

# A Structure-Oriented Unsupervised Crawling Strategy for Social Media Sites

Keyang Xu

Language Technologies Institute  
Carnegie Mellon University  
Pittsburgh, PA, USA 15213  
xky0714@gmail.com

Kyle Yingkai Gao

Language Technologies Institute  
Carnegie Mellon University  
Pittsburgh, PA, USA 15213  
kyle.ygao@gmail.com

Jamie Callan

Language Technologies Institute  
Carnegie Mellon University  
Pittsburgh, PA, USA 15213  
callan@cs.cmu.edu

## ABSTRACT

Existing techniques for efficiently crawling social media sites rely on URL patterns, query logs, and human supervision. This paper describes SOUrCe, a structure-oriented unsupervised crawler that uses page structures to learn how to crawl a social media site efficiently. SOUrCe consists of two stages. During its unsupervised learning phase, SOUrCe constructs a sitemap that clusters pages based on their structural similarity and generates a navigation table that describes how the different types of pages in the site are linked together. During its harvesting phase, it uses the navigation table and a crawling policy to guide the choice of which links to crawl next. Experiments show that this architecture supports different styles of crawling efficiently, and does a better job of staying focused on user-created contents than baseline methods.

## KEYWORDS

Web Crawling; Page Clustering; Link Prediction;

### ACM Reference format:

Keyang Xu, Kyle Yingkai Gao, and Jamie Callan. 2018. A Structure-Oriented Unsupervised Crawling Strategy for Social Media Sites. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17)*, 10 pages.  
DOI: 10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Web crawlers use a variety of methods to identify and download a useful set of web pages within time and bandwidth constraints. There are several differences between social media crawlers and general web crawlers. One important difference is the opportunity for a social media crawler to use information about how a site is organized to crawl it more efficiently. Prior research observed that social media sites tend to have large regions of consistent structures where web pages are generated dynamically using a small number of page templates and content stored in a database. Knowing this structure enables a crawler to prioritize some types of pages (e.g., recent user-generated content) over others, or to spread its effort evenly to obtain a representative sample [4, 6, 15], or to avoid downloading the same page via different URLs [26]. Information about how the site is organized is provided manually, recognized

---

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

*Conference'17, Washington, DC, USA*

© 2018 Copyright held by the owner/author(s). 978-x-xxxx-xxxx-x/YY/MM...\$15.00  
DOI: 10.1145/nnnnnnn.nnnnnnn

by heuristics, or learned by recognizing consistent patterns in the site [4, 13, 15, 25, 28].

Most prior work uses URL patterns to represent page templates and to classify uncrawled links. However, URL naming conventions are not necessarily stable for pages created at different times, and crawlers must cope with duplication and redirection problems. Simple patterns may be learned without supervision, but websites with complex URL patterns may require training data that must be generated by other methods [17, 18]. During the crawling phase, other resources, supervision, or policies are required to convert a set of URL patterns and a queue of uncrawled links into link scores or traversal paths that guide the crawler.

Using page templates to represent site structure is a powerful intuition. However the correspondence between URL patterns and page templates is a weak signal for fully exploiting this intuition. This paper proposes a new approach to crawling social media that infers the presence of page templates from the structure of the web pages themselves, rather than from their URLs. Web pages are longer and have more structure than URLs, which provides more information for inferring the presence of page templates and for deciphering how they are interconnected. Our contributions are three-fold:

- An automatic method of clustering and classifying web pages based only on structural information;
- A crawler that predicts the page type of an uncrawled link without the help of URL patterns, content information, or human supervision; and
- A crawling framework that naturally supports several distinct crawling policies.

Experimental results show that the new approach to crawling social media is better than or comparable to existing methods that require external information.

The rest of this paper is organized as follows. Section 2 gives a brief review of related prior work. Sections 3 and 4 describe the problem setting, framework, and SOUrCe crawler architecture. Section 5 reports on data collection, and experiments with SOUrCe and several baseline methods. Finally, Section 6 summarizes the paper and concludes.

## 2 RELATED WORK

We review two lines of studies that are relevant to our work: web crawling strategy and web pages clustering.

### 2.1 Web Crawling Strategy

Typically web crawlers traverse the web, which is a connected graph, by recursively recognizing links in a page to other pages, and

downloading those other pages. Due to resource limitations such as bandwidth, crawlers must be selective in choosing which among its discovered links to crawl next. A good intuition is weighting links based on their usefulness and importance, which can be estimated by PageRank and in-degree. However, link structure can be difficult to recognize accurately while crawling. Prior research estimates importance using partial graphs [5, 6], historically crawled data [3], cash flow [1] and other features such as the host domain and page titles [2].

Cho, et al. [5] proposed to weight links based on in-links count and partial PageRank calculation. Baeza-Yates, et al. [3] indicated using historically crawled data is effective in ordering a crawling queue, as well as estimating the PageRank of unseen links. OPIC [1] utilizes the concept of cash flow and estimates the importance of a link by “cash” that is distributed equally by pages that point to it. RankMass [6] uses all available links during crawling to calculate a lower bound on PageRank to prioritize the crawling queue. Fractional PageRank [2] incorporates features such as host domain and page title in computing PageRank scores and proposes to reduce the computational cost by skipping the path towards a set of already downloaded pages.

The recent importance of social media requires crawlers that are effective within specific sites. iRobot [4] clusters pages using both repetitive regions and URL patterns, and filters out unimportant clusters via an informativeness measure. In its crawling phase, a traversal path is generated, which is a spanning tree on the graph, to guide crawlers. However, the process of generating a traversal path requires human supervision. Importance of clusters can be estimated by hub and authority scores [20]. However, this work utilizes search logs, which is an additional external resource.

Vidal et al.[25] propose TPM, which focuses on crawling a specific type of pages, known as “target”, from one website. An example page is first used to find pages that share similar structure in a set of pages; then TPM learns URL navigation patterns that lead to target pages efficiently. Jiang et al.[15] propose FoCuS, which is a supervised method using regular expressions to find thread and index pages in web forums. Zhang et al. [28] explore focused crawling in social media by classifying pages into pre-defined groups. However, both methods are supervised [15, 28], which can be not generalizable when crawling other sites.

## 2.2 Web Pages Clustering

Web pages clustering methods can be categorized into four types based on features employed: link-based, content-based, URL-based and structure-based.

Link-based methods [7, 9] state that links are considered to be topically similar if they share similar parent nodes. Content-based approaches [23, 24] categorize web pages according to the words and sentences contained. These methods argue that pages with similar functions share similar topic features. The vector space model [22] is commonly used to represent documents. However, content-based approaches are limited due to vagueness of topical differences and difficulties in understanding cross-language websites. URL-based methods [14, 17, 18] have an advantage of clustering pages without downloading files. However, dealing with duplication [26] and redirection problems is a non-trivial task. Once

the patterns of URLs are decided, regular expression can be used to cluster and classify web pages. Structure-based approaches are more general ways to cluster web pages. Some previous research estimates the similarity of pairwise web pages through their tree structures [10, 21]. However, it is computationally expensive to do pairwise page comparison and not suitable for web-scale applications. Another idea is representing web pages with vector space models. For example, Crescenzi et al. [8] indicate that two pages are considered to be structurally similar if they have common links placed in the same location. Minimum Description Length Principle (MDL) is adopted in combining small clusters. Grigalis et al [12] further explore this idea by exploiting Xpaths in recording positions of specific links. Their method, known as UXCluster, is claimed to be more accurate and efficient than the method produced by pairwise distance. Cai et al [4] propose to cluster web pages represented by repetitive region patterns extracted by web wrappers [27, 29]. A drawback of most of structure-based clustering is that it requires heuristic threshold to decide whether to form a new cluster or to merge into an existing cluster.

