A Survey about Algorithms Utilized by Focused Web Crawler

Yong-Bin Yu, Shi-Lei Huang, Nyima Tashi, Huan Zhang, Fei Lei, and Lin-Yang Wu

Abstract—Focused crawlers (also known as subject-oriented crawlers), 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 that the popular algorithms utilized at the process of focused web crawling, basically refer to webpage analyzing algorithms and crawling strategies (prioritize the uniform resource locator (URLs) in the queue). Advantages and disadvantages of three crawling strategies are shown in the first experiment, which indicates that the best-first search with an appropriate heuristics is a smart choice for topic-oriented crawling while the 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 that 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—Crawling strategies, focused crawler, harvest rate, uniform resource locator (URL) prioritizing, webpage analyzing.

1. Introduction

With the skyrocketing of information and explosion of web page in the 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. The limitations of general-purpose search engine, such as lacking in precision, lead to enormous “topic-irrelevant” information problems, or being unable to deal with complicated information needs of certain individuals or organizations. These problems have increased the phenomena that users would like to choose the vertical search engine for specialized, high-quality, and up-to-date results. As for the vertical search engine, the focused crawler is a core part. In contrast to a general-purpose spider which crawls almost every web page and stores them into a 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 uniform resource locators (URLs) as their start points. For focused crawlers, increasing the precision of fetching topic-specific webpage significantly makes the seeds choose 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 (the percentage of relevant pages retrieved) by utilizing search engines as a source of seed URL[s]. The approach of seeds selecting is beyond the scope of this survey but for 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 presented by an edge. Furthermore, all seeds with their each offspring pages form a forest and each seed page becomes the root of a single tree of the forest, as shown in Fig. 1. The spider uses a certain crawler algorithm to traverse the whole graph (forest).

At the beginning of crawling, the seed URL (seed page) is sent to the URLs manager. Then, the downloader fetches the URL and gets it downloaded after the manager ensures that there exists a new URL. The contents of the webpage are 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 is 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. 1. Web crawling forest.

Fig. 2. Simplified crawler workflow.

For a well-designed subject-oriented crawler, it is necessary to be equipped with appropriate algorithms for the URLs manager and parser, crawling strategies, and webpage analyzing algorithms, respectively. 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 is designed to extract topic-relevant URLs and dump irrelevant ones, and evaluate the pertinence of the webpage to the subject. The two components are not mutually exclusive but cooperate closely 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 the focused crawler.

The rest of this paper is organized as follows. Related work in recent years is introduced in Section 2. Sections 3 and 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 crawlers, from which the future research tendency is exposed.

To alleviate the problems of local searching algorithms (like depth-first and breadth-first), the genetic algorithm (GA) is proposed to improve the quality of searching results in focused crawling. In [6], the results showed GA can effectively prevent the search agent from being trapped in local optimal and it also can significantly improve the quality of search results. As a global searching algorithm, GA allows the 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 B. W. Yohanes et al.’s paper, heuristics referred 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 C. Yu et al.’s research, GA was used to acquire optimal combination of the full text and anchor text instead of the linear combination which may lack objectivity and authenticity. Apart from GA, a focused clawer which applied the cell-like membrane computing optimization algorithm (CMCFC) was proposed in W. Liu and Y. Du’s work. 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, was 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. 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. and K. S. Dilip et al. proposed self-adaptive focused crawlers based on ontology learning. H. Dong et al. managed to precisely discover, format, and index relevant web documents in the uncontrolled web.
environment, while K. S. Dilip et al. aimed at overcoming the issues caused by tagged web assets. R. Gaur et al.\cite{13} leveraged an ontology-based method to semantically expand the search topic and make it more specific. To find the most relevant webpage, which meet, the requirements of users, G. H. Agre et al.\cite{34} utilized the knowledge path based on the ontology tree to get relevant URLs. R. Zunino et al.\cite{15} presented an approach integrated with the semantic networks and ontology to build a flexible focused crawler. Some researches\cite{29,25} on focused crawler based on the semantics have been published in recent years.

