A Survey about Algorithms Utilized by Focused Web Crawler

Yongbin Yu, Shilei Huang, Nyima Tashi, Huan Zhang, Fei Lei, Linyang Wu

Abstract—Focused crawlers (also known as subject-oriented crawler), as the core part of vertical search engine, collect topic-specific web pages as many as they can to form a subject-oriented corpus for the latter data analyzing or user querying. This paper demonstrates the popular algorithms utilized at the process of focused web crawling, basically referring to webpage analyzing algorithms and crawling strategies (prioritize the URLs in the queue). Advantages and disadvantages of three crawling strategies are shown in the first experiment, where indicates the best-first search with an appropriate heuristics is a smart choice for topic-oriented crawling while depth-first search is helpless in focused crawling. Besides, another experiment on comparison of improved ones (with a webpage analyzing algorithm added) is carried out to verify crawling strategies alone are not quite efficient for focused crawling and in most cases their mutual efforts are taken into consideration. In light of the experiment results and recent researches, some points on the research tendency of focused crawler algorithms are suggested.

Index Terms—Focused crawler, webpage analyzing, crawling strategies, URL prioritizing; harvest rate

1. Introduction

With the skyrocketing of information and explosion of web pages in World Wide Web, it becomes much harder and inconvenient for “advanced users” to retrieve relevant information. In other words, it leads to a big challenge for us, about how to distill the miscellaneous web resources and get the results we really want. On account of the limitations of general-purpose search engine, such as lacking in precision, leading to enormous “topic-irrelevant” information, or unable to deal with complicated information needs of certain individuals or organizations, increasing users choose vertical search engine for specialized, high-quality and up-to-date results. As for vertical search engine, focused crawler [1] is a core part. In contrast to a general-purpose spider which crawls almost every web page and stores them into database, a focused crawler tries to download as many pages pertinent to the predefined topic as it can, and keep the irrelevant ones downloaded to a minimum at the same time.

Before starting scraping, both kinds of crawlers are given a set of seed URLs as their start points. For focused crawlers, increasing the precision of fetching topic-specific webpages significantly makes the seeds choosing of paramount importance. Selection of appropriate seed URLs relies on external services which usually are general web search engines like Google and Yahoo, and several attempts have already been made to improve the harvest rate (percentage of relevant pages retrieved) by utilizing search engines as a source of seed URLs [2, 3]. The approach of seeds selecting is beyond the scope of this survey but out of its importance, we still bring it up.

The complete process of crawling is modeled as a directed graph (no loop with duplicate URL checking algorithms applied and not taking into account of the URLs downloading sequence), each webpage is represented by a node and a link from one page to another is presented by an edge. Furthermore, the whole seeds with their each offspring pages form a forest and each seed page becomes the root of a single tree of the forest, as Fig. 1. The spider uses a certain crawler algorithm to traverse the whole graph (forest).

![Fig. 1. Web crawling forest.](http://www.journal.uestc.edu.cn)
At the beginning of crawling, the seed URL (seed page) is sent to URLs manager. Then, the downloader fetches the URL and gets it downloaded after the manager ensures there exists a new URL. The content of the webpage is fed to a specifically designed parser which distills the irregular data to get new URLs we truly want queued in the URLs manager. Webpages or parsed data are stored in a container which can be a local file system or database, depending on the requirements of crawler design. The process repeats ad infinitum while the focused crawler traverses the web until the URLs manager is empty or some certain preset stop conditions are fulfilled. The simplified crawler workflow is shown in Fig. 2.

Fig. 2. Simplified crawler workflow

For a well-designed subject-oriented crawler, it’s necessarily equipped with appropriate algorithms for URLs manager and parser, crawling strategies and webpage analyzing algorithms respectively [4, 5]. Crawling strategies determine the downloaded sequence of the URLs, namely prioritizing the URLs, which mostly relies on the evaluation of relatedness to a predefined subject. As for the parser, it’s designed to extract topic-relevant URLs, dump irrelevant ones and evaluate the pertinence of the webpage to the subject. The two components are not mutually exclusive but cooperate cheek by jowl with each other to address the key challenge of focused crawling, identifying the next most relevant link to fetch from the URLs queue.