## 3 PROBLEM SETTING

We begin by introducing some definitions and notations that are used in this paper. Then we present two important tasks: (1) Using layout structure to represent and cluster web pages; and (2) analyzing the potential destination cluster of a given anchor link.

### 3.1 Definitions

**Page Type:** For a given social media site, pages that serve the same purpose are usually generated by the same layout template. Each layout template defines a distinct page type. We denote the collection of page types for a given site as  $\mathcal{T} = \{T_1, T_2 \dots T_m\}$  and  $|\mathcal{T}| = m$ .

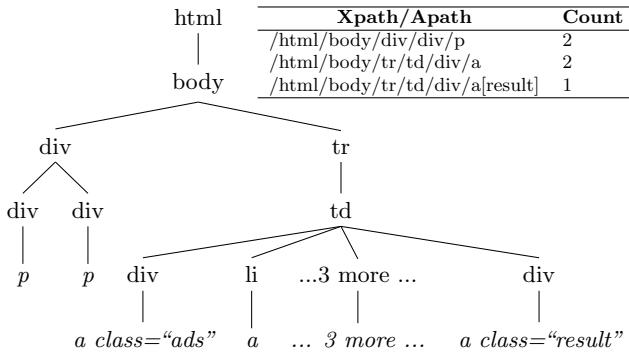
**Page Cluster:** We use a clustering algorithm to produce page clusters, each of which contains structurally similar pages. We denote the collection of page clusters as  $C = \{C_1, C_2 \dots C_n\}$  and  $|C| = n$ . The crawler’s task is to produce a collection of clusters  $C$  that matches as closely as possible to the collection of (hidden) page types  $\mathcal{T}$ .

**Sitemap:** Following the previous definition [4], a sitemap is a directed graph consisting of a set of vertices and existing arcs, where each vertex represents a page cluster and each arc indicates an observed anchor link. Multiple arcs between a pair of clusters  $(C_i, C_j)$  indicates that there are multiple links from pages in cluster  $C_i$  to pages in cluster  $C_j$ .

**Adjacency Matrix:** At the cluster levels, we define the adjacency matrix for graph  $G = (V, E, C)$  as  $A$ , where  $C$  are the page clusters for set of pages  $V$  and set of directed anchor links  $E$ . The size of  $A$  is  $|C| \times |C|$  and its element  $A_{ij}$  represents the weight between the directed pair  $(C_i, C_j)$ , which is the total number of links from any node  $V_m \in C_i$  directed to  $V_n \in C_j$ .

### 3.2 Structural features for Web Pages

A web page can be represented as a tree structure, following the Document Object Model (DOM). Nodes in the hierarchical DOM tree represent HTML elements. We use a bag-of-Xpaths model to represent the DOM tree. An Xpath is defined as a string that



**Figure 1: An example of an HTML DOM Tree. Paths that end with element ‘a’ are instances of anchor paths. [ ] in the Apath contains the class attributes for element ‘a’.**

concatenates HTML elements from the root to a leaf node. We only extract Xpaths that end with a leaf node, such as anchor links, texts or images, because they are the elements displayed in browsers. Similarly to the bag-of-words model for text retrieval, we use term frequency and inverse document frequency [22] to weight an Xpath  $x$  in a web page  $d$  as follows:

$$w_{x,d} = \log(tf_{x,d} + 1) \times \log\left(\frac{|D|}{df_{x,D}} + 1\right) \quad (1)$$

$tf_{x,d}$  is the frequency of Xpath  $x$  in document  $d$ ,  $|D|$  is the size of page collections  $D$ , and  $df_{x,D}$  is the number of pages that contain Xpath  $x$  in  $D$ . The log of  $tf_{x,d}$  prevents high frequency Xpaths from dominating the clustering process. Inverse document frequency reduces the weight of Xpaths that appear on almost every page. After extracting features for each page, we apply L1 normalization:

$$w_{x,d} = \frac{w_{x,d}}{\sum_{x'} w_{x',d}} \quad (2)$$

Euclidean distance is used to measure the similarity between two pages.

### 3.3 Analyzing Anchor Links

**Anchor Path:** Anchor links are identified by Xpaths that end with html element ‘a’. We call them *anchor paths (Apath)*. Note that we only focus on anchor links that direct to pages **within the same website**. SOUrCe uses all Xpaths in clustering pages and utilizes Apaths in helping crawlers to traverse the web graph. Figure 1 presents an example DOM tree and several instances of Xpaths and Apaths generated from the tree.

The same Apath may represent different semantics on the same page, especially in a table element. For example, in Figure 1, the Apath /html/body/tr/td/div/a represents both links for advertisements and results. To reduce these ambiguities, we take advantage of Cascading Style Sheets (CSS), which describes properties of HTML elements, by adding `class` attributes for leaf elements of Apaths.

**Apath Navigation Table:** An ideal social media crawler predicts whether an anchor link is worth following before downloading its destination page. Unlike prior studies that utilize URL patterns and rule-based methods, SOUrCe generates a navigation table to

**Figure 2: Example of a page with identical Apaths leading to the same destination page cluster.**

guide the crawler using only structural information. As in prior work [8, 12], our intuition is that links extracted from identical Apaths on the same page are likely to point to the same destination page types. Figure 2 shows an example in which links in dashed ovals with the same color are extracted from the same Apaths and direct to the same page type. In our work, we hold a weaker assumption that the destination pages of links extracted by the same ( $cluster, Apath$ ) pair have the same probability distribution. Suppose we have a page  $p$  from cluster  $C(p)$ , and extract an anchor link  $u$  from  $p$ . Then the conditional probability distribution for the destination page  $d(u)$  satisfies:

$$\sum_{C_i \in C} P(d(u) \in C_i | C(p), x_u) = 1 \quad (3)$$

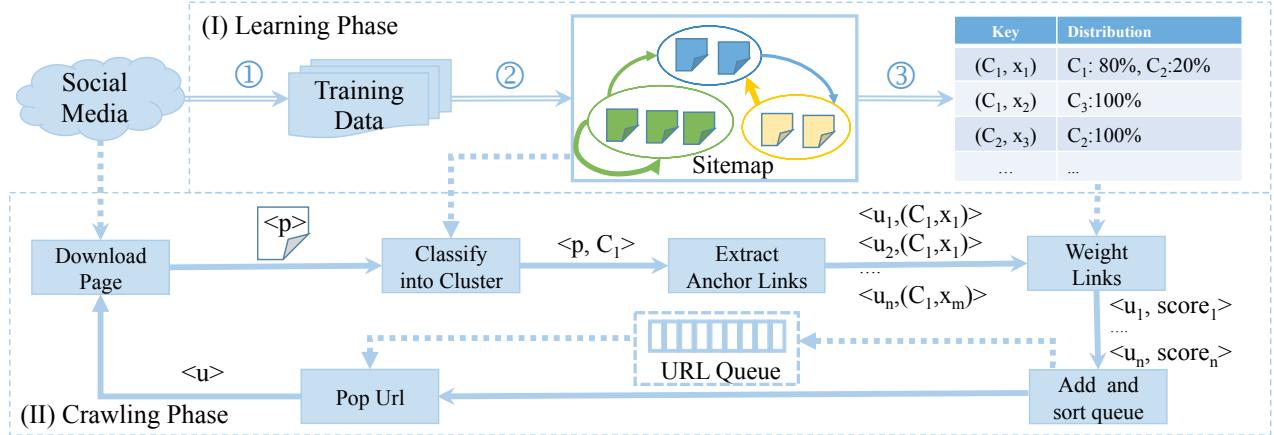
where  $C_i$  is a cluster from  $C$ , and  $x_u$  represents the Apath from which  $u$  is extracted in page  $p$ . We also denote the probability of directing to a cluster  $C_i$  as  $P(C_i | C(p), x_u)$ .

## 4 CRAWLER ARCHITECTURE

Figure 3 shows the crawling architecture. It contains two main components: (i) An unsupervised learning phase; and (ii) an online crawling phase. In this section, we introduce the learning phase in three steps, namely training data sampling, sitemap construction and navigation table generation. Then we discuss two common crawling scenarios, crawling-for-UCC and crawling-for-target, for the online crawling phase.

### 4.1 Sampling Training Data

The first step of the unsupervised learning stage (① in Figure 3) is to obtain an initial sample of pages  $D$  that can be used to infer the site’s structure (i.e., construct a sitemap and a navigation table). There are two requirements: (i) Most of the ( $cluster, Apath$ ) pairs should be covered, otherwise there will not be enough data for generating the



**Figure 3: Structure-oriented unsupervised crawler (SOURCe) architecture.** Arrow  $\Rightarrow$  indicates three steps in the learning phase, concatenating three generated instances, including training data, sitemap and navigation table. Arrow  $\longrightarrow$  connects flows of continuous procedures in the crawling phase and arrow  $\dashrightarrow$  feeds an instance as input to a procedure.

Apath navigation table; and (ii) the training set should be as small as possible, to ensure total crawling efficiency. In previous work, the training data were sampled using a double-ended queue [4] or breadth-first search (BFS) [25]. However, neither of these strategies are well-designed to meet these two requirements.

To explain the intuition of our sampling method, we use the example of links extracted from the red oval in Figure 2. Using breadth-first search, all of the ten links would be crawled as training data. However, since their destination pages are very likely to be in one common page cluster, usually it is not productive to crawl all of them. Thus, only one randomly-selected link is crawled for each unique Apath; other uncrawled URLs are saved in a list  $U_{p,x}$  for learning the navigation table later. This strategy only uses 10% of the bandwidth required by BFS to sample the red oval of Figure 2. The remaining bandwidth can be used to sample other Apaths, or can be left unused to improve crawler efficiency.

## 4.2 Sitemap Construction

The next step (Figure 3, part ②) is to construct the sitemap. SOURCe extracts features from the training pages, as described in Section 3.2, and then uses DBSCAN [11] to cluster them. DBSCAN has well-defined processes for generating new clusters and dealing with outliers. Outliers represent page templates with few instances; typically they are administrative and other pages that are not the main content of the site. Usually social media is crawled to obtain the site's main areas of user-created content (UCC), which appears in pages generated from a small number of commonly-used page templates. Outlier recognition enables trivial parts of the site to be ignored during crawling, if desired.

**DBSCAN clustering** is a density based clustering algorithm proposed by Ester et al. [11]. It incrementally groups points that are closely packed together, and labels points in low density regions as outliers. Two parameters,  $minPts$  and  $eps$  are required.  $minPts$  is the minimum number of points to form a dense region, also known as a cluster.  $eps$  is the maximum distance between two points that are close enough to be in the same region, which is supposed to be

a value specific for each dataset. In this work,  $minPts$  is set to be 4, following DBSCAN authors' suggestion.

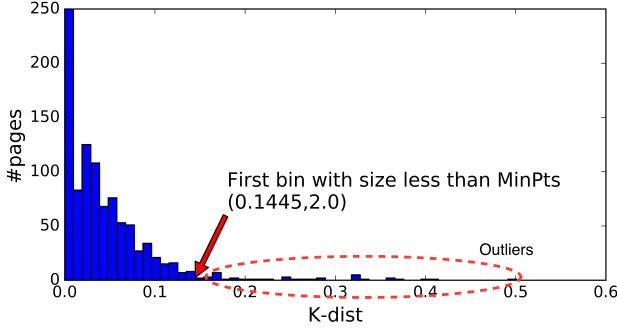
**Setting distance threshold:** Previous page clustering methods usually select a constant and heuristic distance threshold for generating new clusters. However, the optimal threshold may vary for different datasets. In DBSCAN [11], finding  $eps$  is considered as finding the “valley point” on a K-dist graph, where  $K$ -dist is the distance between a node and its  $K$ th nearest neighbor and  $K = minPts$ . However, this method prefers a small value of  $eps$  and creates more outliers than expected. To improve this, we plot the sorted  $K$ -dist histogram of the training data with  $Minpts = 4$ . Figure 4 shows an example of  $K$ -dist histogram for the Hupu dataset. The number of bins is proportional to the number of features as:  $NumBins = w \cdot NumFeats$ , where  $w$  is a parameter only related to the size of training data. For example,  $w = 4.8$  when the training data has 1,000 pages.

From small to large  $K$ -dist, we find the sizes of bins typically decreases, similar to a power-law distribution. Therefore, our method chooses to set a cut-off size value to  $minPts$  for bins in histogram. Specifically, it scans  $K$ -dist values on X-axis from small to large and stop at the first bin with size less than  $minPts$  and select the X-axis value to be  $eps$ . Another trick to prevent small  $eps$  is to ensure the cumulative number of pages at optimal  $eps$  is larger than 50% of the size of training data at the same time. An example is shown in Figure 4.

**Classify a new page:** Given a newly-crawled page, its bag-of-Xpaths features are extracted, and it is classified into an existing cluster using k-Nearest Neighbor with  $k = minPts - 1$ . Note that new pages can also be classified as outliers.