It is quite normal for one irrelevant webpage to contain several segments pertinent to the predefined topic and for one relevant webpage to encompass irrelevant parts. To predict more precisely, the webpage is divided into smaller units for further analysis. A. Seyfi et al.\cite{24,23} designed an algorithm which segments the webpage in the light of headings into content blocks and calculates the relevancy of each block separately\cite{20}.

To analyze the features of documents, they can be represented by the vector space model (VSM). The relevancy between documents is measured by their similarity and a basic method used to do correlation analysis was introduced in [26].

With the boom of artificial intelligence (AI), some researchers tend to study on learning-based focused crawlers. The classifier is a commonly used tool in learning-based focused crawlers just as what were presented in [27] and [28]. J. A. Torkestani proposed 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\cite{29}. M. Kumar et al.\cite{25} proposed to design a database for information retrieval of Indian origin academicians. H. Lu et al.\cite{30} presented 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 the focused crawler based on links and contents. 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, SiteRank, vision-based page segmentation (VIPS), as well as densitometric segmentation, four algorithms are introduced.

3.1 PageRank

PageRank\cite{3][31] was proposed by S. Brin and L. Page, the founders of Google. The PageRank of a page represents the probability that a random surfer who follows a 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. Sourcepages distribute their PageRank across all of their outlinks. However, PageRank based on hyperlink analysis without any notion of page contents, 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\cite{22}.

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 [33], a combination of PageRank and similarity with the topic was used to guide the crawler. An improved PageRank algorithm called T-PageRank was proposed in [34] and [35], which is based on the “topical random surfer” in contrast to “a random surfer” in PageRank, and for a better performance, it is 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\cite{36} put forward a higher abstraction level of Web graph, that is SiteGraph instead of DocGraph. They distributed the task of computing page ranking to a set of distributed peers each of which crawled and stored a small fraction of web graph. In complete contrast to setting up a centralized storage, indexing, link analysis system used to compute the global PageRank of all documents based on the global Web graph and document link structure, they built a decentralized system whose participating servers computed 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 shows that the computation of the SiteRank of such a Web-scale SiteGraph is fully tractable in a low-end personal computer (PC). Compared with PageRank, the cost is largely reduced while SiteRank keeps good quality of the ranking results. Additionally, it is difficult to spam SiteRank for a PageRank spammer since they have to set up a quantity of spamming Web sites to take advantage of the spamming SiteLinks.

3.3 VIPS

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\cite{20}, was proposed in [38]. 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ound. 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 contents, by means of utilizing visual cues and
document object model (DOM) tree to better partition a page at
the semantic level. The algorithm is based on the fact that
semantically related contents are 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, and colors. The output of the
algorithm is a content structure tree whose nodes depict
semantically coherent content\textsuperscript{[39]}.

In \[38\], it diagramed the working process of VIPS in
details. It first extracts all the suitable blocks from the hyper
text markup language (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 of the URLs ranked in the URLs manager.

### 3.4 Densitometric Segmentation

According to Kohlschütter, for the segmentation problem,
only three types exist: Segmentation as a visual problem,
segmentation as a linguistic problem, and segmentation as a
densitometric problem\textsuperscript{[39]}.

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 contents in small blocks like the header and
footer are usually lacking for analysis.

Kohlschütter’s densitometric approach has a tendency to
work as well as a visual algorithm, while as fast as a
linguistic approach. The basic process is walking through
nodes, and assigning a text density to each node (the density
is defined as the result of dividing the number of tokens by
the 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 that
they managed to get the consuming time down to 15 ms per
page on average. By contrast, visual segmentation takes 10 s to
process a single webpage on average.

### 4. Crawling Strategies

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

#### 4.1 Depth-First Search Strategy

It is an algorithm of traversing graph data structures.
Generally, when the algorithm starts, a node is chosen as a root
node and next nodes are selected in a way that the algorithm
explores as far as possible along the branch. In short, the
algorithm begins by diving down as quickly as possible to the
leaf nodes of the tree. When there is no node for the 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, the seed page 1 is searched, then page
1.1, page 1.1.1, and 1.1.2 get searched in sequence, and finally,
page 1.2, page 1.2.1, and page 1.2.2 are searched in order.