The contributions of this paper are as follows:

1) Provide a comprehensive review to webpage analyzing algorithms and crawling strategies.
2) The close relationships between the two types of algorithms are stated in detail.
3) Experiments are carried out to prove that the combination of the two types of algorithms tremendously improves the efficiency of focused crawler.

The rest of this paper is organized as follows. Related work in recent years will be introduced in section 2. Section 3 and section 4 provide overviews on webpage analyzing algorithms and crawling strategies respectively. Experimental results and comparisons as well as some thoughts are shown in section 5. Finally, concluding remarks are given in Section 6.

2. Related Work

This section provides a quick review of recent studies in focused crawler, from which the future research tendency are exposed.

To alleviate the problems of local searching algorithms (like depth-first and breadth-first), genetic algorithm is proposed to improve the quality of searching results in focused crawling [5]. In [6], the results show GA can effectively prevent the search agent from being trapped in local optimal and it also significantly improve the quality of search results. As a global searching algorithm, GA allows crawler to find pertinent pages without any distance limit and it does not introduce noise into the pages collections. GA is an adaptive and heuristic method for solving optimization problems. In Yohanes's paper, heuristics refers to effective combination of content-based and link-based page analysis. Their results show GA-crawler can traverse the web search space more comprehensively than traditional focused crawler. In Yu’s research [7], GA is used to acquire optimal combination of full text and anchor text instead of linear combination which may lacks objectivity and authenticity. Apart from GA, a novel focused clawer which applies cell-like membrane computing optimization algorithm (CMCFC) is proposed in Liu’s work [8]. Similarly, the purpose of this algorithm is to overcome the weakness of empirical linear combination of weighted factors.

A two-stage effective deep web harvesting framework, namely SmartCrawler is proposed in [9] to explore more web resources which are beyond the reach of search engine. The framework encompasses two stages. In the first stage, SmartCrawler performs site-based searching for the most relevant web site for a given topic with the help of search engines. In the second stage, it achieves fast in-site searching by excavating the most pertinent links with an adaptive link-ranking. The studies show the approach achieves wide coverage for deep net interfaces and maintains extremely efficient locomotion. In contrast to other crawlers, it gets a higher harvest rate.

Ontology can be used to represent the knowledge underlying topics and web documents [10]. Semantic focused crawlers are able to download relevant pages precisely and efficiently by automatically understanding the semantics underlying the web information and predefined topic. The uncertain quality of ontology leads to the limitation of ontology-based semantic focused crawlers, so in order to work it out, ontology learning technologies are integrated. With ontology learning, facts and patterns from a corpus of data can be semi-automatically extracted and turned into machine-readable ontology. H. Dong et al. [11] and K. S. Dilip et al. [12] propose self-adaptive focused crawlers based on ontology learning. D. Hai manages to precisely discover, format and index relevant web documents in the uncontrolled web environment, while K. S. Dilip aims at overcoming the issues caused by tagged web assets. Gaur et al. [13] leverage an ontology-based
method to semantically expand the search topic and make it more specific. To find the most relevant webpages which meet the requirements of users, Agre et al. [14] utilize the knowledge path based on ontology tree to get relevant URLs. Zunino et al. [15] present an approach integrated with semantic networks and ontology to build a flexible focused crawler. Some researches [16-20, 50] on focused crawler based on semantics have been published in recent years.

It’s quite normal for one irrelevant webpage to contain several segments pertinent to predefined topic and for one relevant webpage to encompass irrelevant parts. To predict more precisely, webpage is divided into smaller units for further analysis. Seyfi et al., in their works [21, 22], propose a sort of focused crawler based on T-graph to prioritize each unvisited URL with high accuracy. A heuristic-based approach, CBP-SLC (Content Block Partition-Selective Link Context), is proposed in [23], with which highly pertinent regions in an excessive jumble webpage will not be obscured. Similarly, Ganguly et al. design an algorithm which segments the webpages in the light of headings into content blocks and calculates the relevancy of each block separately [24].

