Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2009, Article ID 856037, 13 pages
doi:10.1155/2009/856037

Research Article

Profile-Based Focused Crawling for Social Media-Sharing Websites

Zhiyong Zhang and Olfa Nasraoui

Department of Computer Engineering and Computer Sciences, University of Louisville, Louisville, KY 40292, USA

Correspondence should be addressed to Olfa Nasraoui, olfa.nasraoui@louisville.edu

Received 31 May 2008; Accepted 6 January 2009

Recommended by Timothy Shih

We present a novel profile-based focused crawling system for dealing with the increasingly popular social media-sharing websites. In this system, we treat the user profiles as ranking criteria for guiding the crawling process. Furthermore, we divide a user's profile into two parts, an *internal part*, which comes from the user's own contribution, and an *external part*, which comes from the user's social contacts. In order to expand the crawling topic, a cotagging topic-discovery scheme was adopted for social media-sharing websites. In order to efficiently and effectively extract data for the focused crawling, a *path string*-based page classification method is first developed for identifying *list pages, detail pages*, and *profile pages*. The identification of the correct type of page is essential for our crawling, since we want to distinguish between list, profile, and detail pages in order to extract the correct information from each type of page, and subsequently estimate a reasonable ranking for each link that is encountered while crawling. Our experiments prove the robustness of our profile-based focused crawler, as well as a significant improvement in harvest ratio, compared to breadth-first and online page importance computation (OPIC) crawlers, when crawling the Flickr website for two different topics.

1. Introduction

Social media-sharing websites such as Flickr and YouTube are becoming more and more popular. These websites not only allow users to upload, maintain, and annotate media objects, such as images and videos, but also allow them to socialize with other people through contacts, groups, subscriptions, and so forth. Two types of information are generated in this process. The first type of information is the rich text, tags and multimedia data uploaded and shared on such web sites. The second type of information is the users' profile information, that can tell us what kind of interests they have. Research on how to use the first type of information has gained momentum recently. However, little attention has been paid to effectively exploit the second type of information, which are the user profiles, in order to enhance focused search on social media websites.

Prior to the social media boom, the concepts of *vertical search engines* and *focused crawling* have gradually gained popularity against popularity-based, general search engines. Compared with general search engines, topical or vertical search engines are more likely to become experts in specific topic areas, since they only focus on these areas. Although they lack the broadness that general search engines have, their depth can win them a stand in the competition.

In this paper, we explore the applicability of developing a focused crawler on social multimedia websites for an enhanced search experience. More specifically, we exploit the users' profile information from social media-sharing websites to develop a more accurate focused crawler that is expected to enhance the accuracy of multimedia search. To begin the focused crawling process, we first need to accurately identify the *correct type* of a page. To this end, we propose to use a *Document Object Model (DOM) path string*-based method for page classification. The correct identification of the right type of page not only improves the crawling efficiency by skipping undesirable types of pages, but also helps to improve the accuracy of the data extraction from these pages. In other words, the identification of the correct type of page is *essential* for our crawling, since we want to distinguish between list, profile, and detail pages in order to extract the right information, and subsequently estimate a reasonable ranking for each link that is encountered. In addition, we use a cotagging method for topic discovery as we think that it suits multimedia crawling more than the traditional taxonomy methods do, because it can help to discover some hidden and dynamic tag relations that may not be encoded in a rigid taxonomy (e.g., "tree" and "sky" may be related in many sets of scenery photos).

This paper is organized as follows. In Section 2, we review the related work in this area. In Section 3, we define the three types of pages that prevail on most social media-sharing websites, and discuss our focused crawling motivation. Then in Section 4, we explain the *path string*-based method for page classification. In Section 5, we introduce our profile-based focused crawling method. In Section 6, we discuss the cotagging topic discovery for the focused crawler. In Section 7, we wrap the previous sections to present our complete focused crawling system. In Section 8, we present our experimental results. Finally in Section 9, we make our conclusions and discuss future work.

2. Related Work

Focused crawlers were introduced in [1], in which three components, a classifier, a distiller, and a crawler, were combined to achieve focused crawling. A Bayes rule-based classifier was used in [2], which was based on both text and hyperlinks. The distillation process involves link analysis similar to *hub* and *authority* extraction-based methods. Menczer et al. [3] presented a comparison of different crawling strategies such as breadth-fist, best-first, PageRank, and shark-search. Pant and Srinivasan [4] presented a comparison of different classification schemes used in focused crawling and concluded that Naive Bayes was a weak choice when compared with support vector machines or neural networks.

In [5, 6], Aggarwal et al. presented a probabilistic model for focused crawling based on the combination of several learning methods. These learning methods include content-based learning, URL token-based learning, link-based learning, and sibling-based learning. Their assumption was that pages which share similar topics tend to link to each other. On the other hand, the work by Diligenti et al. [7] and by Hsu and Wu [8] explored using context graphs for building a focused crawling system. The two-layer context graph and Bayes rule-based probabilistic models were used in both systems.

Instead of using the page content or link context, another work by Vidal et al. [9] explored the page structure for focused crawling. This *structure-driven* method shares the same motivation with our work in trying to explore specific page-layouts or structure. In their work, each page was traversed twice: the first pass for generating the navigation pattern, and the second pass for actual crawling. In addition, some works [10, 11] for focused crawling used metasearch methods, that is, their method is based on taking advantage of current search engines. Among these two works, Zhuang et al. [10] used search engine results to locate the home pages of an author and then used a focused crawler to acquire missing documents of the author. Qin et al. [11] used the search results of several search engines to diversify the crawling seeds. It is obvious that the accuracy of the last two systems is limited by that of the seeding search engines. In [12], the authors used *cash* and *credit history* to simulate the page importance and implemented an Online Page Importance Computation (OPIC) strategy based on web pages' linking structure (*cash flow*).

Extracting tags from social media-sharing websites can be considered as extracting data from structured or semistructured websites. Research about extracting data from structured websites include RoadRunner [13, 14], which takes one HTML page as the initial wrapper, and uses *Union-Free Regular Expression* (UFRE) method to generalize the wrapper under mismatch. The authors in [15] developed the EXALG extracting system, which is mainly based on extracting *Large and Frequently occurring EQuivalence classes* (LFEQs) and differentiating roles of tokens using *dtokens* to deduce the template and extract data values. Later in [16], a tree similarity matching method was proposed to extract web data, where a tree edit distance method and a *Partial Tree Alignment* mechanism were used for aligning tags in the HTML tag tree. Research in extracting web record data has widely used a web page's structure [17] and a web page's visual perception patterns. In [18], several filter rules were proposed to extract content based on a DOM-tree. A human interaction interface was developed through which users were able to customize which type of DOM-nodes are to be filtered. While their target was for general HTML content and not for web records, they did not suit their methods to structured data record extraction. Zhao et al. [19] proposed using the tag tree structure and visual perception pattern to extract data from search engine results. They used several heuristics to model the visual display pattern that a search engine results page would usually look like, and combined this with the tag path. Compared with their tag path method, our *path string* approach keeps track of all the parent-child relationships of the DOM nodes in addition to keeping the parent-first-child-next-sibling pattern originally used in the DOM tree. We also include the node property in the *path string* generation process.

