International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.8 (2014), pp.251-260
http://dx.doi.org/10.14257/ijmue.2014.9.8.22

The improved Shark Search Approach for Crawling Large-scale Web Data

Youwei Yuan[1*], Dou Chen[1], Yong Li [1], Dongjin Yu[1], Lamei Yan[2] and Zhixiang Zhu[1]

[1]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, Zhejiang, P.R China, 310018*
[2] *School of Media and Design, Hangzhou Dianzi University, Hangzhou, Zhejiang, P.R China, 310018*
**yyw@hdu.edu.cn*

Abstract

Web crawling is an important approach for collecting larger-scale web data on, and keeping up with, the rapidly expanding Internet. This paper puts forward the improved shark search approach for crawling large-scale Web data based on link clustering and the technology of tunnel. In this study we focus on the classification of Web links instead of downloaded web pages to determine relevancy which can avoid local optimum of the traditional shark search algorithm. The experiments show that the improve shark search algorithm can provide the simplest alternative for conquering the issue of instantaneous page which are ranked lowly allied to the given topic at hand.

Keywords: *Link clustering, Tunneling, Shark-Search, Focused Crawler*

1. Introduction

A web crawler (also known as a web spider or web robot) is a program or automated script which browses the World Wide Web in a methodical, automated manner. Web crawlers are mainly used to automatically maintain tasks on a Web site, such as checking links or validating HTML code and create a copy of all the searched web data for later processing by a search engine, which will index the downloaded pages to provide efficient searches. With The World Wide Web (WWW) rapidly developing, the number of Web pages is growing exponentially. According to the statistics [1], up to June 2013, China has over 100 billion Web pages and nearly 2.94 million websites. So it is really difficult for net users to gather the information that they want. Although we have general search engine such as Google, BIDU and so on, it is still inconvenient for us to get the information in specific area. To solve this problem, researchers proposed the vertical search engine [2].

Topical crawling is currently a hot research topic that holds the promise of benefiting from several sophisti-cated data mining techniques [3]. The performance of a focused crawler mainly decided by the richness of links in relevant topic being searched, and focused crawling usually depends on a general web search engine for providing seed points. Focused crawlers can be mainly categorized as: classic focused crawlers、Semantic Crawlers and Learning Crawlers [4] .Classic Focused Crawlers [5] take as input a user query that can describe the related topic and a set of seed page URLs. Text similarity is computed by the Boolean or the Vector Space Model (VSM) [6]. Semantic Crawlers [7] are a variation of classic focused crawlers for harvesting semantic web content which has been implemented in Java using the Jena API. Semantic Crawlers provide a modular framework with huge flexibility in configuring the retrieval and storage of harvested content through the storage of HTTP caching data [8-9].

ISSN: 1975-0080 IJMUE

Learning Crawlers use some training process for assigning visit priorities to web pages which can guide the crawling process. Context Graphs and Hidden Markov Models are considered as the page content and the corresponding classification related the searching topic and the link structure of the Web [10]. Above three types of classification techniques are popular and well established in the areas of text and data mining with readily available implementations and analyze in several programming languages [11].

A vertical search engine is different from a general web search engine which focuses on a specific segment of online content. The vertical content area may be basically depends on the media type and genre of content [12-14]. Compared with general search engine, the vertical search engine has some advantages: (1) it focuses more on the specific field, and has more professional perspective; (2) it can provide more concrete search results; (3) it can provide more accurate results. Owing to the topical crawler, all these specialties of the vertical search engine could be realized. It was the first time that topical crawler which could crawl the specific topic Web pages was suggested in 1999. A well-designed crawling strategy could be key point to topical crawler, if a topical crawler has an efficient crawling strategy [15-16].

A variety kind of topical crawler appeared since the topical crawler was suggested. The Shark-Search [17] algorithm (OSS, Original Shark-Search) is a classical topical searching algorithm. It has a lot of advantages such as: (1) proposed "similarity engine" to evaluate the relevance of documents to a given query; (2) brought in the vector space model (VSM)[18] to calculate similarity score of the theme; (3) proposed to use anchor text of the link in order to decide how to proceed net surfing on the Web. However, the algorithm still has two shortcomings such as the deficiency of anchor text of link and the deficiency of local optimum [19-20].

