272
Hoopchina	2935	2829	2593

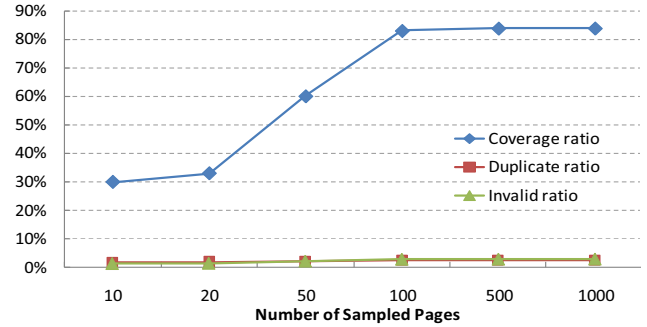

Figure 10. The crawling qualities using the sitemaps discovered based on various numbers of sampled pages.

The third part is to evaluate how many sampled pages are enough to provide an acceptable performance. To find an appropriate number, we built a series of sitemaps by varying the number of sampled pages. Then we repeatedly measured the quality for each sitemap with the three criteria in (5), and the average performance is shown in Fig. 10. In Fig. 10, it is noticed that when the number of sampled pages is larger than 100, the performance becomes stable. This indicates that iRobot only needs a small effort to learn the necessary knowledge from the target site to guide the following crawling.

At last, we evaluate how accurate the page relationships can be preserved, i.e., be followed with $\theta = 2$ in (4), by iRobot. To do this, we manually wrote rules for each forum to label all the pages with relationships (mostly are pages within the same boards or the same threads). This is taken as the ground truth and the numbers of correctly preserved page relationships are listed in Table 3. Here, Baidu is an exception because it is too large to find any page flipping in the top 20,000 mirrored pages. From Table 3, it is encouraging that in the crawled pages, there are around 95% of such relationships that are correctly preserved by iRobot. However, as there are around 15% valuable pages missed in the crawling, the corresponding relationships may be broken and thus cannot be discovered by the system.

6.3 Comparisons with a Generic Crawler

Effectiveness and efficiency are two important criteria to evaluate a Web crawler. Effectiveness means given a number of retrieved pages, how many of them are valuable and informative. Effectiveness is very important for saving network bandwidth and storage space. Efficiency means how fast a crawler can retrieve a given number of valuable pages. Efficiency is important as it determines how quickly a crawler can update its repository and indexing. In this subsection, we compare iRobot with a standard generic crawler, on both effectiveness and efficiency.

To evaluate the effectiveness, we retrieved 5,000 pages from the mirrored sites using the generic crawler and iRobot, respectively. According to the ground truth of invalids and duplicates, we can find out how many valuable pages were visited by the two crawlers, as shown in Fig. 11 (a) and (b). In the top 5,000 pages crawled by the generic crawler, the ratio of valuable pages is around 69%, averaged across the seven forums. In comparison,

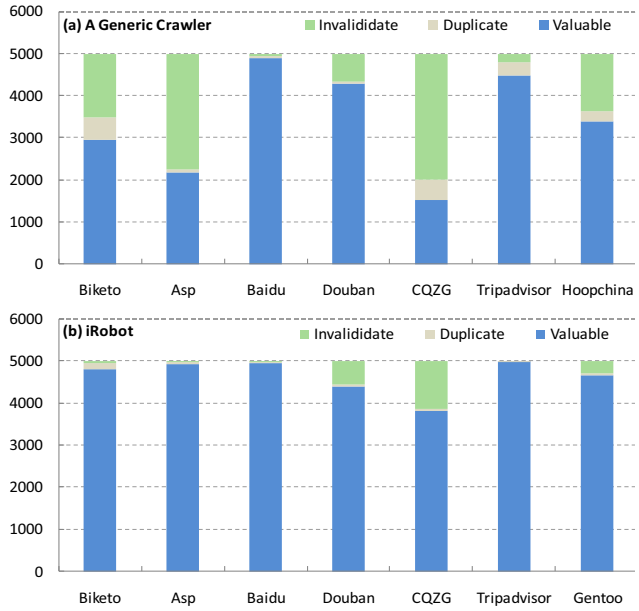

Figure 11. The comparison of effectiveness between a generic crawler (a) and iRobot (b), on the top 5,000 retrieved pages.