However, this algorithm might end up in an infinite loop\textsuperscript{[41]},
that is 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 level) is
a trial of solving this trap problem, even if it is not exhaustive.
The depth-first alone is not appropriate to focused crawling in
which we hope to cover relevant webpages as many as possible
whereas the depth-first ignores quite a few.

For the experiment purposes, the standard version of
algorithm was modified in order to apply it to the Web search
domain\textsuperscript{[42]}. Instead of selecting the first node, there is selected a
list of seed links, which is the beginning point of the algorithm.
Thereafter, a link is chosen as the 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 a
proper website, it will land to the bucket of found links. In
other case, the 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 a last in first out (LIFO)
queue.

#### 4.2 Breadth-First Search Strategy

The same as the depth-first, breadth-first is an algorithm of
traversing graph data structures as well. The breadth-first
search strategy 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 will go down to the next level
after completely going through one level. For example, in Fig.
1, the breadth-first behaves like the process that in the left tree,
the seed page 1 is searched at the beginning, after which page
1.1 and page 1.2 get searched in sequence and finally, page
1.1.1, page 1.1.2, page 1.2.1, as well as page 1.2.2 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 the breadth-first can be adopted as a focused crawler is based on the theory that webpages at the same level or within a certain level away from the seed URL have a large probability to be topic-relevant [5]. 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 the algorithm starts, there is a chosen starting node, in the case of focus Web crawling, there is a bunch of chosen 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 land in the searching tree on the leaves of the current node (one level below the current node). When the crawler visits all the URLs from the same level, it goes a deeper level and starts the same operation again. So it traverses the tree level by level. The URLs queue behaves in this situation as a first in first out (FIFO) queue [60].

4.3 Best-First Search Strategy

The 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 the candidate URLs and target webpage or candidates’ relatedness to the predefined topic, picking the best one to fetch.

Firstly, a rule about what is “best” should be defined, which usually is a score or rank. In most cases, a classifier algorithm is applied such as the naive bayes, cosine similarity, support vector machine (SVM), and string-matching are used for scoring.

Unlike the blind search algorithms—depth-first and breadth-first, best-first are algorithms that use 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 serve useful information for solving problems. In our case the role of heuristics is to evaluate the website URL, based on the given knowledge before fetching its contents. The name of 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 contents, e.g. evaluating the words in URL and evaluating the anchor text of the link, and more detailed researches in this area have been described in [43] to [49]. 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 the nearest neighbors. The previously stated algorithms the depth-first search and breadth-first search have this restriction.

5. Experiments and Comparisons

5.1 Comparisons 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 a 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 means 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 the geographic names that usually appeared in news relevant with Tibet. The other keywords list was based on ‘发展’, ‘建设’, ‘成果’, ‘成绩’, ‘业绩’, ‘成就’, ‘成果’, ‘发展’, 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 and authority.

In relation to the method of judging whether a webpage was relevant or not, two parts were taken into consideration, the title and the main contents, extracted from ‘title’ and ‘p’ elements of the webpage, respectively. There were two processes doing the estimate. The webpage with at least one geographic keyword in its title and at least three times the keyword in its main contents 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 [89] 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 documents. Afterwards, the processed webpage with the main contents in which the total number of occurrence of any keyword in the other keywords list is greater than 3, was marked as a relevant webpage. We have to admit that 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, the list is taken as the 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 the best-first search, for 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 webpage\cite{51}. 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 which is also short topical descriptions of webpages. The value of the outlink 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 the descending order, and in each crawl the first element was popped out.

Fig. 3 and Fig. 4 illustrate the comparison of the amount of the obtained relevant webpage and frontier (crawler’s request queue) size by three different algorithms. Fig. 3 depicts an amount of relevant webpage versus crawling times. Fig. 4 presents the size of frontier versus crawling times.

![Fig. 3. Amount of relevant webpages changing with crawling times.](image)

![Fig. 4. Frontier size changing with the crawling times.](image)

As it can be seen in Fig. 3, the depth-first search hardly gives good results and provides 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 the breadth-first search, it slowly finds the relevant links. In our expectation, the best-first search performs best, for as the first of the three tested algorithms encounters the high quality link, that is, high ranking in the sorted list.