3. Motivation for Profile-Based Focused Crawling

3.1. Popularity of Member Profile. In Section 2, we reviewed several focused crawling systems. These focused-crawling systems analyze the probability of getting pages that are in their crawling topics based on these pages' parent pages or sibling pages. In recent years, another kind of information, which is the members' *profiles*, started playing a prominent role in social networking and resource-sharing sites. Unfortunately, this valuable information still eludes all current

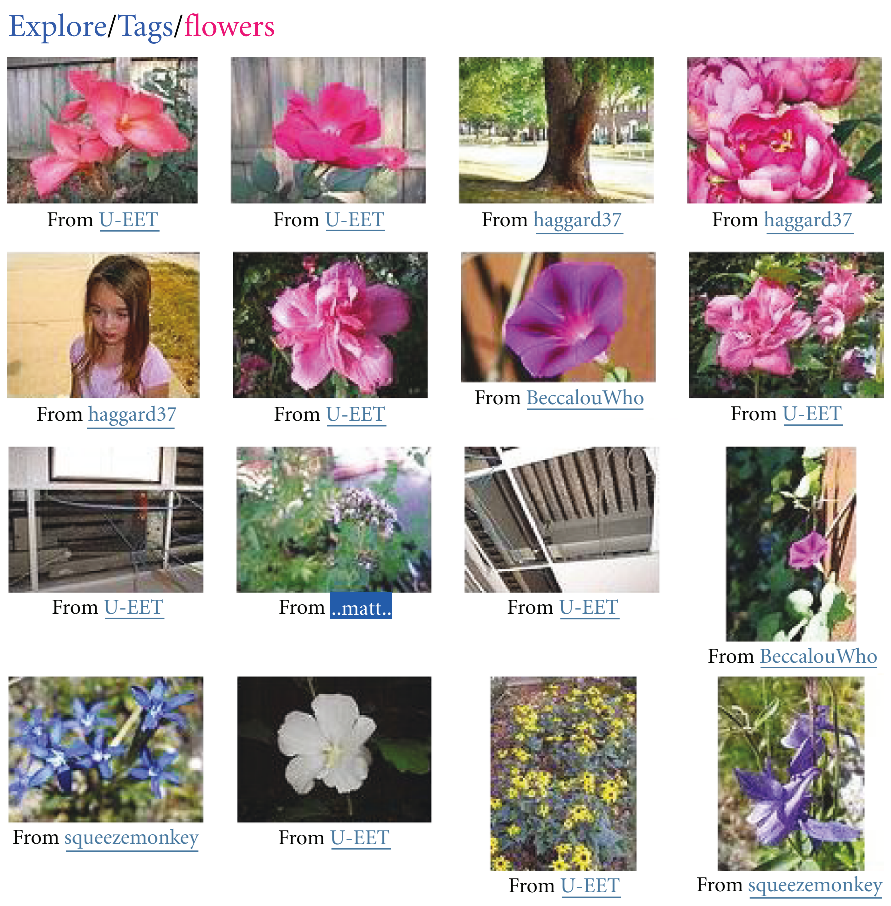


Figure 1: An example list page on Flickr.

focused crawling efforts. We will explore the applicability of using such information in our focused-crawling system. More specifically, to illustrate our profile-based focused crawling, we will use Flickr as an example. But our method can be easily expanded to other social networking sites, photo-sharing sites, or video-sharing sites. Hence, we refer to them as "social multimedia websites."

3.2. Typical Structure of Social Media-Sharing Websites. Social media-sharing Websites, such as Flickr and YouTube, are becoming more and more popular. Their typical organization structure are through the different *types* of web pages defined in what follows.

(1) A *list page* is a page with many image/video thumbnails and their corresponding uploaders (optionally some short descriptions) displayed below each image/video. A *list page* can be considered as a *crawling hub page*, from where we start our crawling. An example *list page* is shown in Figure 1.

(2) A *detail page* is a page with only one major image/video and a list of detailed description text such as title, uploader, and tags around it. A *detail page* can be considered as a *crawling target page*, which is our final crawling destination. An example *detail page* is shown in Figure 2.

(3) A *profile page* is a page that describes a media uploader's information. Typical information contained in such a page includes the uploader's image/video sets, tags, groups, and contacts, and so forth. Further, such information can be divided into two categories: *inner properties*, which describe the uploader's own contributions, such as the uploader's photo tags, sets, collections, and videos, and *inter properties*, which describe the uploader's networking with other uploaders, such as the uploader's friends, contacts, groups, and subscribers. We will use information extracted from *profile pages* to guide our focused crawling process.

A *list page* has many outlinks that point to *detail pages* and *profile pages*. Its structure is shown in Figure 3, in which two image thumbnails in a *list page* link to two *detail pages* and corresponding *profile pages*.

3.3. Profile-Based Focused Crawling. Our motivation while crawling is to be able to assess the importance of each outlink or *detail page link* before we actually retrieve that *detail page* given a *list page* and a crawling topic. For the case of Figure 3, suppose that we are going to crawl for the topic *flowers*, then we would intuitively rank the first *detail page link*, which links to a real flower, higher than the second *detail page link*, which

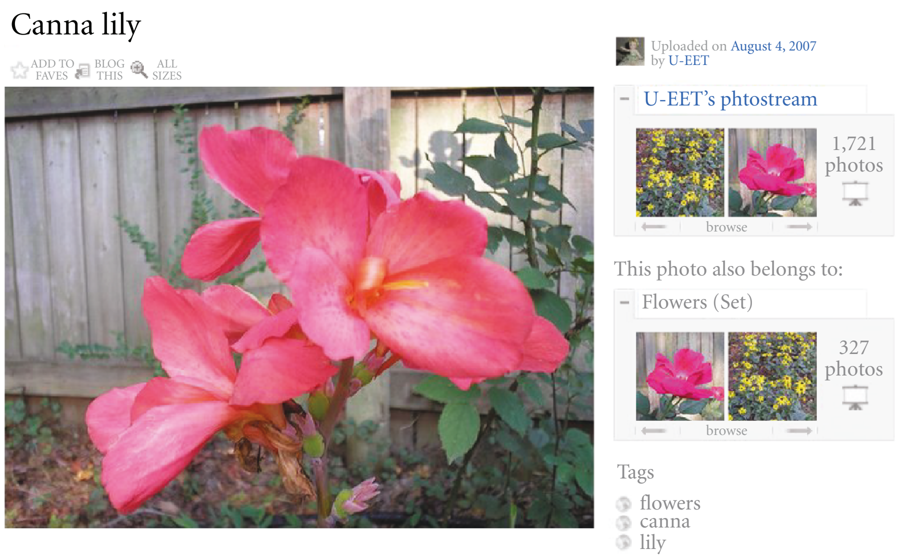


Figure 2: An example detail page on Flickr.

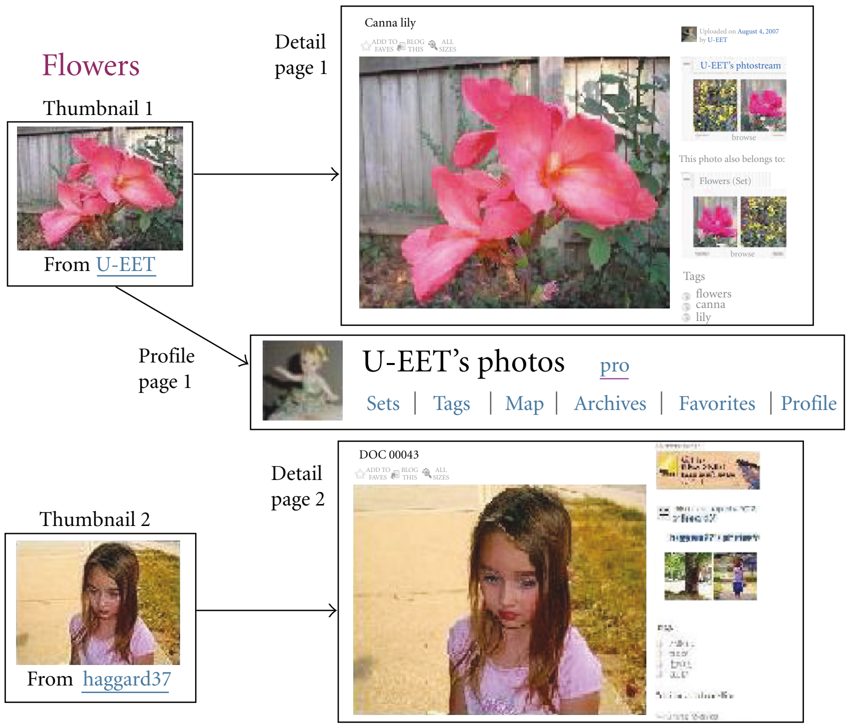


Figure 3: Typical structure of list, detail, and profile pages.

links to a walking girl and happened to be also tagged as "flower." The only information available for us to use is the photo thumbnails and the photo uploaders such as "U-EET" and "haggard37." Processing the *content* photo thumbnails to recognize which one is more conceptually related to the concept of real *flowers* poses a challenging task. Hence, we will explore the photo uploader information to differentiate between different concepts. Luckily, most social media-sharing websites keep track of each member's profile. As shown in Figure 3, a member's profile contains the member's collections, sets, tags, archives, and so forth. If we process all this information first, we can have a preliminary estimate of which type of photos the member would mainly upload and maintain. We can then selectively follow the *detail page links* based on the corresponding uploader profiles extracted.