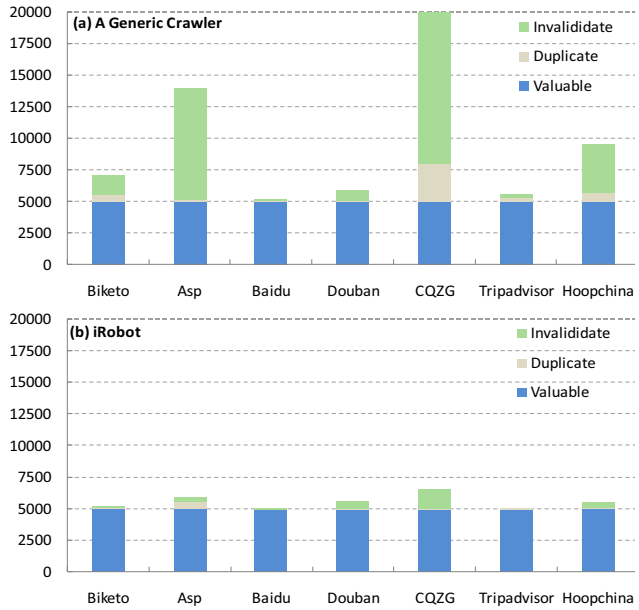

Figure 12. The comparison of efficiency between a generic crawler (a) and iRobot (b), to retrieve 5,000 valuable pages.

the average ratio of valuable pages crawled by iRobot is 93%, which is a significant improvement compared with the generic crawler. In other words, given a fixed bandwidth, iRobot can crawl almost 1.35 times valuable pages than a generic crawler.

To evaluate the efficiency, we continually crawled each mirrored site until 5,000 valuable pages are retrieved. Then we investigate how many pages have to be downloaded respectively using the generic crawler and iRobot. The results are shown in Fig. 12 (a) and (b). From Fig. 12, it can be found that to archive 5,000 valuable pages, a generic crawler averagely needs to crawl around 9600 pages; while in contrast iRobot needs to fetch only about 5500 pages. Thus, supposing a constant downloading time for each page, a generic crawler will require 1.73 times crawling time than iRobot to archive the same number of valuable pages.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented an intelligent crawler called iRobot for Web forum crawling. The main idea of iRobot is to first learn the sitemap of a forum site with a few pre-sampled pages, and then decide how to select an optimal traversal path to avoid duplicates and invalids. First, to discover the sitemap, those pre-sampled pages are grouped into multiple clusters according to their content layout and URL formats. In this part, we proposed a repetitive region-based layout clustering algorithm, which has been proven to be robust in characterizing forum pages. Then, the informativeness of each cluster is automatically estimated and an optimal traversal path is selected to traverse all the informative pages with a minimum cost. The major contribution in this step is to describe the traversal paths with not only their URL patterns but also their locations of the corresponding links. In such a way, we can provide a more strict discrimination between links with similar URL formats but different functions. Moreover, iRobot can also integrate manual inspections, to improve the crawling performance with minimal human efforts.

Experimental evaluations on various forums show very promising results. Compared with a generic crawler, iRobot can significantly reduce duplicate and invalid pages, without losing the coverage of valuable ones. As a small cost, iRobot only need to pre-sample no more than 500 pages for discovering necessary knowledge. This is very economic in comparison to the saved bandwidth and storage in the crawling stage. At last, iRobot can keep around 95% page relations in crawling, which is very useful for further indexing and data mining tasks.

We have investigated in this paper how to effectively and efficiently crawl Web forums. However, in a practical system, there are still two important components that should be further studied, that is, refresh schedule and archive structure. First, the refresh schedule in forum crawling is different from that in general Web crawling. In Web forums, different pages often have different fresh frequencies, and page content is usually changed in an incremental way. Second, how to design a repository for forum archiving is still an open problem. As in forum pages there are many objects like user, post, and date, an ideal repository should well organize such information for further access. Although with iRobot we already can re-construct a post thread divided into multiple pages, it is still not enough for object-level storage. In the future, we will study these two problems, besides continually improving the performance of iRobot.