To analyze features of documents, they can be represented by Vector Space Model (VSM). The relevancy between documents is measured by their similarity and a basic method used to do correlation analysis is introduced in [25].

With the boom of AI, some researchers tend to study on learning-based focused crawlers. Classifier is a commonly used tool in learning-based focused crawlers just as what are presented in [26, 27]. Askari Torkestani proposes an adaptive algorithm based on learning automata with which the focused crawler can get the most relevant URLs and know how to approach to the potential target documents [28]. Manish Kumar et al. [51] propose to design a database for information retrieval of Indian origin academicians. H-Lu et al. [52] present an approach which uses web page classification and link priority evaluation.

As for this paper, a review work, it should be noted that some references may have been overlooked owing to numerous variants, just like focused crawler based on link and content. In the next two sections, we consecutively review two basic sorts of algorithms, webpage analyzing algorithms and crawling strategies.

3. Webpage Analyzing Algorithms

In this section, PageRank, Site-Rank, VIPS as well as densitometric segmentation, four algorithms will be introduced.

3.1 PageRank
PageRank [3, 29] is proposed by S. Brin and L. Page, founders of Google. The PageRank of a page represents the probability that a random surfer who follows link randomly from one page to another will be on that page at any given time. A page’s score depends recursively upon the scores of the pages that point to it. Source pages distribute their PageRank across all of their out links. However, PageRank based on hyperlink analysis without notion of page content, analyzing the authority of hyperlinks, undoubtedly is an excellent URL ordering metric but this is just useful if we are just looking for hot pages but not topic-specific pages [30].

Based on the fact that the original PageRank algorithm is a topic-independent measure of the importance of webpages, several improvements have been made to guide topic-driven crawlers. In [31], a combination of PageRank and similarity with the topic is used to guide the crawler. An improved PageRank algorithm called T-PageRank is proposed in [32, 33], which is based on “topical random surfer” in contrast to “a random surfer” in PageRank, and for a better performance, it’s combined with the topic similarity of the hyperlink metadata.

3.2 SiteRank
Classical centralized algorithms like PageRank are both time-consuming and costly for the whole web graph. J. Wu and K. Aberer in their work [34] put forward a higher abstraction level of Web graph, viz., SiteGraph instead of DocGraph. They distribute the task of computing page ranking to a set of distributed peers each of which crawls and stores a small fraction of web graph. In complete contrast to setting up a centralized storage, indexing, link analysis system to compute the global PageRank of all documents based on the global Web graph and document link structure, they build a decentralized system whose participating servers compute the global ranking of their locally crawled and stored subset of Web based on the local document link structure and the global SiteRank.

The result turns out to be the fact that the computation of the SiteRank of such a Web-scale SiteGraph is fully tractable in a low-end PC machine. Compared to PageRank, the cost is largely reduced while SiteRank keeps good quality of the ranking results. Additionally, it’s difficult to spam SiteRank for PageRank spammer since they have to set up a quantity of spamming Web sites to take advantage of the spamming SiteLinks.
3.3 VIPS (Vision-Based Page Segmentation)

There are a lot of links in one page, only a few of which have something to do with the predefined topic. PageRank and SiteRank do not distinguish these links, leading to noise like ads links while analyzing the webpages. PageRank and SiteRank are the algorithms of page granularity and site granularity respectively. VIPS a block granularity of webpage algorithm used to compute the relevance and accurately predict the unvisited URLs [35], is proposed [36]. The algorithm spans over three phases, visual block extraction, visual separator detection and content structure construction. These three steps as a whole are regarded as a round. The visited webpage is segmented into several big blocks, and for each big block, the same segmentation process repeats recursively until we get adequately small blocks.