4. Path String-Based Page Classification

Before we actually do the crawling, we need to identify the *type* of a page. In this section, we will discuss our page classification strategy based on the DOM *path string* method. Using this method, we are able to identify whether a page is a *list page, detail page, profile page*, or none of the above.

4.1. DOM Tree Path String. The *DOM* defines a hierarchy of *node* objects. Among the different types of nodes, *element node* and *text node* are the ones that are most relevant to our crawling. Figure 4 gives a simple example web page and its *DOM* tree representation.

In Figure 4, the whole DOM tree can be seen as a *document node*, whose child is the *element node <html>*, which further has two children *<head>* and *<body>*, both *element nodes*, and so on. The *element nodes* are all marked with <> in Figure 4. At the bottom of the tree, there are a couple of *text nodes*. In the DOM structure model, the *text nodes* are not allowed to have children, so they are always the leaf nodes of the DOM tree. There are other types of nodes such as *CDATASection nodes* and *comment nodes* that can be leaf nodes. *Element nodes* can also be leaf nodes. *Element nodes* may have properties. For example, "<tr class="people">" is an *Element Node* "<tr>" with property "class="people"." Readers may refer to http://www.w3.org/DOM/ for a more detailed specification.

A *path string* of a node is the string concatenation from the node's immediate parent all the way to the tree root. If a node in the path has properties, then all the *display* properties should also be included in the concatenation. We use "-" to concatenate a property name and "/" to concatenate a property value.

For example, in Figure 4, the *path strings* for "John," "Doe," and for "Alaska" are "*<td>< tr-class/people>< table>< body>< html>*."

Note that when we concatenate the property DOM node into *path strings*, we only concatenate the *display* property. A *display* property is a property that has an effect on the node's outside appearance when viewed in a browser. Such properties include "font size," "align," "class," and so forth. Some properties such as "href," "src," and "id" are not *display* properties as they generally do not affect the appearance of the node. Thus, including them in the *path string* will make the *path string* overspecified. For this reason, we will not concatenate these properties in the *path string* generation process.

A *path string node value (PSNV)* pair $P(ps, nv)$ is a pair of two text strings, the *path string ps*, and the *node value nv* whose *path string* is *ps*. For example, in Figure 4, "*< td >< tr-class/people><table><body><html>*" and "John" are a *PSNV* pair.

A *perceptual group* of a web page is a group of text components that look similar in the page layout. For example, "Sets," "Tags," "Map," and so on, in the *profile page* of Figure 3 are in the same *perceptual group*; and "U-EET" and "haggard37" are in the same *perceptual group* in the list page in Figure 1.

4.2. DOM Path String Observations. We propose to use the *path string* information for page classification as it has the following benefits.

(1) *Path string efficiency.* First, when we extract *path strings* from the *DOM* tree, we save a significant amount of space, since we do not need to save a *path string* for every *text node*. For example, we only need *one path string* to represent *all* different "tags" in a *detail page* shown in Figure 2, as all these "tags" share the same *path string*. Second, transforming the tree structure into linear string representation will reduce the computational cost.

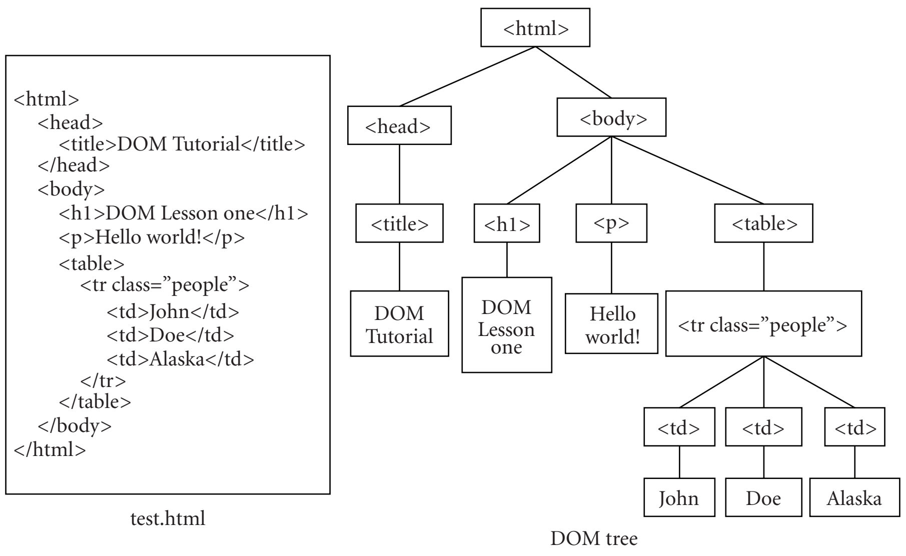
(2) *Path string differentiability.* Using our *path string* definition, it is not hard to verify that *text nodes* "flowers," "canna," and "lily" in Figure 2 share the same *path string*. Interestingly, they share a similar appearance when displayed to users as an HTML page, thus, we say that they are in the same *perceptual group*. Moreover, their display property (*perceptual group*) is different from that of "U-EET," "haggard37," and so on, in Figure 1, which have different *path strings*. Generally, different *path strings* correspond to different *perceptual groups* as the back-end *DOM* tree structure decides the front-end page layout. In other words, there is a unique mapping between *path strings* and *perceptual groups*. At the same time, it is not hard to notice that different types of pages contain different types of perceptual groups. *List pages* generally contain the *perceptual group* of uploader names, while *detail pages* usually contain the *perceptual group* of a list of tags, and their respective *path strings* are different. These observations have encouraged us to use *path strings* to identify different types of pages, and the identification of the types of pages is essential for our crawling, since we want to distinguish between profile and detail pages in order to extract the right ranking for a link.

4.3. Page Classification Using Path String

4.3.1. Extracting Schema Path String Node Value Pairs. Our first step in the page classification process is to extract the schema *PSNV* pairs that occur in all pages. For instance, "Copyright," "Sign in," and "Terms of Use" are the possible text nodes that occur in the schema *PSNV* pairs. We need to ignore such data for more accurate classification. For this case, the schema deduction process is given in Algorithm 1.

In code 1, we adopt a simple way for identifying schema data and real data. That is, if the data value and its *PSNV* pair occur in every page, we identify them as a *schema pair*, otherwise it is considered a *real data pair*. The for loop of line 2–4 performs a simple intersection operation on the pages, while line 5 returns the schema. Note that this is a simple and intuitive way of generating schema. It can be extended by using a threshold value. Then, if a certain *PSNV* pair occurs in at least a certain percent of all the pages, it will be identified as schema data.