## 4.3 Apath Navigation Table Generation

The third and last phase of the unsupervised learning phase (Figure 3, part ③) is to generate the Apath navigation table. As stated in Section 4.1, during the sampling SOURCe only enqueues one URL  $u$  from all of the URLs extracted from the same Apath  $x$  within one page  $p$ ; but, it saves the whole URL list  $U_{p,x}$  for learning the



**Figure 4: Sorted K-dist histogram for Hupu dataset, which contains 1000 pages and 250 Xpath features. The estimated  $\epsilon$  is 0.1455 where  $K=3$ . Points with  $K\text{-dist} > 0.1455$  are considered as outliers.**

navigation table. After the sitemap is constructed, cluster labels for all pages in the training data are available. SOUrCe expands URL lists with key  $(page, Apath)$  to key  $(cluster, Apath)$  by concatenating URL lists from pages in the same cluster:

$$U_{C_i, x} = \bigcup_{p \in C_i; C_i \in C} U_{p, x} \quad (4)$$

where  $\bigcup$  represents the concatenation operation for URL lists.

It is guaranteed that at least one page with a cluster label exists in  $U_{C_i, x}$ . Those pages with cluster labels are used to estimate the conditional transition probability of a given  $(cluster, Apath)$  pair  $(C_i, x)$  as follows:

$$P(C_j | C_i, x) = \frac{|d(u) \in C_j \text{ and } u \in U_{C_i, x}|}{\sum_{C_k \in C} |d(u) \in C_k \text{ and } u \in U_{C_i, x}|} \quad (5)$$

where  $d(u) \in C_j$  indicates the destination page of URL  $u$  is in the training data and clustered to  $C_j$  in the sitemap. The adjacency matrix  $A$  at the cluster level is estimated as:

$$A_{ij} = \sum_{x \in C_i} P(C_j | C_i, x) \cdot |U_{C_i, x}| \quad (6)$$

#### 4.4 Online Crawling Scenarios

We consider two common crawling scenarios, crawling for user-created content (*crawling-for-UCC*) and crawling for a target page type (*crawling-for-target*) to show how a structure-oriented sitemap supports different social media crawling strategies.

**4.4.1 Crawling-for-UCC.** Typically when crawling social media, some types of pages are considered less interesting or useful. For example, near-duplicates and non-user-created pages are less valuable than pages shared by users. The goal is to guide the crawler to only follow links that lead to pages with user-created contents (UCC) as their main components. To achieve this goal, we propose a weighting scheme for an uncrawled URL  $u$  extracted from page  $p$  based on three terms, namely informativeness score, similarity score and balance score.

**Info( $C_i$ ):** We first follow the idea of HITS [16], to calculate the authority and hub scores for clusters in sitemap with the help of

adjacency matrix  $A$  as follows:

$$h^k = A a^{(k-1)} \quad (7)$$

$$a^k = A^T h^k \quad (8)$$

where hub score  $h^k = (h_1^k, \dots, h_n^k)^T$  and  $h_i^k$  indicates the hub score for cluster  $C_i$  at the  $k$ th iteration. The authority score  $a^k$  is defined similarly as  $h^k$ . We iteratively compute hub and authority scores until convergence. Then we define the informativeness of a cluster  $C_i$  as follows:

$$\text{Info}(C_i) = \alpha \cdot a_i + (1 - \alpha) \cdot h_i \quad (9)$$

**Dsim( $C_i$ ):**  $\text{Info}(C_i)$  only depicts importance measured by link structures in the sitemap. Some constant pages can receive high authority scores but do not contain UCC, such as policy description pages that are linked by most of the pages in the site. Therefore, we introduce the term  $DSim(C_i)$  to measure the structural importance of a cluster  $C_i$ . Clusters that contain UCC are more diverse in structures, because users are usually free to add or delete elements on those pages. In contrast, clusters that represent constant pages or duplicate pages are more likely to be similar within themselves.  $DSim(C_i)$  denotes the variance of pairwise distance within one cluster, which equals to the average distance to its cluster centroid  $M_i$  as follows:

$$DSim(C_i) = \frac{\sum_{V_j \in C_i} \|V_j - M_i\|_2}{|C_i|} \quad (10)$$

where  $\|\cdot\|_2$  denotes L2 norm of a vector.

**Balance( $C_i$ ):** This term increases crawling diversity by penalizing too much focus on one cluster.  $\text{Balance}(C_i) = 1 - r(C_i)$ , where  $r(C_i)$  is the current ratio of pages in cluster  $C_i$  to total number of crawled pages.

**Score( $u$ ):** The total ordering score  $\text{Score}(C_i)$  for cluster  $C_i$  is summarized as follows:

$$\text{Score}(C_i) = \text{Info}(C_i) \cdot DSim(C_i) \cdot \text{Balance}(C_i) \quad (11)$$

The score of an URL  $u$  from page  $p$  is estimated as follows:

$$\text{Score}(u) = \sum_{C_i \in C} P(C_i | C(p), x_u) \cdot \text{Score}(C_i) \quad (12)$$

The conditional probability  $P(C_i | C(p), x_u)$  can be derived from the navigation table as described in Section 4.3, after  $p$  is classified into  $C(p)$ . During crawling, the uncrawled URL with the highest score in the queue is selected next.

**4.4.2 Crawling-for-target.** Often it is desirable to focus crawling on pages that match the page type of an example page provided by some other process (a *target* page type). For example, the goal may be to crawl all user profile pages on Quora without crawling question pages or other non-UCC pages. In this case, SOUrCe only focuses on crawling target pages or pages that can provide links to target pages. This scenario can be understood as a special case of crawling-for-UCC. Suppose the example page is classified into cluster  $C_i$  in the sitemap. Then instead of using Equation (8) to update authority score, we set  $a_i = 1$  and all other authority scores equal to 0 at each iteration. Updating hub scores still follows Equation (7). In this case, we set  $\text{Score}(C_i) = \text{Info}(C_i)$ , without considering diversity and intra-cluster variance.

**Table 1: Social media sites used in our experiments.**

Name	Description
forums.asp.net (ASP)	Official Forum for ASP.NET
youtube.com (YouTube)	Video Sharing Website
movie.douban.com (Douban)	Chinese Movie Community
voice.hupu.com (Hupu)	Chinese Sports Forum
rottentomatoes.com (Rott)	Movie Rating Community
stackexchange.com (Stack)	Q&A Website for Programming

**Table 2: SOUrCe clustering statistics for 6 datasets.  $|D| = 1,000$  pages per dataset.**

Site	Xpaths	Page Types (Annotated)	Page Clusters (DBSCAN)	Outlier Pages (Annotated)	Outlier Pages (DBSCAN)
ASP	194	9	12	5	13
YouTube	446	4	12	18	13
Douban	408	22	28	30	15
Hupu	250	7	11	34	19
Rott	414	15	20	31	27
Stack	416	15	20	24	55

## 5 EXPERIMENTS

This section describes our experimental methodology and the experiments used to evaluate the SOUrCe crawler.