To solve the above problem, in the study we proposed an improved algorithm (SSCT, Shark-Search based on link Clustering and Tunneling) that was combined with two mechanisms called link clustering and tunneling to overcome two shortages respectively.

The rest of the paper is structured as follows. In section II we present Shark-Search algorithm of related work. In Section III we cover in detail the design of our suggested shark-search algorithm. In Section IV we present the experimental results with discussion. Finally, we conclude the paper with future work in section V.

2. The Shark-Search Algorithm and Related Work

The Shark-Search algorithm a refined version of Fish-Search [21] algorithm is a kind of strategy, which was based on the analysis of text content. For the algorithms based on this strategy, they evaluate a link by computing the similarity among the contents of Web pages, the topics, and the similarity between anchor text and the topics.

Since the Shark-Search algorithm was put forward, many scholars have improved it in delicate ways. In literature [22], due to a lot of noisy links will be contained when using Shark-Search algorithm, it improved the algorithm by proposing page segmentation which can accurately evaluate the relevance from three granularities: page, block and link. In literature [23], it improved the Shark-Search in search width, link similar judgment and crawling link selecting strategies, takes "first search, after the judgment" of the search process. In literature [24], a combination of URL-analysis and host-control strategy is proposed to overcome the viscousness phenomenon of the original Shark-Search algorithm.

The similarity score of Shark-Search algorithm child node is determined by three factors: the anchor text context, anchor text and the similarity inherited from parent node. The score of URL in the URL list is calculated by equation (1).

$$potential\_score(child) = \gamma * inherited(child) + (1 - \gamma) * neighborhood(child).(\gamma < 1) \quad (1)$$

Where *inherited* (child) is used to calculate the inherited score of child node from parent node which is defined as equation (2).

$$inherited(child\_url) = \begin{cases} \alpha * sim(t, father\_url), & sim(t, father\_url) > 0 \\ \alpha * inherited(father\_url), & else \end{cases} \quad (2)$$

In the equation (2), α is a predefined decay factor which is range from 0 to 1 and t is topic string. The *sim(t,parent)* is the similarity between t and parent which is calculated by VSM and defined as equation (3).

$$sim(d_1, d_2) = \frac{\sum_{k=1}^{n} (w_{ik} * w_{jk})}{\sqrt{(\sum_{k=1}^{n} w_{ik}^2)(\sum_{k=1}^{n} w_{jk}^2)}} \quad (3)$$

In which d_1 and d_2 are two different documents. The n means the dimension of the vector. $(w_{i1}, w_{i2}, w_{i3},\ldots,w_{ik})$ is feature vector of the text, $(w_{j1}, w_{j2}, w_{j3},\ldots,w_{jk})$ is feature vector of topical keywords.

The *neighborhood (child)* in equation (1) is used to calculate the score of the neighborhood anchor which is defined as equation (4).

$$neighborhood(url) = \beta * anchor\_score(url) + (1 - \beta) * anchor\_context\_score(url).(\beta < 1) \quad (4)$$

Where β is a predefined constant range from 0 to 1 and *anchor_score (child)* is used to calculate the similarity score of the anchor content which is defined as equation (5). For *anchor_context_score(child)* is used to calculate the similarity score of the anchor context and defined as equation (6).

$$anchor\_score = sim(t, anchor\_text) \quad (5)$$

$$anchor\_context\_score(child) = \begin{cases} 1, & anchor\_score(child) > 0 \\ sim(t, anchor\_text\_context), & else \end{cases} \quad (6)$$

In the equation (5), *anchor_text* is the content of the anchor. In the equation (6), *anchor_text_context* is the textual context of the anchor.

In Shark-Search algorithm, we should predefine four parameters: α, β, γ. For the most web content, its links and text emerge continuously. Because of this particularity, the Shark-Search takes context information of the anchor as the important factor of *potential_score*.

3. The Improvement of the Shark-Search

3.1. Adding Link Clustering to the algorithm

Compared with Fish-Search algorithm, Shark-Search algorithm has many improvements: (1) using similarity engine to evaluate the relevance of documents to a given query instead of the binary (relevant/irrelevant); (2) refining the calculation of the potential score of the children not only by propagating ancestral relevance scores deeper down the hierarchy, but also by making use of the meta information contained in the links to documents; (3) using the close textual context of the link, and combining the information extracted from it with the information extracted from the anchor text. However, after analysis of the Shark-Search algorithm, we can find some shortcomings described as follows.