4.3.2. Classifying Pages Based on Real Data Path Strings. Noting that the same types of pages have the same *perceptual*



```
<html>
    <head>
        <title>DOM Tutorial</title>
    </head>
    <body>
        <h1>DOM Lesson one</h1>
        <p>Hello world!</p>
        <table>
            <tr class="people">
                <td>John</td>
                <td>Doe</td>
                <td>Alaska</td>
            </tr>
        </table>
    </body>
</html>
```

FIGURE 4: DOM tree of an example web page.

```
Input: $N$ Pages for schema extraction
Output: schema PSNV-pairs, Pi($nv$, PS($nv$)), $i = 1, \ldots, n$
Steps
(1) Schema = All PSNV-pairs of Page 1.
(2) for $i = 2$ to $N$
(3)     do Temp = All PSNV-pairs of Page $i$
(4)         Schema = intersection(Schema, Temp)
(5) Return Schema.
```

ALGORITHM 1: Deduce schema PSNV pairs.

```
Input: $N$ Pages of the same type
        for page type path strings extraction
Output: A Set of Path Strings, PS$i$, $i = 1, \ldots, n$
Steps
(1) Set = All Path Strings of Page 1 - Schema PSs.
(2) for $i = 2$ to $N$
(3)     do Temp = All PSs of Page $i$ - Schema PSs
(4)         Set = intersection(Set, Temp)
(5) Return Set.
```

ALGORITHM 2: Extracting a page type's path strings.

groups and further the same *path strings*, we can use whether a page contains a certain set of *path strings* to decide whether this page belongs to a certain type of pages. For example, as we already know that all *list pages* contain the *path string* that corresponds to *uploader names*, and almost all *detail pages* contain the *path string* that corresponds to *tags*, we can then use these two different types of *path strings* to identify *list pages* and *detail pages*. Algorithm 2 gives the procedure of extracting characteristic *path strings* for pages of a given type.

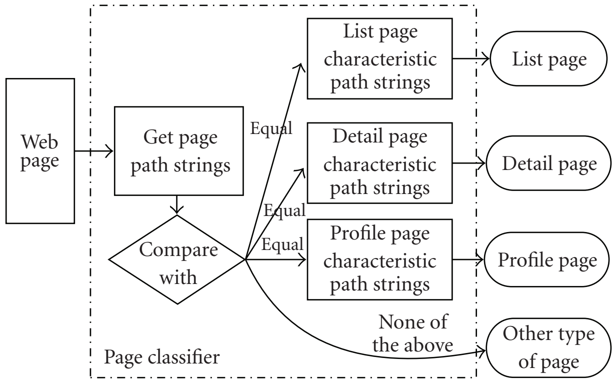


FIGURE 5: Page classifier.

By applying Algorithm 2 on each type of page (*list page, detail page*, and *profile page*) we are able to extract a group of characteristic *path strings* for each type. Then given a new page, the classifier would only need to check whether that page contains all the *path strings* for a group to decide whether that page belongs to that type of page. This process is depicted in Figure 5. Note that most of the time, we do not even need to compare the whole group of *page path strings* with *characteristic path strings*; in fact, a few typical *path strings* would suffice to differentiate different types of pages. For example, our tests on *Flickr* showed that only one *path string* for each type of page was sufficient to do the classification.

5. Profile-Based Focused Crawler

Now, that we are able to identify the correct page type using the *path string* method, we are equipped with the right tool to start extracting the correct information from each type of page that we encounter while crawling, in particular, *profile* *pages*. In this section, we discuss our profile-based crawling system. The basic idea is that from an uploader's profile, we can gain a rough understanding of the topic of interest of the uploader. Thus, when we encounter a media object such as an image or video link of that uploader, we can use this prior knowledge which may relate to whether the image or video belongs to our crawling topic in order to decide whether to follow that link. By doing this, we are able to avoid the cost of extracting the actual *detail page* for each media object to know whether that page belongs to our crawling topic. To this end, we further divide a user profile into two components, an *inner profile* and an *inter profile*.

5.1. Ranking from the Inner Profile. The *inner profile* is an uploader's own property. It comes from the uploader's general description of the media that they uploaded, which can roughly identify the type of this uploader. For instance, a "nature" fan would generally upload more images and thus generate more annotations about nature; an animal lover would have more terms about animals, dogs, pets, and so on, in their profile dictionary. For the case of the Flickr photo-sharing site, an uploader's *inner profile terms* come from the names of their "collections," "sets," and "tags." As another example, for the YouTube video-sharing site, an uploader's *inner profile* comes from their "videos," "favorites," "playlists," and so on. It is easy to generalize this concept to most other multimedia sharing websites.

The process for calculating the *inner profile rank* can be illustrated using Figure 6. After we collect all the profile pages for an uploader, we extract terms from these pages, and get a final profile *term vector*. We then calculate the *cosine similarity* between the profile *term vector* and the topic *term vector* to get the member's *inner profile rank*. We use (1) to calculate a user's inner rank:

$$\text{Rank}_{\text{inner}}(u \mid \tau) = \text{Cos}\left(\vec{x}_u, \vec{x}_\tau\right), \tag{1}$$

where $\vec{x}_u$ is the term vector of the user, and $\vec{x}_\tau$ is the topic term vector.

5.2. Ranking from the Inter Profile. In contrast to the inner profile which gives an uploader's *standalone* property, strictly related to their media objects, we note that an uploader in a typical *social* media-sharing website, tends to also socialize with *other* uploaders on the same site. Thus, we may benefit from using this social networking information to rank a profile. For instance, a user who is a big fan of one topic, will tend to have friends, contacts, groups, or subscriptions, and so forth, that are related to that topic. Through social networking, different uploaders form a graph. However, this graph is typically very sparse, since most uploaders tend to have a limited number of social contacts. Hence, it is hard to conduct a systematic analysis on such a sparse graph. In this paper, we will use a simple method, in which we accumulate an uploader's social contacts' *inner ranks* to estimate the uploader's *inter rank*.

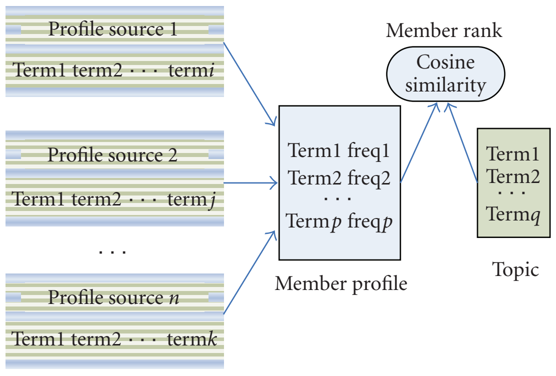


FIGURE 6: Inner profile ranking.

Suppose that a user u has N contacts, c_i, then the *inter rank* of the user, relative to a topic τ, can be calculated using (2) which aggregates all the contacts' inner ranks:

$$\text{Rank}_{\text{inter}}(u \mid \tau) = \frac{1}{N}\sum_{i=1}^{N}\text{Rank}_{\text{inner}}(c_i \mid \tau), \tag{2}$$

where τ is the given crawling topic, and $\text{Rank}_{\text{inner}}(c_i \mid \tau)$ is the user's ith contact's *inner rank*.

5.3. Combining Inner Rank and Inter Rank. For focused crawling, our final purpose is to find the probability of following link L_n given the crawling topic τ so that we can decide whether we should follow the link. Using Bayes rule, we have

$$\Pr\left(L_n \mid \tau\right) = \frac{\Pr\left(\tau \mid L_n\right) * \Pr\left(L_n\right)}{\Pr(\tau)}. \tag{3}$$

Suppose there are N total candidate links, then

$$\Pr(\tau) = \sum_{0<i\le N}\left(\Pr\left(\tau \mid L_i\right) * \Pr\left(L_i\right)\right). \tag{4}$$

Our task is then transformed into calculating the conditional probability $\Pr(\tau \mid L_n)$, that is, given a link, the probability of that link belonging to the crawling topic τ. We propose to calculate the prior based on *inner ranks* and *inter ranks*, such that each factor gives us a reward of following the link. We do this by combining them as follows:

$$\Pr\left(\tau \mid L_n\right) = \alpha \times \text{Rank}_{\text{inner}}(u_m) + \beta \times \text{Rank}_{\text{inter}}(u_m), \tag{5}$$

where L_n is the nth image thumbnail link and u_m is the mth user that corresponds to the nth image thumbnail link. $\text{Rank}_{\text{inner}}(u_m)$ and $\text{Rank}_{\text{inter}}(u_m)$ are calculated using (1) and (2), respectively. We could further normalize $\Pr(\tau \mid L_n)$ to obtain probability scores, however, this will not be not needed, since they are only used for *ranking* links.