VIPS here is adopted to segment webpages to extract relevant links or content, by means of utilizing visual cues and DOM tree to better partition a page at the semantic level. The algorithm is based on the fact that semantically related content is usually grouped together and the entire page is divided into regions for different contents using explicit or implicit visual separators such as lines, blank areas, images, font sizes, colors, etc. The output of the algorithm is a content structure tree whose nodes depict semantically coherent content [37].

In [36], it diagrams the working process of VIPS in details. It first extracts all the suitable blocks from the html DOM tree, and then it tries to find the separators denote the horizontal and vertical lines in a webpage that usually cross with no blocks. Finally, based on these separators, the semantic structure for the webpage is constructed. The URLs and their anchor text in the pertinent blocks are extracted for calculating the scores. Based on the scores the URLs are ranked in the URLs manager.

3.4 Densitometric Segmentation

According to Kohlschütter, for segmentation problem, only three types exist: segmentation as a visual problem, segmentation as a linguistic problem and segmentation as a densitometric problem [38].

Visual segmentation is the easiest one for common people to understand. A man looks at a webpage and he is able to immediately distinguish one section from another. A computer vision algorithm can act similarly. However, it has to render each webpage and do a pile of things that are computationally quite expensive. The linguistic approach is somewhat more reasonable. Distributions of linguistic units such as words, syllables and sentences have been widely used as statistical measures to identify structural patterns in plain text. The shortage is that it only works for large blocks of text, which means linguistic content in small blocks like header, footer is usually lacking for analysis.

Kohlschütter’s densitometric approach has a tendency to work as well as a visual algorithm, while being as fast as a linguistic approach. The basic process is walking through nodes, and assigning a text density to each node (density is defined as the result of dividing number-of-tokens by number-of-‘lines’). Merge neighbor nodes with the same densities and repeat the process until desired granularity is reached somewhat like VIPS.

What is brilliant is the simplicity of this algorithm and they managed to get the consuming-time down to 15ms per page on average. By contrast, visual segmentation takes 10 seconds to process a single webpage on average.

4. Crawling Strategies

In this section, for crawling strategy, there are three algorithms taken into consideration, breadth-first Search, depth-first Search and best-first Search.

4.1 Depth-First Search Strategy

It’s an algorithm of traversing graph data structures. Generally, when the algorithm starts, there is chosen a root node and next nodes are selected in a way that the algorithm explores as far as possible along the branch. In short, algorithm begins by diving down as quickly as possible to the leaf nodes of the tree. When there is no node for next searching step, it goes back to the previous node to check whether there exists another unvisited node. Taking the left tree in Fig. 1 as an example, first it searches seed page1, then page1.1, page1.1.1 and 1.1.2 getting searched in sequence and finally, page1.2, page1.2.1, page1.2.2 are searched in order.

However, this algorithm might end up in an infinite loop [39], viz., that the crawler may get trapped, leading to that only a very tiny part of tree is explored, with the reality taken into consideration that the web graph we want to traverse is so tremendously enormous that we can consider it as an infinite graph, which means the graph has infinite nodes or paths. A depth limit (the algorithm can only go down a limited levels) is a trial of solving this trap problem, even if it’s not exhaustive. Depth-first alone is not appropriate to focused crawling in which we hope to cover relevant webpages as many as possible whereas depth-first ignores quite a few.

For the experiment purposes, standard version of algorithm was modified in order to apply it to the Web search domain [40]. Instead of selecting the first node there
is selected a list of seed links which is the beginning point of the algorithm. Thereafter, there is chosen first link from the list. It is being evaluated by the algorithm of classification whether it satisfies criteria of specified domain. If it is classified as proper website it lands to the bucket of found links. In other case link is only marked as visited. Then the chosen website is parsed in order to extract new links from a page and all found links are inserted at the beginning of the URLs queue. The queue behaves in this case like LIFO (last in, first out) queue.