8. REFERENCES

[1] Internet Forum Software. http://en.wikipedia.org/wiki/category:internet_forum_software.

[2] RFC 1738 − Uniform Resource Locators (URL). http://www.faqs.org/rfcs/rfc1738.html.

[3] The Sitemaps Protocol. http://sitemaps.org/protocol.php.

[4] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[5] R. A. Baeza-Yates, C. Castillo, M. Marin, and A. Rodriguez. Crawling a country: better strategies than breadth-first for Web page ordering. In *Proc. 14th WWW*, pages 864−872, Chiba, Japan, May 2005.

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

[7] O. Brandman, Junghoo Cho, H. Garcia-Molina, and N. Shivakumar. Crawler-friendly Web servers. *ACM SIGMETRICS Performance Evaluation Review*, 28(2):9−14, Sept. 2000.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1-7):107−117, 1998.

[9] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proc. 6th WWW*, pages 1157–1166, Santa Clara, California, USA, Apr. 1997.

[10] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. VIPS: a vision based page segmentation algorithm. *Microsoft Technical Report*, MSR-TR-2003-79, 2003.

[11] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11-16):1623−1640, 1999.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Sept. 2001.

[13] V. Crescenzia, P. Merialdoa, and P. Missier. Clustering Web pages based on their structure. *Data & Knowledge Engineering*, 54(3):279−299, Sept. 2005.

[14] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th SCG*, pages 253–262, NY, USA, Jun. 2004.

[15] N. Glance, M. Hurst, K. Nigam, M. Siegler, R. Stockton, and T. Tomokiyo. Deriving marketing intelligence from online discussion. In *Proc. 11th KDD*, pages 419−428, Aug. 2005.

[16] Y. Guo, K. Li, K. Zhang, and G. Zhang. Board forum crawling: a Web crawling method for Web forum. In *Proc. 2006 IEEE/WIC/ACM Int. Conf. Web Intelligence*, pages 745−748, Hong Kong, Dec. 2006.

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

[18] J. P. Lage, A. S. da Silva, P. B. Golgher, and A. H. F. Laender. Automatic generation of agents for collecting hidden Web pages for data extraction. *Data & Knowledge Engineering*, 49(2):177−196, May 2004.

[19] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for Web crawling. In *Proc. 16th WWW*, pages 141−150, Banff, Alberta, Canada, May 2007.

[20] S. Pandey and C. Olston. User-centric Web crawling. In *Proc. 14th WWW*, pages 401−411, Chiba, Japan, May 2005.

[21] S. Raghavan and H. Garcia-Molina. Crawling the hidden Web. In *Proc. 27th VLDB*, pages 129−138, Sept. 2001.

[22] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic Web news extraction using tree edit distance. In *Proc. 13th WWW*, pages 502−511, NY, USA, May 2004.

[23] L. Rosenfeld and P. Morville. *Information Architecture for the World Wide Web*. O'Reilly, Feb. 1998.

[24] R. Song, H. Liu, J.-R. Wen, W.-Y. Ma. Learning important models for Web page blocks based on layout and content analysis. *ACM SIGKDD Explorations Newsletter*, 6(2):14−23, Dec. 2004.

[25] M. L. A. Vidal, A. S. Silva, E. S. Moura, and J. M. B. Cavalcanti. Structure-driven crawler generation by example. In *Proc. 29th SIGIR*, pages 292−299, Seattle, USA, Aug. 2006.

[26] 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, Dec. 2006.

[27] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *Proc. 16th WWW*, pages 221−230, Banff, Canada, May 2007.

[28] S. Zheng, R. Song, J.-R. Wen, and D. Wu. Joint optimization of wrapper generation and template detection. In *Proc. 13th KDD*, pages 894−902, San Jose, CA, USA, Aug. 2007.