6. Cotagging Topic Discovery

To start the focused crawling process, we need to feed the crawler with a crawling topic. The crawling topic should not

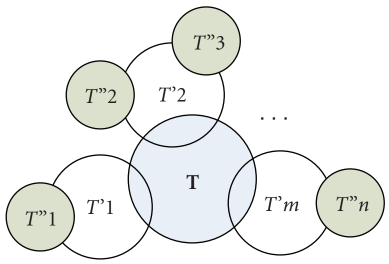


Figure 7: Two-layer cotagging topic discovery.

be set to only one tag as that would be too narrow. For example, if we choose the crawling topic "animals," all tags that are closely related to "animals," which may include "cat," "dog," "pet," and so on, may need to also be included in the crawling topic tags. Hence, to set a crawling topic properly, we need to expand the topic's tagging words. Our method to conduct this task is by exploiting the cumulative image/video cotagging (i.e., tag co-occurrence) information. We use for this purpose, a voting-based method. If one tag, say **T1**, and the topic tag **T** co-occurred in one photo, we count this as one vote of **T1** also belonging to our crawling topic. When we accumulate all the votes through many photos, we would get a cumulative vote for **T1** also belonging to our crawling topic. When such a vote is above a threshold, we will include tag **T1** in our crawling topic tags. This mechanism boils down to using a correlation threshold:

$$\varphi = \frac{P(T \cap T1)}{P(T) \times P(T1)}, \tag{6}$$

where $P(T \cap T1)$ is the number of pictures cotagged by both tag **T** and tag **T1**, and $P(T)$ and $P(T1)$ are the number of pictures tagged by tag **T** and tag **T1**, respectively. Suppose that tag **T** belongs to the crawling topic, then φ gives the score of whether **T1** also belongs to the crawling topic. When φ is bigger than a preset threshold, we will count **T1** as belonging to the crawling topic.

In order to make the crawling topic tags more robust, we further use the following strategies.

(1) Take only one image if multiple images are tagged with an identical set of tags. This is usually because an uploader may use the same set of tags to tag a group of images that they uploaded to save some time.

(2) From the top cotagging tags, start a new round of cotagging discovery. This process is depicted in Figure 7. Then use the expanded cluster of high frequency co-occurring tags as the final crawling topic.

7. Profile-Based Focused Crawling System

We developed a two-stage crawling process that includes a cotagging topic discovery stage and a profile-based focused crawling stage. Both of these stages use the page classifier extensively to avoid unnecessary crawling of undesired types of pages and to correctly extract the right information from the right page. The details of crawling are explained in Sections 7.1 and 7.2.

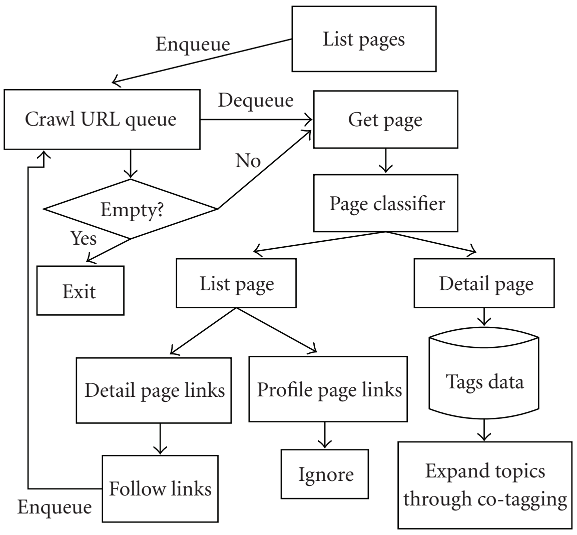


Figure 8: Stage one: cotagging topic expansion stage.

```
Input: Initial Crawling Topic Tag, $T$
       List pages, $p\_1, \ldots, p\_N$
Output: Expanded Topic Tags, $T, T\_1, \ldots, T\_k$
Steps
(1) Set Queue $Q$ = empty
(2) for $i = 1$ to $n$
(3) do Enqueue $p\_i$ into $Q$
(4) while $Q!$ = Empty
(5)     do page $p$ = Dequeue $Q$
(6)         classify $p$
(7)         if $p$ = List Page
(8)             then <$o\_1, \ldots, o\_m$> = Outlinks from $p$
(9)                 if $o\_i$ = Detail Page Link
(10)                    then Enqueue $o\_i$ to Q
(11)                else if $o\_i$ = Profile Page Link
(12)                    then discard $o\_i$
(13)        else if $p$ = Detail Page
(14)            then extract tags data from $p$
(15) analyze the tags to get the most frequent
(16) co-occurring tags <$T, T\_1, \ldots, T\_k$>
(17) return <$T, T\_1, \ldots, T\_k$>
```

Algorithm 3: Stage one: cotagging topic discovery.

7.1. Cotagging Topic Discovery Stage. The first stage of our profile-based focused system is the cotagging topic discovery stage. In this stage, we collect images that are tagged with the initial topic tag, record their cotags, process the final cotagging set, and extract the most frequent co-occurring ones. Figure 8 gives the diagram of the working process of this stage, and Algorithm 3 gives the detailed steps.

In Algorithm 3, lines (4)–(14) do the actual crawling work. The page classifier described in Section 4 is used in

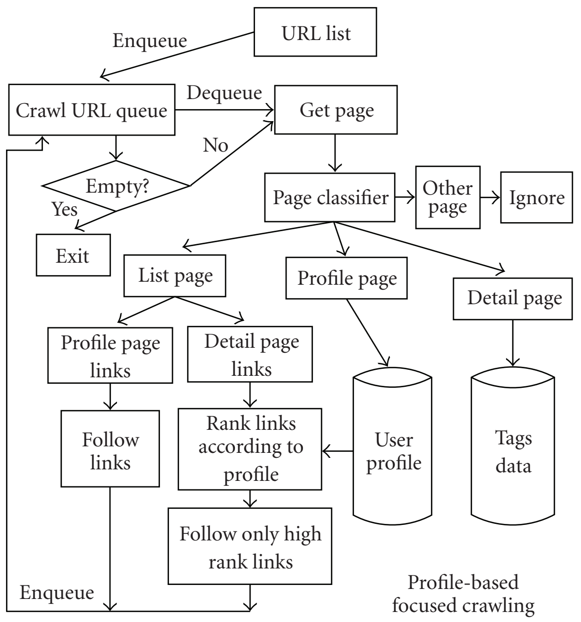


Figure 9: Stage two: profile-based focused crawling stage.

```
Input: Crawling Topic Tags, <T_1, . . . , T_k>
       Crawling URLs < url_1, . . . , url_n>
Output: Crawled Detail Pages
Steps
(1) Queue Q = empty
(2) for i = 1 to n
(3)     do Enqueue url_i into Q
(4) while Q ! = Empty
(5)     do page p = Dequeue Q
(6)         classify p
(7)         if p = List Page
(8)             then <o_1, . . . , o_m> = Outlinks from p
(9)                 if o_i = Detail Page Link
(10)                    then if Rank(u_o_i) > RANK_TH
(11)                        then Enqueue o_i to Q
(12)                    else Discard o_i
(13)                else if o_i = Profile Page Link
(14)                    then Enqueue o_i to Q
(15)        else if p = Detail Page
(16)                then Extract Tags Data from p
(17)        else if p = Profile Page
(18)                then Extract Prof Data from p
(19)        else if p = Other Type Page
(20)                then ignore p
(21) Return Detail Pages Tags Data
```

Algorithm 4: Stage two: profile-based focused crawling.

line (6) to decide whether a page is a *list page* or a *detail page*. We already know that in social media-sharing websites, *list pages* have outlinks to *detail pages* and *profile pages*, and we name such links *detail page links* and *profile page links*, respectively. It is usually easy to differentiate them because in the *DOM* tree structure, *detail page links* generally have image thumbnails as their children, while *profile page links* have *text nodes*, which are usually the uploader names, as their children. Combined with our *path string* method, we can efficiently identify such outlinks. In lines (11)-(12), by not following *profile page links*, we save a significant amount of effort and storage space. Since we are not following *profile page links*, the classification result for page p in line (6) would not be a *profile page*. Lines (15)-(16) do the cotagging analysis and line (17) returns the expanded topic tags.