4.2 Breadth-First Search Strategy

Same as depth-first, breadth-first is an algorithm of traversing graph data structures as well. Breadth First Search is the simplest form of crawling algorithm, which starts with a link and carries on traversing the connected links (at the same level) without taking into consideration any knowledge about the topic. It’ll go down to the next level after completely going though one level. For example, in Fig. 1, breadth-first behaves like the process that in the left tree, the seed page1 is searched at the beginning, after which page1.1 and page1.2 getting searched in sequence and finally, page1.1.1, page1.1.2, page1.2.1 as well as page1.2.2 are searched in order. Since it does not take into account the relevancy of the path while traversing, it is also known as the Blind Search Algorithm. The reason why breadth-first can be adopted for focused crawler is based on the theory that webpages at the same level or within a certain levels away from the seed URL have a large probability to be topic-relevant [30]. For instance, in Fig. 1, there are 3 levels in total, the offspring of each seed is likely to share the pertinent subject with their own seed page.

When algorithm starts there is chosen a starting node, in the case of focus Web crawling there is chosen a bunch of root URL (seed URLs) and the algorithm explores them in a way to firstly download pages from the same level on the branch. On each page there are extracted external links from the Web pages and all these links lands in the searching tree on the leafs of current node (one level below the current node). When the crawler visits all the URLs from the same level it goes one level deeper and starts the same operation again. So it traverses the tree level by level. The URLs queue behaves in this situation as FIFO (first in first out) queue [40].

4.3 Best-First Search Strategy

Best-first Search algorithm is also a graph based algorithm used for traversing the graph. According to a certain webpage analyzing algorithm, it predicts the similarity between candidate URLs and target webpage or candidates’ relatedness to predefined topic, picking the best one to fetch.

Firstly, a rule about what is “best” should be defined, usually a score or rank. In most cases, a classifier algorithm is applied such as Naive Bayes, Cosine Similarity and SVM as well as string-matching are used for scoring.

Unlike blind search algorithms: depth-first and breadth-first, best-first is an algorithm that uses heuristic methods to improve its results. Heuristics here refers to a general problem-solving rule or set of rules that do not guarantee the best solution, but serves useful information for solving problem. In our case the role of heuristics is to evaluate the website URL, based on the given knowledge before fetching its content. The name best-first refers to the method of exploring the node with the best score first. An evaluation function is used to assign a score to each candidate node. There are many different ways of evaluating the link before fetching the whole content, e.g. evaluating the words in URL, evaluating the anchor texts of the link and more detailed researches in this area are described in [41, 42, 43, 44, 45, 46, 47]. The algorithm maintains two lists, one containing a list of candidates to explore, and one containing a list of visited nodes. Since all unvisited successor nodes of every visited node are included in the list of candidates, the algorithm is not restricted to only exploring successor nodes of the most recently visited node. In other words, the algorithm always chooses the best of all unvisited nodes that have been graphed, rather than being restricted to only a small subset, such as nearest neighbors. Previously stated algorithms depth-first Search and breadth-first Search have this restriction.

5. Experiments and Comparisons

5.1 Comparison of Three Crawling Strategies

In order to compare the efficiency of these three crawling strategies in pertinent topic webpages crawling process, one experiment was conducted. The crawling platform was written in python language and deployed on Windows 10 operating system.

Before starting crawling, we needed to choose appropriate seed URLs and set the standard that one webpage could be classified as relevant webpage. Thus, two keywords lists were created for seeds generation, correlation test as well as link analysis. In this experiment, the main topic was ‘西藏建设成果’ which meant the development achievements of Tibet. One keywords list contained ‘西藏’, ‘拉萨’, ‘日喀则’, ‘青藏’, ‘昌都’, ‘林芝’, ‘山南’, ‘那曲’, ‘阿里’, referring to geographic name that usually appeared in news relevant with Tibet. The other keywords list was based on ‘发展’, ‘建设’, ‘成果’, ‘成绩’, ‘伟业’, ‘创举’, ‘事业’, ‘业绩’, ‘成就’, ‘果实’, ‘硕果’, ‘进展’, which were the synonyms of ‘achievement’ or ‘developing’, or shared the same semantics with ‘achievement’ or ‘developing’.
Famous search engines like Google, Yahoo, Baidu, and Bing were adopted to retrieve high quality seed URLs. After most relevant results were returned by these search engines with topic ‘development achievement of Tibet’ as input, then we selected the most pertinent ones manually to ensure quality, authority.