### 5.1 Datasets

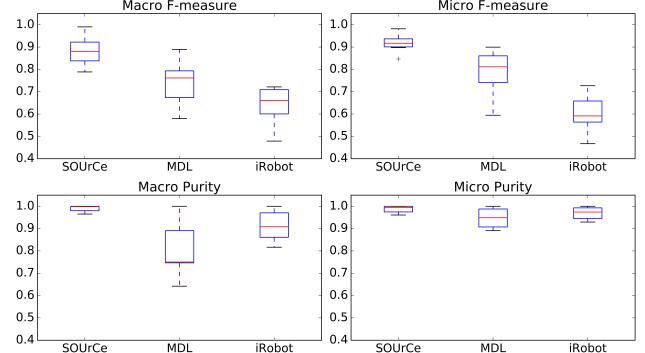
To the best of our knowledge, there are no benchmark datasets for our task, thus we collected data from six social media sites to investigate our ideas. Table 1 briefly describes the datasets. Although URL patterns are sometimes not stable, we chose datasets that do have reliable URL patterns to simplify quantitative evaluation.

**Page Type Annotation:** The ground truth labels for page types (Table 2) were constructed in two stages. Firstly, for each site, we randomly sampled 1,000 pages from its large collection ( $> 20,000$  pages) crawled by BFS. Then we manually clustered pages based on URL patterns, which are regular expressions. Secondly, we compared pages of different URL patterns in a browser and combined patterns that denote the same page type together. After combining, pages denoted by URL patterns with frequency less than  $Min_{Pts}$  were labeled as annotation outliers.

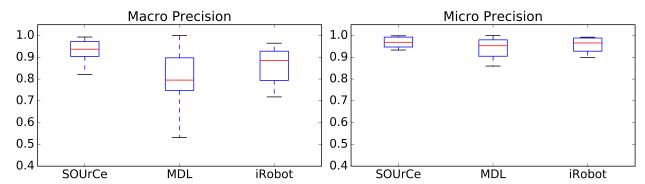
When crawling-for-target (Section 5.4), the URL patterns associated with the example page provide the ground truth labels used to measure the crawler's accuracy.

**User-created Content Annotation:** To assess the accuracy of crawling-for-UCC (Section 5.3), we manually annotated clusters as containing user-created content (UCC) or not. User-created content clusters have pages that are mainly content generated by users; policy description pages and login redirection pages are not UCC.

**Path Filtering:** According to previous findings [12, 19], filtering out paths with low document frequency will help to increase the accuracy in clustering documents. Therefore, we filtered out low-df Xpaths by setting a minimum frequency threshold  $\theta = min_{Pts}$  which is kept constant for every dataset. The number of Xpaths after filtering for each website can be seen in Table 2.



(a) Purity and F-measure on clustering training data.



(b) Precision of classifying pages with 4-fold cross validation

**Figure 5: Evaluations of web pages clustering on training data  $D$  over 6 datasets for two experiments.**

### 5.2 Page Clustering

Page clustering is a crucial part of the SOUrCe crawler. When pages are classified into the wrong clusters, the destination distribution is noisy. Therefore, we conducted two experiments to evaluate the accuracy of the clustering algorithm. The first evaluated the clustering solutions for training data  $D$ . The second tested classification accuracy with 4-fold stratified cross validation.

As for baseline methods, we reimplemented method proposed by Crescenzi et al. work (MDL) [8] and repetitive region representation clustering in iRobot [4]. The threshold  $dt$  for MDL method is set to be 0.2, following authors' suggestion [8]. Three parameters for iRobot [4] are chosen by a parameter sweep, where  $Min_{Depth} = 3$ ,  $Min_R = 0.3$  and  $Min_{Epsilon} = 0.5$ .

We utilize the well-known Purity and F-measure metrics to evaluate the clustering solution on training data  $D$  for the clustering experiment. We use both micro-averaging (weighted by cluster size) and macro-averaging (equal weighting) for the F-measure, to learn about behavior on large and small clusters.

As for classifying test pages into corresponding clusters, we define the Precision metric for evaluation. It first maps a page cluster  $C_i$  with a page type  $T_j$  and examines whether a test page is correctly classified. Different from Purity, which maps between  $C^{\text{test}}$  and  $T^{\text{test}}$ , Precision uses training set  $C^{\text{train}}$  and  $T^{\text{train}}$  to do the mapping. For example, the macro- and micro- Precision are defined as:

$$\begin{aligned} Macro-Prec(C^{\text{test}}, T^{\text{test}}) &= \frac{1}{|C^{\text{test}}|} \sum_i \frac{|C_i^{\text{test}} \cap T_j^{\text{test}}|}{|C_i^{\text{test}}|} \\ Micro-Prec(C^{\text{test}}, T^{\text{test}}) &= \frac{\sum_i |C_i^{\text{test}} \cap T_j^{\text{test}}|}{N^{\text{test}}} \end{aligned}$$

where  $N^{\text{test}}$  is the total number of pages in test data and  $j = \operatorname{argmax}_j |C_i^{\text{train}} \cap T_j^{\text{train}}|$ .

In this part, annotation outliers are not included for all methods and outliers created by clustering algorithms are considered as singular clusters. Clustering statistics for all datasets can be seen in Table 2.

Figures 5(a) and 5(b) show the results for clustering and 4-fold classification experiments. SOUrCe's DBSCAN clustering method performs significantly better than baseline methods in both experiments. MDL prefers to merge small clusters into an existing large one, which reduces its Purity greatly. As for iRobot, the features generated by repetitive regions are usually subsets of our bag-of-Xpaths model. Since the number of features is not very large, iRobot clustering is very likely to generate very small clusters, reducing its F-measure. Also, since the threshold parameter is constant across different datasets, MDL and iRobot have larger variances than SOUrCe. Macro-averaging emphasizes the accuracy on small clusters. SOUrCe's DBSCAN achieves better macro-averaged F-measure values than the two baselines.

### 5.3 Crawling-for-UCC

The second experiment investigated the crawler's ability to collect user-created content (UCC). Typically user-created content is contained in the most frequent page types, which are expected to correspond to the largest clusters.

Our social media websites are too large for us to crawl and mirror entirely. Therefore we follow a previous experimental setup [4], by using breadth-first search (BFS) to collect the first 20,000 pages of the site. Then we offer crawlers a budget  $B$  to crawl at most  $B < 20,000$  pages within this small mirrored corpus.  $B$  is set to be 5000. Two baselines, BFS and iRobot are adopted. We re-implemented iRobot in a completely automatic environment, without human supervision. Specifically, in building the traversal path for iRobot, we skip the step where people examine all generated paths and then remove redundant and add missing paths. Although this is not ideal for iRobot, human effort is limited in practical crawling situations, and iRobot is compared to a fully-automatic crawler (SOUrCe). For SOUrCe,  $\alpha$  in Equation 9 is set to be 0.5, which balances the importance of hub and authority scores, following the setting of Liu, et al. [20].

We evaluate the quality of crawled pages using Valid Ratio and Recall, defined as follows:

$$\begin{aligned} \text{Valid Ratio (VR)} &= \frac{\#(\text{Crawled UCC})}{\#(\text{All Crawled})} \\ \text{Recall} &= \frac{\#(\text{Crawled UCC})}{\#(\text{All UCC})} \end{aligned}$$

We modify F-measure by replacing Precision with valid ratio. We also measure the diversity of crawled results by Entropy. A higher Entropy indicates a more diverse result.

$$\text{Entropy} = - \sum_t p_t \cdot \log_2 p_t \quad (13)$$

where  $p_t$  is the proportion of page type  $t$  in the result.