7.2. Profile-Based Focused Crawling Stage. In the second stage, which is the actual crawling stage, we use the information acquired from the first stage to guide our focused crawler. For this stage, depending on the system's scale, we can choose to store the member profiles either on disk or in main memory. The system diagram is shown in Figure 9, and the process detail is shown in Algorithm 4. In Algorithm 4, similar to the cotagging stage, we classify page p in line (6). The difference is that, since we are not pruning *profile page links* in lines (13)-(14) and we follow them to get the user profile information, we will encounter the *profile page* branch in the classification result for line (6), as shown in lines (17)-(18). Another difference is how we handle *detail page links*, as shown in lines (10)–(12). In this stage, we check whether a *detail page link*'s *user profile rank* according to the crawling topic. If the rank is higher than a preset threshold, $\text{RANK}_T H$, we will follow that *detail page link*, otherwise, we will discard it. Note that in this process, we need to check whether a user's *profile rank* is available or not, which can be done easily by setting a *rank available flag*, and we omit this implementation detail in the algorithm. In lines (17)-(18), we process profile pages and extract profile data. Another issue is deciding when to calculate the user profile rank since the profiles are accumulated from multiple pages. We can set a fixed time interval to conduct the calculation or use different threads to do the job, which is another implementation detail that we will skip here.

8. Experimental Results

8.1. Path String-Based Page Classification. Our tests on Flickr and YouTube showed that only one or two *path strings* suffice to get a 100% classification accuracy. Hence, we will not give further experimental results on the page type classification. Instead, we will demonstrate the performance of the more challenging *path string* differentiation for the same page type on different websites. This experiment serves to see how the *path string* can differentiate different schema data from real-value data. Our assumption for using the *path string* method to extract web data is that the path string for schema data and for real data share little in common. Thus, we can first use path strings to differentiate real data and schema data. In case the *path string* cannot totally differentiate among the two, we can further use node data value to differentiate between them. Also, we assume that using the *path string* method, if we do not need to consider schema *path strings*, then we save a lot of effort for extracting real data. For this experiment, we used "wget" to download

TABLE 1: Path string differentiation.

Site	T	S	V	US	UV	INT
Flickr	133	111	22	36	16	3
YouTube	488	179	309	40	73	9
Amazon(book)	837	411	426	101	115	22
Ebay	474	183	291	56	113	15
SpringerLink	140	100	40	27	20	5
ACM DL	124	62	62	15	19	4

TABLE 2: Top cotagging tags for the topic "flowers."

Flowers	Flower	Nature	Macro	Spring
Yellow	Pink	Garden	Green	White
Plants	Red	Flowers	Purple	Blue

the real web data from the popular sites, "Flickr," "YouTube," "Amazon," and so forth. For each website, we randomly downloaded 10 pages of the same type. For instance, in the Amazon book site, we only downloaded the pages that contain one detailed information of a specific book. For "Flickr," we only downloaded the page that contains the detailed image page. We will name these pages *object pages*. After downloading these object pages, we *use* our implementation (written in java, and using the *nekohtml* parser APIs, http://people.apache.org/~andyc/neko/doc/html, for parsing the web page) to build the DOM tree and conduct our experiments. The results are shown in Table 3, where T is the number of total *PSNV* pairs, S is the number of schema PSNV pairs, V is the number of value data *PSNV* pairs, and US is the number of *unique path strings* for schema data. Notice that some schema data with different text data value may share the same path string. The same applies to value data. Different value data may also share the same *path strings*. UV is the number of *unique path strings* for value data. Finally, INT is the number of intersections between US and UV. We can see from this table that our assumption is well founded. The low intersections between US and UV means that very few pages have the same *path strings* for schema data and for true value data. This tells us that we can indeed use *path strings* to differentiate between schema data and real data. Also, notice that the number of *unique path strings* is much lower than the number of actual *PSNV* pair (US is less than S, UV is less than V), this means that converting from a text node value *path string* to *unique path strings* can save some time and space in processing.

8.2. Topic Discovery through Cotagging. We tested two topics for the cotagging topic discovery process using Flickr photo-sharing site. In the first test, we used the starting tag "flowers," and we collected 3601 images whose tags contain the keyword flowers. From this 3601-image tag set, we found the following tags that occur in the top cotagging list (after removing a few noise tags such as "nikon," that are easy to identify since they correspond to camera properties and not media object properties).

TABLE 3: Top cotagging tags for "nyc."

nyc	New	York	City	Manhattan
Brooklyn	Street	Art	NY	Newyork
Graffiti	Winter	Park	Gothamist	USA

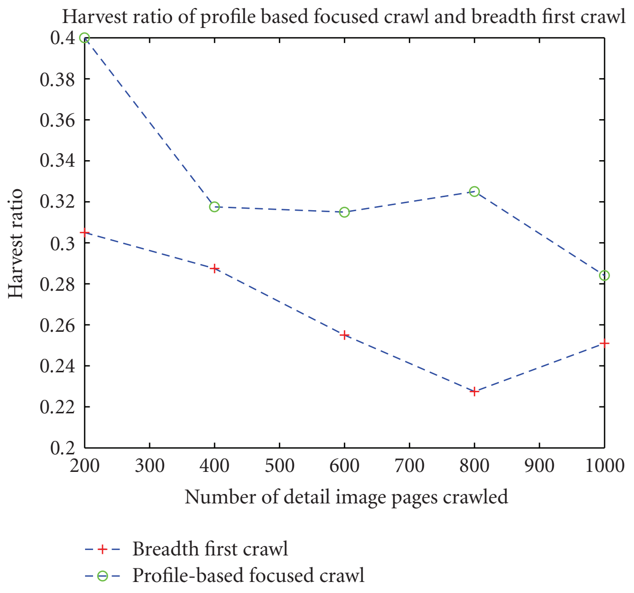


FIGURE 10: Crawling harvest ratio for topic "flowers" (threshold = 0.01).

In the second round of tests, we used the starting tag "nyc," and after collecting 3567 images whose tag sets contain "nyc," we obtained the following expanded topic tag set.

We can see that these results are reasonable. We then used these two sets of crawling topics for the following focused crawling experiments.

8.3. Profile-Based Focused Crawling. The *harvest ratio* is often used to evaluate focused crawlers. It measures the rate at which relevant pages are acquired and how effectively irrelevant pages are filtered. We will calculate the harvest ratio using the following formula:

$$\text{Harvest Ratio} = \frac{N_r}{N_a}, \tag{7}$$

where N_r is the number of relevant pages (belonging to the crawl topic) and N_a is the number of total pages crawled. To calculate the harvest ratio, we need a method to calculate the relevancy of the crawled pages. If the crawled page contains any of the tags that belong to the crawl topic, we would consider this page as relevant, otherwise it will be considered as irrelevant. For comparison, we compared our focused crawling strategy with the breadth-first crawler.