Due to the deficiency of anchor text context, we added link clustering to the Shark-Search algorithm. We used a kind of clustering algorithm-K-mean algorithm to cluster links from different blocks of the page, then we calculated the similarity between the topic and class after the clustering and replaced anchor text context with class score lastly. We

use K-mean algorithm because it's not only easy to implement, but also efficient and its time complexity is O(KN). Besides, the algorithm is easy to converge and the solution based on the nearest one.

The description of K-mean algorithm [25] as follows:

(1)Choose any K texts as centroid of the initial class;

(2)Repeat the following steps until no change happen:

a) Merged every text to the nearest class on the basis of text mean value;

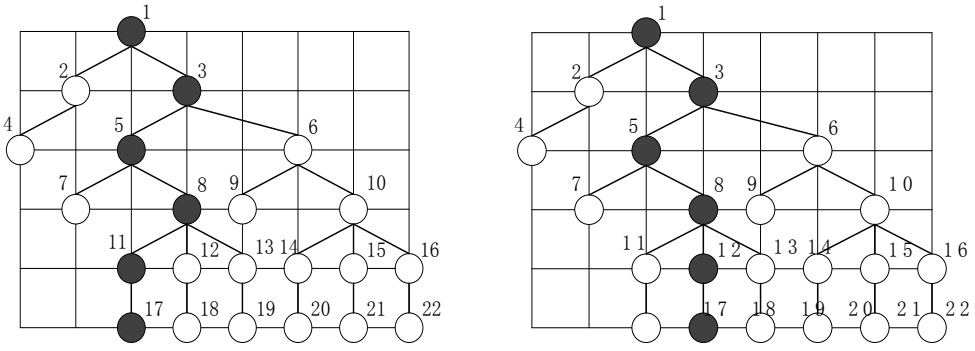
b) Update the mean value of the class.

The pseudocode of K-mean as follows:

```
KMean(X1,X2,…XN,K)
{
    Initialize clustering allocation plan, assume the N vectors as A[1],A[2],…A[N] ;
    repeat
            change=false;
      for(i=1 to N)
      {
          K = arg min_k dist(X_i,C_k)//calculate the minimum distance between X_i and C_k;
          if(A[i] not equals to K)
          {
             A[i]= K;
             change=true;
           }
      }
  until change is false return A[1],A[2],…,A[N];
}
```

The steps of how to obtain class score are as follows:

(1)Transform the information of Web page into the document object model tree; then number the different node of the tree according to the different layer; finally, extract the number path of the links. The method of how to cluster can be described as this: the number path of the link1 is 1-3-5-8-11-17(Figure 1(a)), and the number path of the link2 is 1-3-5-8-12-18(Figure 1(b)), we can cluster them as the same class since they have the same number path 1-3-5-8.



(a) is the path of link1 (b) is the path of link2

Figure 1. The Path of Tow Links

(2) Put the link of the page into queue in accordance with the order of extraction, and find the maximum same path string between two nodes which are not less than 2. Then fetch all the elements of the string and merge them into the class they belong to. Repeat the step (2) until all links cluster into its own class;

(3) We assume L as all link collections to be clustering; Gi as link collections belong to class I; class_num as current class number; flag as a mark. Then do the following steps:

a)Initialization : set L={u1, u2,… un};G1, G2,…Gn=ϕ ; class_num=1; flag=1;

b)When the collection of L is non-null and flag=1, set flag=0;

c)Traverse every ui in the collection of L, if there exist the same path to ui that greater than 1, then put ui into the corresponding Gclass_num, class_num plus 1, set flag=0;

d)Repeat the step b) until flag=0 or L is empty.

(4)From the step (3) we can obtain the number of links $| G_i |$, we assume cluster_url_num as the total number of classes, cluster_url_num=Max(class_num), class score is defined as equation (7).

$$\text{class\_score} = \frac{\sum \text{anchor\_score(url)}}{\text{cluster\_url\_num}} \tag{7}$$

(5)Replace anchor_context_score with class_score we can get a new neighborhood_score as equation (8).

$$neighborhood\_score(url) = \beta * anchor\_score(url) + (1-\beta) * class\_score(url) \tag{8}$$

From the above 5 steps, we can calculate potential score of the improved algorithm.