Results in Table 3 show that SOUrCe can achieve 0.9989 Valid Ratio and 0.2767 Recall on average, which significantly outperforms two baselines. High Valid Ratio indicates that the crawler is able

to stay focused on user-created content. SOUrCe has a high Recall because it ranks all discovered URLs instead of filtering out unimportant URLs.

The variance for BFS is very large. For ASP, login failure pages account for 20 percent of the corpus, however for YouTube the Valid Ratio is 0.997.

Although iRobot produces good results with human supervision, it performs less well in an automatic environment. Its overall Valid Ratio is better than BFS but inferior to SOUrCe. The reason for lower Valid Ratios on some datasets is that iRobot adopts a filtering method, which occasionally makes mistakes. In an automatic environment, mistakenly kept invalid nodes in the traversal path are not removed manually. Similarly, without adding missing paths to the traversal path, iRobot's Recall is greatly affected. We also found that its page-flipping detection method requires a more complicated vocabulary for social media sites; simple anchor texts such as 'Next', 'Previous', or digits are not always sufficient.

### 5.4 Crawling-for-target

The third experiment investigated the crawler's ability to focus on a particular (*target*) page type, exemplified by an example page. Detail and profile pages are usually the most common and informative types of pages [28]; thus, we focus on crawling these two types of web pages for all but one site. Douban is an exception because it requires authorization to retrieve user profiles; therefore, on this site we crawl movie and review pages instead.

We used the same BFS-mirrored corpora used in the previous experiment.

The total numbers of target pages in each corpus varies greatly, thus it is inappropriate to assign a fixed budget for different targets; for some (corpus, target) pairs, a fixed budget would be too generous, while for others it would be inadequate. For example, our YouTube corpus had 14,580 video pages, but our Stack corpus had only 3,854 profile pages. Thus, we used URL patterns to find #(All Target), the total number of target pages in the corpus. The crawling budget was set to be #(All Target). The example target page was selected randomly from the desired page type in training data.

$\alpha$  in Equation 9 is set to be 0.8, because this task is focused more on finding targets than hubs.

The baseline methods were BFS and Target Page Map (TPM) proposed by Vidal, et al. [25]. We modified TPM to support jumping to the target URL pattern on each level to make this baseline more effective. For example, for the original pattern list  $\{p1, p2, p3, p4\}$ , which only matches  $p4$  in  $p3$ , we also allow URLs in patterns  $p1, p2$  and  $p4$  to match  $p4$ . This modification helps to improve the performance of TPM.

In this crawling scenario, with its larger crawling budgets and tight focus on a particular type of pages, it is more likely that a crawler will reach the perimeter of our 20,000 page mirrored corpus and try to download a page that we do not have. We do not wish to penalize a crawler for selecting a hub page that links to pages outside of the corpus. Thus, we give this information to crawlers before crawling. For SOUrCe, the hub scores of those pages are preset to zero. For TPM, those pages will not be selected as intermediate results to retrieve target patterns.

**Table 3: Evaluations on the crawling-for-UCC task. Best results in each metric are marked bold.**

Site	Valid Ratio			Recall			F-measure		
	SOUrCe	iRobot	BFS	SOUrCe	iRobot	BFS	SOUrCe	iRobot	BFS
ASP	0.9988	0.7419	0.8566	0.3187	0.0466	0.2733	0.4833	0.0876	0.4144
YouTube	0.9992	0.9977	0.997	0.2503	0.0218	0.2497	0.4003	0.0427	0.3994
Douban	0.9996	0.8994	0.6064	0.2907	0.0437	0.1764	0.4504	0.0833	0.2732
Hupu	0.9978	1.000	0.9924	0.2501	0.0864	0.2488	0.400	0.1590	0.3978
Rott	0.9998	0.9796	0.982	0.2527	0.0367	0.2482	0.4035	0.07029	0.3963
Stack	0.9984	0.9979	0.8328	0.2978	0.0277	0.2484	0.4588	0.054	0.3827
Average	<b>0.9989</b>	0.9361	0.8779	<b>0.2767</b>	0.0438	0.2408	<b>0.4327</b>	0.0828	0.3773

**Table 4: Evaluations on the crawling-for-target task. Best results in each metric are marked bold.**

Site	Category	Budget	Harvest Rate			Recall			F-measure		
			SOUrCe	TPM	BFS	SOUrCe	TPM	BFS	SOUrCe	TPM	BFS
ASP	Detail	8314	0.804	0.689	0.416	0.804	0.077	0.416	0.804	0.138	0.416
	Profile	4189	0.846	0.802	0.265	0.846	0.090	0.265	0.846	0.162	0.265
YouTube	Detail	14580	0.952	0.994	0.723	0.952	0.752	0.723	0.952	0.856	0.723
	Profile	4244	0.881	1.000	0.265	0.881	0.368	0.265	0.881	0.538	0.265
Douban	Review	1795	0.772	0.870	0.096	0.772	0.101	0.097	0.772	0.182	0.097
	Movie	2488	0.780	1.000	0.279	0.780	0.536	0.279	0.780	0.698	0.279
Hupu	Detail	15285	0.971	0.995	0.816	0.971	0.136	0.816	0.971	0.239	0.816
	Profile	4541	0.854	0.150	0.146	0.854	0.150	0.146	0.854	0.150	0.146
Rott	Detail	4722	0.645	0.937	0.134	0.645	0.103	0.134	0.645	0.186	0.134
	Profile	8635	0.816	0.975	0.534	0.816	0.425	0.534	0.816	0.592	0.534
Stack	Detail	8578	0.897	0.997	0.393	0.897	0.418	0.393	0.897	0.589	0.393
	Profile	3854	0.642	0.521	0.196	0.642	0.114	0.196	0.642	0.187	0.196
Average			0.822	<b>0.828</b>	0.355	<b>0.822</b>	0.273	0.355	<b>0.822</b>	0.376	0.355

We define Harvest Rate and Recall to evaluate methods on effectiveness and coverage respectively, as follows:

$$\text{Harvest Rate (HR)} = \frac{\#(\text{Crawled Target})}{\#(\text{All Crawled})}$$

$$\text{Recall} = \frac{\#(\text{Crawled Target})}{\#(\text{All Target})}$$

we replace Precision with Harvest Rate to define F-measure.

The results in Table 4 show that SOUrCe achieves significantly better performance in Recall and F-measure than the baselines. As for Harvest Rate, TPM performs similarly with SOUrCe. Since TPM is a rule-based method, it sometimes runs out of patterns and stops before reaching the crawling budget. And we also observe that TPM has a low harvest rate for Hupu Profile pages, which is caused by the failure of generating precise URL patterns for target pages automatically.