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

5.2 Comparisons 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 preprocessing webpages in order to improve the performance of the focused crawler.

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. The density-based spatial clustering of applications with noise (DBSCAN), 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 paths in the XPath list (XPATH of each element in the webpage is processed to build this list). The 6 values construct the visual features vector of element, value ‘left’, value ‘left + width’, value ‘top’, value ‘top + height’, value ‘(2×left + width)/2’, and value ‘(2×top + height)/2’. We locate every path of each element’s XPATH in the XPATH list, distributing a value to the corresponding dimension in its n-dimensional vector. And for other dimensions, they are assigned value zero as default. The two vectors are joined as a 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 contents and potential useful links extraction no help, and thus we get rid of these two parts from clusters. Figs.5 and 6 are the segmentation results and final output after disposing the noise, header, and footer, respectively.

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 queuing links includes two parts. One is gotten from their anchor text described at the first experiment, and clusters information is taken into consideration for the other. We assume the cluster whose area is the largest contains elements that form the main contents, which is accord with common sense. When the amount of clusters is larger than 5, we assign a value 3 to
Each value has a weight, for reason that the 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 with the same platform, the same data, and the same hardware environment.

Comparing Fig. 7 with Fig. 3, Fig. 8 with Fig. 4, we can draw such a conclusion that the variation trend keeps unchanged. For a better contrast of two experiments, they are put together as what showed in Fig. 9 and Fig. 10.

What is interesting in Fig. 9 is the phenomenon that the segmentation does not show any improvement on the depth-first and breadth-first, on account of that as blind search, the 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 queuing link is not 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 as noise, header, and footer, which cannot fundamentally change their low harvest rate in focused crawling. As for the best-first, there is a significant improvement. In 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, it is obvious that after segmentation and useless clusters filtering, the frontier size keeps increasing under the line of the first experiment. In the central area of a webpage, the links always link to pertinent webpages while links in the header and footer probably not. Our topic is quite narrow, so the links of the central area tend to link to the same website where the topic is widely talked about and that is why the frontier size grows slowly and a quantity of irrelevant links lead to skyrocketing of the frontier size.

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of relevant page</td>
<td>Extremely small</td>
<td>Quite small</td>
<td>Large</td>
<td>Extremely small</td>
<td>Quite small</td>
<td>Quite large</td>
</tr>
<tr>
<td>Computational costs</td>
<td>Low</td>
<td>Extremely large</td>
<td>Quite small</td>
<td>Low</td>
<td>Quite high</td>
<td>Quite high</td>
</tr>
<tr>
<td>Frontier size</td>
<td>Extremely small</td>
<td>Extremely small</td>
<td>Extremely small</td>
<td>Extremely small</td>
<td>Extremely small</td>
<td>Extremely small</td>
</tr>
</tbody>
</table>

Table 1: Comparison of six crawling results about three indicators

5.3 Points on Research Tendency

Undoubtedly, our ultimate target is to raise the harvest rate and cut the processing time down. Even though the harvest rate is the priority, a higher rate cannot be obtained without the cost of time. Visual segmentation, whether VIPS or clustering, takes too much time to process webpages. 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 the comprehensive features (not just the 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 the harvest rate and computational costs.

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

6. Conclusions

As stated above, this survey gives a quick review of algorithms commonly utilized in two main parts of the focused crawler. The basic crawling strategies alone are not appropriate to the topic-driven crawler or 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 links 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, the higher harvest rate and lower computational costs.

References


the 26th Intl. Conf. on Very Large Data Bases, San Francisco, 2000, pp. 527-534.


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

**Shi-Lei Huang** was born in Chongqing in 1993. He received the B.S. degree from UESTC in 2016. He is currently pursuing the M.S. degree with the School of Computer Science and Technology, Zhejiang University, Hangzhou. His research interests include data analysis and natural language processing.

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

**Huan Zhang** was born in Henan in 1993. He received the B.S. degree from UESTC 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 in 1992. He received the B.S. degree from UESTC 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 in 1993. He received the B.S. degree from UESTC in 2016. He is currently pursuing the M.S. degree with the School of Electronics and Information Engineering, Xi’an Jiaotong University, Xi’an. His research interests include big data and memristor.