In relation to the method of judging whether a webpage was relevant or not, two parts were taken in to consideration, the title and the main content, extracted from ‘title’ and ‘p’ elements of the webpage respectively. There were two process doing the estimate. Webpage with at least one geographic keyword in its title and at least three times the keyword in its main content was sent to the subsequent processing or it was dumped. In general, word counts can be a powerful predictor of document relevance and can help distinguish documents that are about a particular subject from those that discuss that subject in passing [48] or those that have nothing to do with the subject. Hence, the minimum three times was utilized to increase the possibility that one webpage was relevant to our topic and at same time, the number three was not so large as to ignore potential pertinent document. Afterwards, the processed webpage whose main content, in which the total number of occurrence of any keyword in the other keywords list is greater than 3, was marked as relevant webpage. We have to admit the relevance-check approach is a quite rough way and not greatly accurate, but it also works in a certain extent in our experiment, since all three algorithms are judged by the same method.

In this experiment, list is taken as data structure for URLs’ scheduling. For simplicity, link evaluating is just grounded on one factor, and anchor text is the only basis of link analysis for best-first search, out of the reason that not only it can be easily fetched from the parsed webpage but also it has two properties that make it particularly useful for ranking web page [49]. First, it tends to be very short, generally several words that often succinctly describe the topic of the linked page. Second, many queries are very similar to anchor text in that they are also short topical descriptions of webpages. The value of the out link in one webpage depends on how many times the keyword in both keywords lists shows up in the anchor text, and if any keyword of both lists occur in the anchor text at the same time, we will add extra 2 to its value. The list was sorted by the value in descending order, and in each crawl the first element was popped out.

Below are shown two graphs, illustrating the comparison of amount of found relevant webpages and frontier (crawler’s request queue) size by three different algorithms. Fig. 3 depicts an amount of relevant webpages compared to crawling times. Fig. 4 presents the size of frontier compared to crawling times.

As it can be seen in Fig. 3, depth-first search hardly gave good results and provided focused crawling nearly with no help. In 200 times crawling, the amount kept unchanged, actually being one, the straight line in the figure. As for breadth-first search, it slowly found the relevant links. In our expectation, best-first search performed the best, for as the first of the three tested algorithms encountering high quality link, viz., high ranking in the sorted list.

From Fig. 4, depth-first stabilized its size of the frontier at a certain level for its low efficiency, and best-first, due to its used URL evaluating rule, had no great disparity with depth-first in frontier size. In the case of breadth-first search, it produced the fastest and the biggest amount of links.

5.2 Comparison of Three Improved Crawling Strategies

The previous subsection shows the performance of three crawling strategies, and in this subsection, we introduce another experiment based on the former one, with a certain webpage analyzing algorithm doing webpage
We choose a block-granularity webpage algorithm, and it utilizes DOM features and visual cues, like VIPS. However, it does page segmentation by visual clustering. The algorithm will be introduced in detail. After downloading the webpage, we extract the useful elements (those most probably contain text information like <p> and <a>) and then, store their XPATH, position (from top and left), width and height. DBSCAN (Density-based spatial clustering of applications with noise), a data clustering algorithm is taken to do the visual clustering. Features fed to the DBSCAN are composed of two parts, visual features as well as DOM features. In this experiment, visual features are represented as a 6-dimensional vector and DOM features are expressed as an n-dimensional vector where n is determined by the amount of distinct path in the XPATH list (XPATH of each element in the webpage is processed to build this list). Value ‘left’, value ‘left + width’, value ‘top’, value ‘top + height’, value ‘(2*left + width)/2’ and value ‘(2*top + height)/2’, the 6 values construct the visual features vector of element. We locate every path of each element’s XPATH in XPATH list, distributing a value to corresponding dimension in its n-dimensional vector, and for other dimensions, they are assigned value zero as default. The two vectors are jointed as feature array. The output of DBSCAN is cluster labels for each element in the dataset. Clusters with one element are dumped, which are regarded as noisy ones. Intuitively and empirically, the header and footer of one webpage offer content and potential useful links extraction no help, and thus we get rid of these two parts from clusters. Below are two pictures, illustrating the segmentation results and final output after disposing of noise, header and footer.