We also conducted this test on the Flickr photo-sharing site. We started our crawler with a list of URLs with popular tags (easily obtained from the main page on Flickr). Our first stage breadth-first crawler starts by recording the uploader profiles that it extracted from the crawled pages. Later in

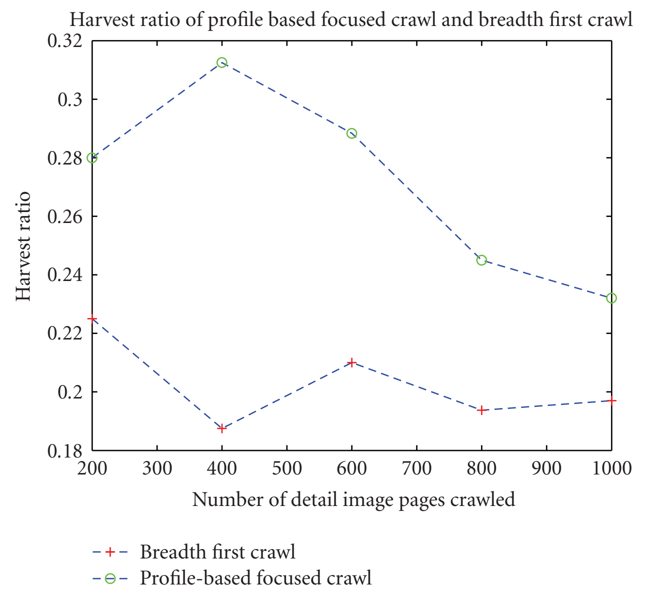


Figure 11: Crawling harvest ratio for topic “nyc” (threshold = 0.01).

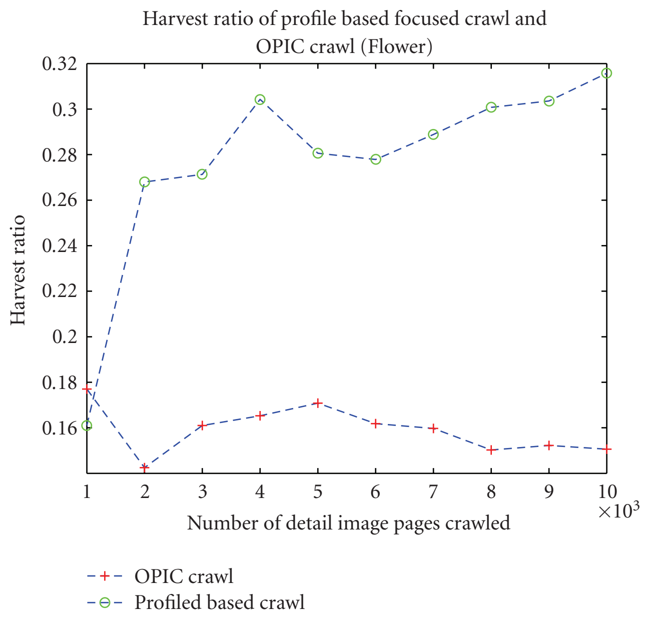


Figure 13: Crawling harvest ratio for topic “flower” (793 valid user profiles accumulated).

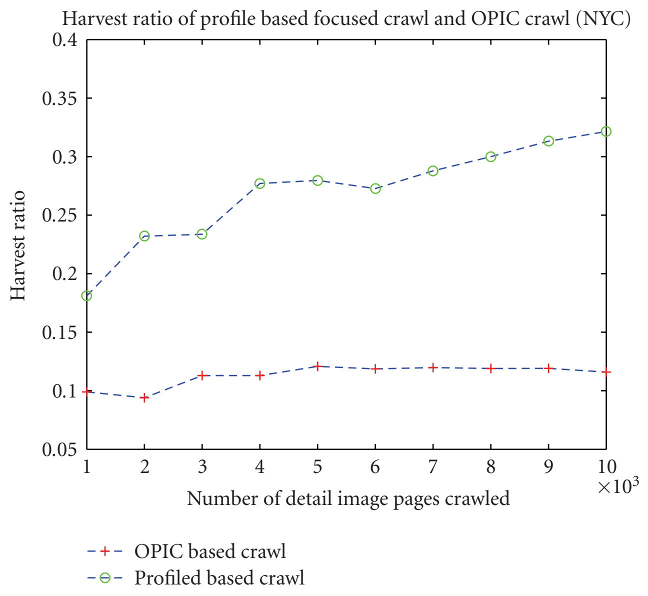


Figure 12: Crawling harvest ratio for topic “nyc” (706 valid user profiles accumulated).

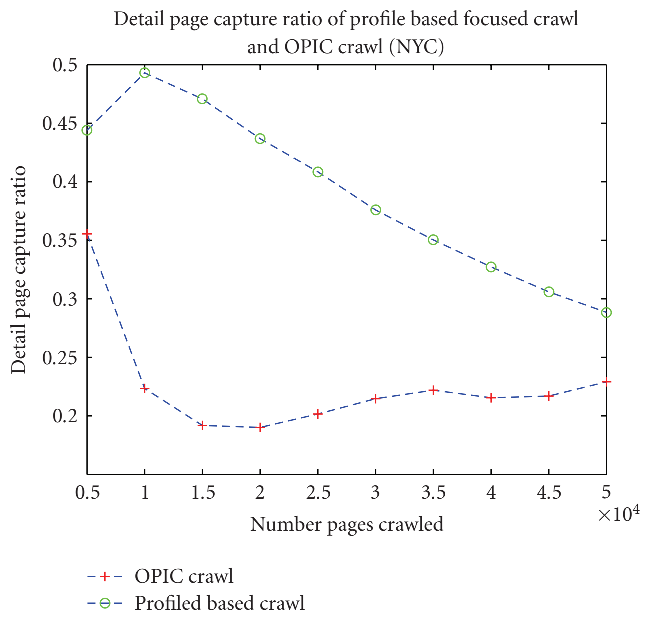


Figure 14: Detail page capture ratio for topic “nyc” (706 valid user profiles accumulated).

our second stage of profile-based focused crawling, we read these profiles, and calculate the corresponding ranks for each outlink according to the user profile. We then prune outlinks with scores lower than a threshold value. Note that in the harvest ratio calculation, we only count the *detail image links* traversed. Figures 10 and 11 give comparisons of focused crawling and breadth-first crawling for two crawling topics, “flowers” and “nyc,” respectively. We can see that our harvest ratio for profile-based focused crawling exceeds that of the breadth-first crawling by a significant margin.

In the next set of experiments, we compared our profile-based focused crawler with that of the OPIC crawler [12] for both the topic “nyc” and “flower.”

For the profile-based crawling, we adjusted the crawling strategy used by OPIC [12] to take the user profile and crawling topic into account. Once we encounter a list page, if we find that the list page is from the crawling topic list (by checking its URL), we reset the score of that link to the initial maximum value (1.0), while we reset the detail page scores or profile page link scores according to their corresponding

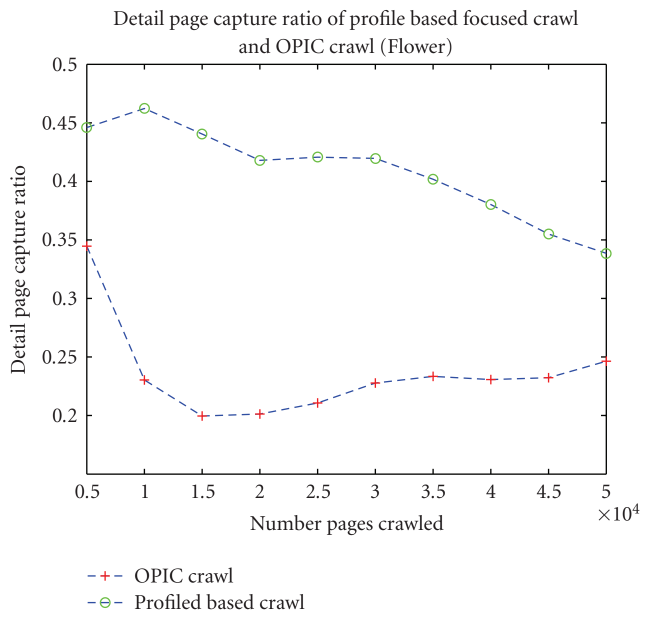


FIGURE 15: Detail page capture ratio for topic "flower" (793 valid user profiles accumulated).