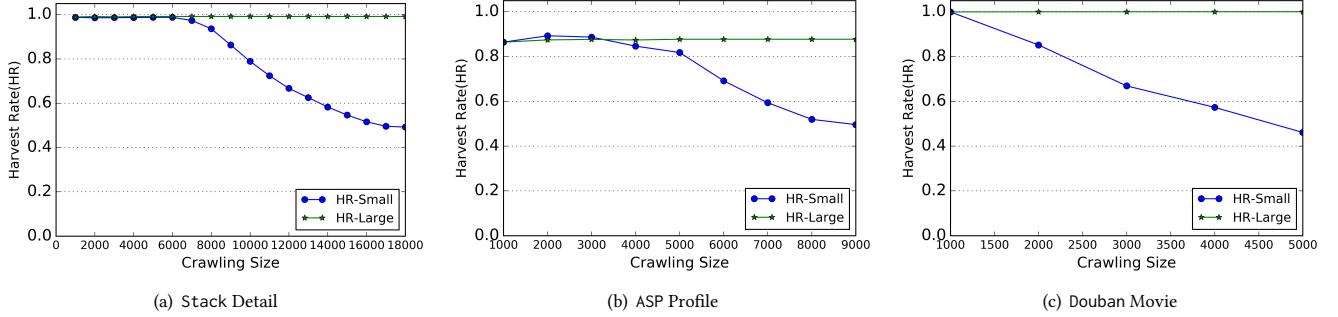
Figure 6 shows some representative examples of the trends of SOUrCe with increasing crawling size. The two lines in Figure 6 represent Harvest Rate in Small corpus (HR-Small) and Harvest Rate in Large corpus (HR-Large). Here small corpus is the first 20,000 pages and large corpus is the unbounded corpus (20,000 pages, plus additional assessments as necessary when the crawler reaches an unjudged page). The trend of HR-Small/HR-Large indicates SOUrCe will first select the most productive paths to follow. At the beginning of Figure 6(a), harvest rate equals to 1. This indicates that SOUrCe follows self-links, such as the related posts section or page-flipping buttons on target pages. Then we observe a drop in HR-Small, which has two causes: (1) SOUrCe exhausts the the most

productive paths and switches to an alternative and less effective path. (2) some newly discovered target pages are already crawled previously, which makes finding new target pages difficult in late phase of crawling. The consistently high HR-Large validates the effectiveness of our method in real crawling scenarios, achieving 0.911 on average.

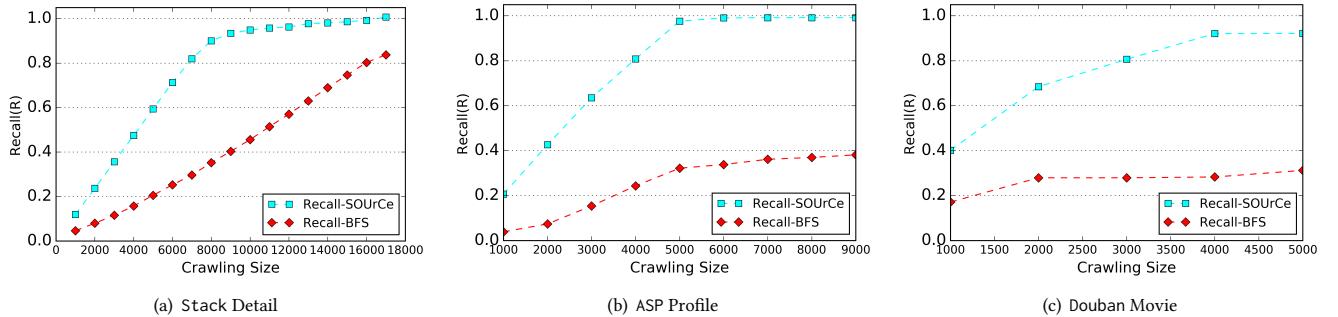
In Figure 7, we compare the performances of our method SOUrCe and BFS on the **Small Corpora**, evaluated by Recall. Two lines represent Recall with BFS (Recall-BFS) and Recall with SOUrCe (Recall-SOUrCe). Comparing two trends, we verify that SOUrCe is able to accumulate target pages at a faster rate than BFS. We report an average Recall-SOUrCe of 0.9498 at the final stage of crawling across 6 datasets. Figure 7(c) shows an example where the final recall for SOUrCe converges to 0.90 instead of 1.0. In this case, some hub pages are classified incorrectly, which caused links from those hub pages to be missed. Note that the intersect point of HR-Small and Recall-SOUrCe is where crawling size equals to #(All Target), and the results of which are reported in Table 4.

## 5.5 Effect of Training Data

Next, we explored the influence of different amounts of training data  $D$  (different sized samples of each site) and different sampling methods, for both tasks. We used the same settings and corpora as in Sections 5.3 and 5.4. Two methods were utilized to sample  $D$ : i) the method described in Section 4.1, and ii) BFS. With  $D$  constructed by both methods, we use SOUrCe for crawling tasks. Results are shown in Figure 8.



**Figure 6: A comparison between SOUrCe crawling-for-target with increasing crawling size on three corpora. “Small” indicates the small (20,000 pages) mirrored corpora. “Large” is the unbounded corpora.**



**Figure 7: A comparison between SOUrCe and BFS when crawling-for-target with increasing crawl sizes on the small mirrored corpora (20,000 pages).**

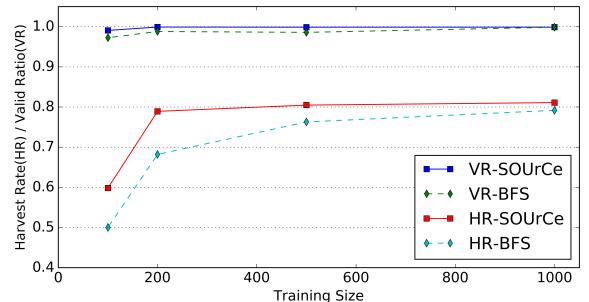
**Table 5: Average Valid Ratio and Entropy for different combinations of features in Equation 11. A higher Entropy score indicates a more diverse crawled result.**

Method	Valid Ratio	Entropy
Dsim+Balance	0.9892	<b>1.539</b>
Info+Balance	0.9896	1.1413
Info+DSim	0.9936	0.4949
Info+DSim+Balance	<b>0.9954</b>	0.8649

We find that the influence of the amount of training data is small in the range 200 to 1,000 pages. We observe a sharp drop from 200 to 100 for the crawling-for-target task. When there is too little training data, the  $K$ -dist histogram for clustering is sparse, which reduces accuracy. However, the results are still good with 100 pages for the crawling-for-UCC task, which is consistent with prior research [4]. Since the crawling-for-target task requires more precise clustering results, the performance drops faster than that of the crawl-for-UCC task. The comparisons between different sampling methods show our method is always more efficient than BFS. On average, 200 pages for our method performs similarly with 1,000 pages sampled by BFS.

## 5.6 Feature Importance

We explored the importance of each term in Equation 11 by conducting an experiment in which each different term is omitted while crawling-for-UCC. The corpus for this experiment is the entire website, instead of the first 20,000 pages, because we want to



**Figure 8: Valid Ratio (VR) and Harvest Rate (HR) trends for different amounts of the training data  $D$ . Solid lines indicate our sampling method SOUrCe and dotted lines represent BFS sampling for  $D$ .**

explore how each feature performs for both effectiveness and diversity in a real crawling environment. The diversity is measured by Entropy defined in Equation 13. We used URL patterns and manual annotation to label these additional pages.