One single color represents one cluster. Clusters in red circles (Fig. 5) have been removed (the middle one is noise), and then we get the final result illustrated in Fig. 6.

The value of queueing links includes two parts. One is gotten from their anchor text that we described at first experiment, and clusters information is taken into consideration for the other. We assume the cluster whose area is largest contains elements that form the main content, which accords to common sense. When the amount of clusters is larger than 5, we assign a value 3 to the largest cluster, a value 2 to the second largest cluster and a value 1 as default to left clusters (‘largest’ here refers to the area of cluster that are composed of area of each single element in it.). While if the amount of clusters is less than 5, we only assign a value 2 to the largest cluster and a value 1 to the left. And the final value of link is:

\[
\text{Value} = 0.9 \times \text{Value(}\text{anchor text}) + 0.1 \times \text{Value(}\text{cluster info}) \tag{1}
\]

Each value has a weight, out of the reason that anchor text conveys direct information about the topic while cluster information just reveals the relations between elements in an individual webpage, and hence 0.9 and 0.1 as weights are assigned to two values respectively.

The second experiment was carried out under the same platform, the same data and the same hardware environment.

As you can see, comparing Fig. 7 with Fig. 3, Fig. 8 with Fig. 4, we can draw such a conclusion that the variation trend keep unchanged. For a better contrast of two experiments, they are put together as what showed below.
What is interesting in Fig. 9 is the phenomenon that the segmentation don’t show any improvements on depth-first and breadth-first, on account of that as blind search, depth-first and breadth-first use no domain knowledge of the problem state, and they work without heuristics. To be more concrete, the estimation information of each queueing link isn’t utilized by depth-first and breadth-first. In this experiment, what the segmentation does for them is only removing links that are not quite relevant to our topic in noise, header and footer, which cannot fundamentally change their low harvest rate in focused crawling. As for best-first, there is a significant improvement. In terms of our experiments, the segmentation by visual clustering increases the harvest rate from 40% to 60%. For our experiments are quite rough, the harvest rate cannot be definitely accurate, but under such a circumstance that two experiments were conducted under the same conditions and judged by the same mechanism, the improvements must mean something. The improvements can be explained by the more accurate out links estimation, which provides a more precise prediction.

In Fig. 10, what is obvious is that after segmentation and useless clusters filtering, the frontier size keep increasing under the line of the first experiment. In the central area of webpage, the links always link to pertinent webpages while links in header and footer probably not. Our topic is quite narrow, so the links of central area tend to link to the same website where the topic is widely talked about and that’s why the frontier size grows slowly and a quantity of irrelevant links lead to skyrocketing of frontier size.

Based on the two experiments, comparisons of the six crawling processes are shown in Table 1.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Number of relevant page</th>
<th>Frontier size</th>
<th>Computational costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algo</td>
<td>Depth-first</td>
<td>Breadth-first</td>
<td>Best-first</td>
</tr>
<tr>
<td></td>
<td>Depth-first with Seg</td>
<td>Breadth-first with Seg</td>
<td>Best-first with Seg</td>
</tr>
<tr>
<td>Number of relevant page</td>
<td>Extremely small</td>
<td>Extremely small</td>
<td>Extremely small</td>
</tr>
<tr>
<td></td>
<td>Extremely small</td>
<td>Extremely large</td>
<td>Extremely small</td>
</tr>
<tr>
<td></td>
<td>Quite small</td>
<td>Quite small</td>
<td>Quite small</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>Large</td>
<td>Large</td>
</tr>
<tr>
<td>Frontier size</td>
<td>Extremely small</td>
<td>Extremely small</td>
<td>Extremely small</td>
</tr>
<tr>
<td></td>
<td>Quite small</td>
<td>Quite small</td>
<td>Quite small</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>Large</td>
<td>Large</td>
</tr>
<tr>
<td>Computational costs</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Quite high</td>
<td>Quite high</td>
<td>Quite high</td>
</tr>
<tr>
<td></td>
<td>Quite high</td>
<td>Quite high</td>
<td>Quite high</td>
</tr>
<tr>
<td></td>
<td>Quite high</td>
<td>Quite high</td>
<td>Quite high</td>
</tr>
<tr>
<td></td>
<td>Quite high</td>
<td>Quite high</td>
<td>Quite high</td>
</tr>
</tbody>
</table>