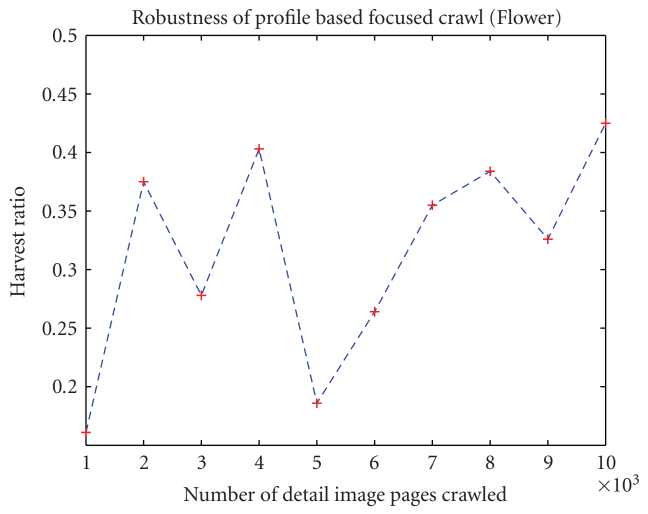


FIGURE 16: Robustness of profile-based crawler for topic "flower."

user profile scores. For the rest of the links, we adopt the OPIC scores. We can see from the results that profile-based focused crawling has a much better harvest ratio than purely OPIC-based crawling. The harvest ratios, for the two topics, are shown in Figures 12 and 13.

We also performed experiments to compare the detail page capture ratio between profile-based focused crawling and OPIC-based crawling.

We can see that in both cases, the detail page capture ratio is higher for the profile-based focused crawler than for the purely OPIC-based crawler.

Finally, we performed robustness experiments on both topics to evaluate how stable a profile-based focused crawler is. Unlike the above harvest ratio experiments, in the robustness experiments, we used a *sliding window* of 1000 pages to observe the harvest ratio on each set of 1000 pages, while for the general harvest ratio experiments we measured the *cumulative* harvest ratio on 1000 pages, 2000 pages, . . . , 10 000 pages. From the experimental results shown in Figures 16 and 17, we can see that for both topics, the profile-based focused crawler is reasonably robust.

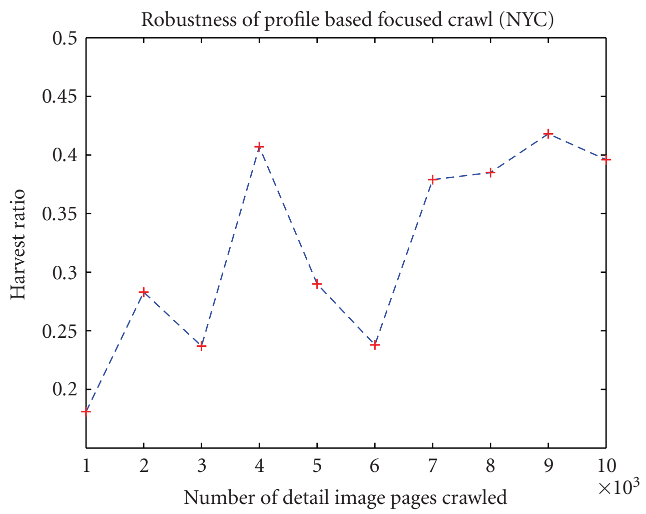


FIGURE 17: Robustness of profile-based crawler for topic "nyc."

9. Conclusions and Future Work

We presented a profile-based focused crawler, which ranks users with more topic-relevant media objects higher during crawling. To further differentiate profiles while taking into account the special characteristics of *social* media sites, we have introduced and used the notions of the *inner profile* and *inter profile*. We have used cotagging in a first stage, for automated crawling topic discovery, and thus build a consistent set of tags for a given topic. In both the cotagging topic discovery process and the profile-based focused crawling process, we used a *path string*-based page classification scheme in order to allow us to extract the correct type of information from each page type, and in order to correctly calculate the profile ranks for a given topic. Our experimental results confirmed the effectiveness of our profile-based focused crawling system from the perspective of harvest ratio and robustness. In the future, we would like to deploy the proposed focused crawling on a real system for real-time vertical social media search.

References

[1] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer Networks*, vol. 31, no. 11–16, pp. 1623–1640, 1999.

[2] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 307–318, Seattle, Wash, USA, June 1998.

[3] F. Menczer, G. Pant, and P. Srinivasan, "Topical web crawlers: evaluating adaptive algorithms," *ACM Transactions on Internet Technology*, vol. 4, no. 4, pp. 378–419, 2004.

[4] G. Pant and P. Srinivasan, "Learning to crawl: comparing classification schemes," *ACM Transactions on Information Systems*, vol. 23, no. 4, pp. 430–462, 2005.

[5] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, "On the design of a learning crawler for topical resource discovery," *ACM Transactions on Information Systems*, vol. 19, no. 3, pp. 286–309, 2001.

[6] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, "Intelligent crawling on the world wide web with arbitrary predicates," in *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 96–105, Hong Kong, May 2001.

[7] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused crawling using context graphs," in *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, pp. 527–534, Cairo, Egypt, September 2000.

[8] C.-C. Hsu and F. Wu, "Topic-specific crawling on the web with the measurements of the relevancy context graph," *Information Systems*, vol. 31, no. 4-5, pp. 232–246, 2006.

[9] M. L. A. Vidal, A. S. da Silva, E. S. de Moura, and J. M. B. Cavalcanti, "Structure-driven crawler generation by example," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pp. 292–299, Seatttle, Wash, USA, August 2006.

[10] Z. Zhuang, R. Wagle, and C. L. Giles, "What's there and what's not?: focused crawling for missing documents in digital libraries," in *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries (JCDL '05)*, pp. 301–310, Denver, Colo, USA, June 2005.

[11] J. Qin, Y. Zhou, and M. Chau, "Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method," in *Proceedings of the 4th ACM/IEEE Joint Conference on Digital Libraries (JCDL '04)*, pp. 135–141, Tucson, Ariz, USA, June 2004.

[12] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive on-line page importance computation," in *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pp. 280–290, Budapest, Hungary, May 2003.

[13] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: towards automatic data extraction from large web sites," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pp. 109–118, Roma, Italy, September 2001.

[14] S. Grumbach and G. Mecca, "In search of the lost schema," in *Proceedings of the 7th International Conference on Database Theory (ICDT '99)*, vol. 1540 of *Lecture Notes in Computer Science*, pp. 314–331, Jerusalem, Israel, January 1999.

[15] A. Arasu, H. Garcia-Molina, and S. University, "Extracting structured data from web pages," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 337–348, San Diego, Calif, USA, June 2003.

[16] Y. Zhai and B. Liu, "Web data extraction based on partial tree alignment," in *Proceedings of the 14th International Conference on World Wide Web (WWW '05)*, pp. 76–85, Chiba, Japan, May 2005.

[17] Z. Li, W. K. Ng, and A. Sun, "Web data extraction based on structural similarity," *Knowledge and Information Systems*, vol. 8, no. 4, pp. 438–461, 2005.

[18] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, "Dom-based content extraction of html documents," in *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pp. 207–214, Budapest, Hungary, May 2003.

[19] H. Zhao, W. Meng, and C. Yu, "Mining templates from search result records of search engines," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 884–893, San Jose, Calif, USA, August 2007.