Table 5 shows the results. Experimental results are averaged over 6 datasets. We observe that both  $Info(C_i)$  and  $Dsim(C_i)$  are effective at keeping a high Valid Ratio. Incorporating  $Balance(C_i)$  does increase the diversity of results. We also observe that combining  $Info(C_i)$  and  $Dsim(C_i)$  together reduces the diversity of results, compared with using them separately. Clusters that have high informative scores usually also have high similarity scores, such as the question pages in Q&A sites, which have diverse structure;

multiplying the two terms increases the preference towards those top clusters. Thus we observe a decrease in Entropy.

### 5.7 Scalability

SOUrCe has the usual crawling costs of downloading pages, parsing pages, extracting links, and maintaining a crawling queue. It also has some additional costs that other crawlers do not have; however, they are not onerous costs.

In the learning phase, the new cost involves clustering web pages represented by bags-of-Xpath vectors and generating navigation tables. The pages are obtained by breadth-first search, which is typical for undirected crawlers. Section 5.5 shows that accurate sitemaps can be developed using hundreds of training examples.

Once learning is finished, SOUrCe becomes a very efficient directed crawler. The new cost involves representing a web page as a bag-of-Xpaths, and using the sample data to perform kNN classification to order the queue. The improved crawler efficiency far outweighs this additional cost. Future research might consider how to prune the sample data to improve classification efficiency.

## 6 CONCLUSION AND FUTURE WORK

This paper presents SOUrCe, which is a structure-oriented unsupervised crawler for social media sites. It consists of an unsupervised learning phase and online crawling phase for each site. In the learning phase, it efficiently samples training data and clusters pages using layout structures. Experiments show that our method can better capture structural information using the bag-of-Xpaths model and the clustering algorithm is well designed to handle noise, which provides cleaner data for generating the navigation table. The navigation table measures link structures between different clusters and is effective in sorting the crawling queue to support different types of tasks.

The SOUrCe crawler architecture was evaluated under two common crawling scenarios. In tasks that aim to crawl UCC pages, the weighting scheme is very effective in measuring the importance of page clusters and achieves promising Valid Ratio and Recall. SOUrCe can also be modified to crawl target pages with one target example and achieves significantly better F-measure and Recall than rule-based baselines. SOUrCe is able to automatically detect and follow the most productive paths to find targets, especially self-links within target clusters. The high recall indicates that it finds most of the useful paths through a site. Perhaps most important, SOUrCe achieves its improved crawling accuracy using only small amounts of unsupervised learning that would be practical in large-scale and is an operational crawler.

## ACKNOWLEDGEMENT

This research was sponsored by National Science Foundation grant IIS-1450545. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the author(s), and do not necessarily reflect those of the sponsor.

## REFERENCES

- [1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003*, pages 280–290, 2003.
- [2] M. H. Alam, J. Ha, and S. Lee. Novel approaches to crawling important pages early. *Knowl. Inf. Syst.*, 33(3):707–734, 2012.
- [3] R. A. Baeza-Yates, C. Castillo, M. Marín, and M. A. Rodríguez. Crawling a country: better strategies than breadth-first for web page ordering. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005 - Special interest tracks and posters*, pages 864–872, 2005.
- [4] R. Cai, J. Yang, W. Lai, Y. Wang, and L. Zhang. irobot: an intelligent crawler for web forums. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*, pages 447–456, 2008.
- [5] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through URL ordering. *Computer Networks*, 30(1-7):161–172, 1998.
- [6] J. Cho and U. Schonfeld. RankMass crawler: A crawler with high pagerank coverage guarantee. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 375–386, 2007.
- [7] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *9th International Conference on Tools with Artificial Intelligence, ICTAI '97*, pages 558–567, 1997.
- [8] V. Crescenzi, P. Merialdo, and P. Missier. Clustering web pages based on their structure. *Data Knowl. Eng.*, 54(3):279–299, 2005.
- [9] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *Computer Networks*, 31(11-16):1467–1479, 1999.
- [10] E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An optimal decomposition algorithm for tree edit distance. In *Proceedings of Automata, Languages and Programming, 34th International Colloquium, ICALP 2007*, pages 146–157, 2007.
- [11] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.
- [12] T. Grigalidis and A. Cenys. Using xpairs of inbound links to cluster template-generated web pages. *Comput. Sci. Inf. Syst.*, 11(1):111–131, 2014.
- [13] Y. Guo, K. Li, K. Zhang, and G. Zhang. Board forum crawling: A web crawling method for web forum. In *2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006)*, pages 745–748, 2006.
- [14] I. Hernández, C. R. Rivero, D. Ruiz, and R. Corchuelo. A statistical approach to url-based web page clustering. In *Proceedings of the 21st World Wide Web Conference, WWW 2012*, pages 525–526, 2012.
- [15] J. Jiang, X. Song, N. Yu, and C. Lin. Focus: Learning to crawl web forums. *IEEE Trans. Knowl. Data Eng.*, 25(6):1293–1306, 2013.
- [16] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
- [17] H. S. Koppula, K. P. Leela, A. Agarwal, K. P. Chitrapura, S. Garg, and A. Sasturkar. Learning URL patterns for webpage de-duplication. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010*, pages 381–390, 2010.
- [18] T. Lei, R. Cai, J. Yang, Y. Ke, X. Fan, and L. Zhang. A pattern tree-based approach to learning URL normalization rules. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010*, pages 611–620, 2010.
- [19] H. Leung, K. F. Chung, S. C. Chan, and R. W. P. Luk. XML document clustering using common xpath. In *2005 International Workshop on Challenges in Web Information Retrieval and Integration (WIRI 2005)*, pages 91–96, 2005.
- [20] M. Liu, R. Cai, M. Zhang, and L. Zhang. User browsing behavior-driven web crawling. In *Proc. of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011*, pages 87–92, 2011.
- [21] A. Nierman and H. V. Jagadish. Evaluating structural similarity in XML documents. In *WebDB*, pages 61–66, 2002.
- [22] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.
- [23] A. Selamat and S. Omata. Web page feature selection and classification using neural networks. *Inf. Sci.*, 158:69–88, 2004.
- [24] S. Staab and A. Hotho. Ontology-based text document clustering. In *Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'03 Conference*, pages 451–452, 2003.
- [25] M. L. A. Vidal, A. S. da Silva, E. S. de Moura, and J. M. B. Cavalcanti. Structure-driven crawler generation by example. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006*, pages 292–299, 2006.
- [26] K. Xu, Z. Liu, and J. Callan. De-duping urls with sequence-to-sequence neural networks. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.
- [27] 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 2005*, 2005, pages 76–85, 2005.
- [28] Z. Zhang and O. Nasraoui. Profile-based focused crawler for social media-sharing websites. In *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), Volume 1*, pages 317–324, 2008.
- [29] S. Zheng, R. Song, J. Wen, and D. Wu. Joint optimization of wrapper generation and template detection. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 894–902, 2007.