5.3 Points on Research Tendency

Undoubtedly, our ultimate target is to raise harvest rate and cut the processing time down. Even though harvest rate is the priority, a higher rate cannot be at the cost of time. Visual segmentation, whether VIPS or clustering, takes too much time to process webpage. Therefore, densitometric segmentation is a better choice. However, segmentation is just a side of webpage analyzing, and what we really want is the relevance information on the webpage. So, to extract comprehensive features (not just anchor text, position and size information), in order to make a more reliable prediction and to keep the consuming-time as less
as possible are the priorities of future research. Before any breakthroughs have been made, we have to strike a balance between harvest rate and computational costs.

In light of recent researches, more algorithms included in the rapidly growing area of Artificial Intelligence are integrated with focused crawler to improve its efficiency, which means methodologies in AI are also future directions in focused crawler researching.

6. Conclusion

As stated above, this survey gives a quick review of algorithms commonly utilized in two main parts of focused crawler. The basic crawling strategies alone are not appropriate to topic-driven crawler, webpage analyzing algorithms neither. In reality, the output of webpage analyzing algorithms is fed to crawling strategies, or in others words, the latter is based on the former, aiming to prioritize the URLs in the queue with mutual efforts. And finally the key challenge of focused crawler is worked out, identifying the next most proper link to fetch from the URLs queue. Depending on different needs of applications, we choose corresponding algorithms.

The future target is to improve the two sorts of algorithms and find a better combination, to enhance the crawling efficiency, basically at two aspects, higher harvest rate and lower computational costs.

Acknowledgement

This work is supported by Research Fund for International Young Scientists of National Natural Science Foundation of China (NSFC Grant No.61550110248), and Tibet autonomous region key scientic research projects (Grant No. Z2014A18G2-13). The authors would like to thank the editor and the reviewers for their helpful suggestions and valuable comments.

References


Yong-Bin Yu was born in Sichuan Province, China, in 1975. He received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2008. He is currently an associate professor in the School of Information and Software Engineering, UESTC. His research interests include big data and memristor.

Shi-Lei Huang was born in Chongqing municipality, China, in 1993. He received the B.S degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2016. He is currently pursuing the M.S degree in the College of Computer Science and Technology, Zhejiang University. His research interests include data analysis and natural language processing.

Nyima Tashi was born in Tibet, China, in 1964. He received the Ph.D. degree from the Sichuan University, Chengdu, in 2009. He is currently a professor in the Tibet University. His research interests include computer network and information system.

Huan Zhang was born in Henan Province, China, in 1993. He received the B.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2015. He is currently pursuing the M.S. degree with the School of Information and Software Engineering, UESTC. His research interests include big data and memristor.

Fei Lei was born in Fujian Province, China, in 1992. He received the B.S. degree from the University of Electronic Science and Technology of China (UESTC), Cheng, in 2015. He is currently pursuing the M.S. degree with the School of Information and Software Engineering, UESTC. His research interests include big data and memristor.

Lin-Yang Wu was born in Henan Province, China, in 1993. He received the B.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2016. He is currently pursuing the M.S. degree with the school of electronics and information engineering, XJTU. His research interests include big data and memristor.