3.2. Adding Tunneling

As for the topical crawler, for the sake of trying utmost to avoid crawl the page irrelevant to the topic, it should forecast the topic of the page. However, we can't forecast topic absolutely correct. To avoid neglecting the potential topic related pages, we added tunneling [26] to the algorithm. Divide ULR waiting queue to queues: relevant_Queue (topic relevance queue, its topic forecast value is not less than a certain threshold) and irrelevant_Queue (topic irrelevance queue, its topic forecast value is lower than a certain threshold). We assume δ (lower than 1 and defined by users) as the threshold which decide page into relevant_Queue or irrelevant_Queue and μ (higher than 1 and defined by users) as the threshold which decide the depth of the crawling.

From the link clustering, we can obtain the topic forecast value of the page, and on the basis of forecast value, we know which URL waiting queue the URL of the page belong to: relevant_Queue or irrelevant_Queue. For the URL in relevant_Queue, because of topical relevance, crawl the corresponding page directly. As to the URL in irrelevant_Queue, continue crawling its descendant nodes and stop crawling its descendant nodes when the crawl depth is not less than μ .

The flow diagram of the improved Shark-Search algorithm is as following Figure 2.

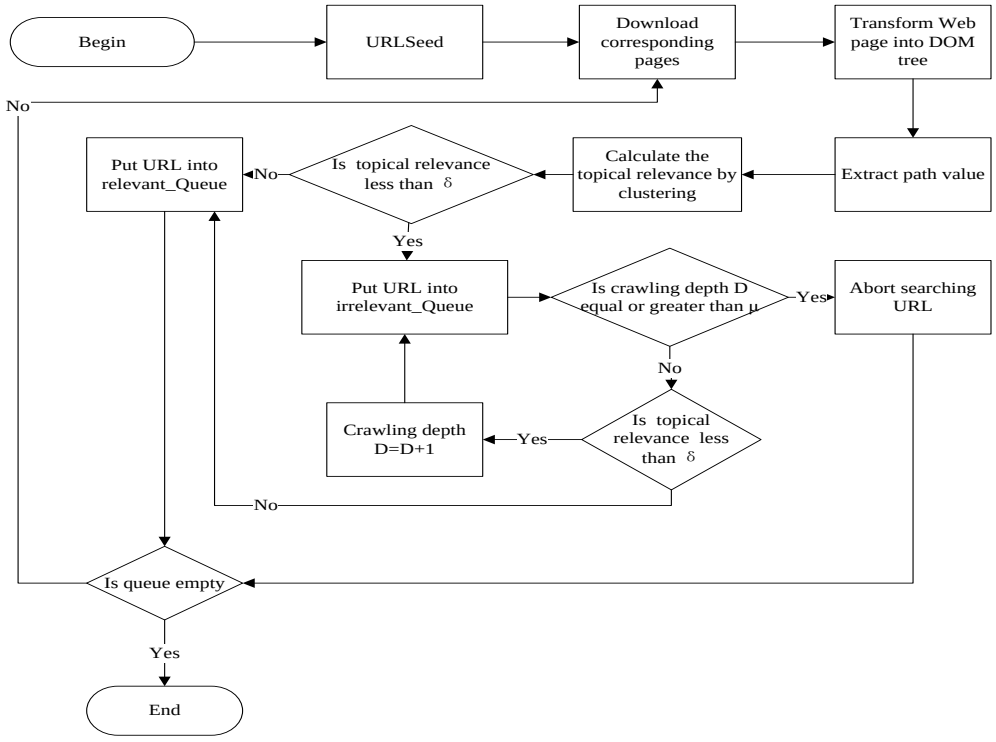


Figure 2. The Flow Diagram of the Improved Shark-Search Algorithm

4. Experimental Results

In this paper, we choose "second-hand house" as the topic, the pages are crawled from six different house information websites: Anjuke, SouFun, 58TongCheng, Ganji, Fangtu and Tou ming shou fang. For the topical crawler, URL seeds and keywords are indispensable, and the detailed information is shown in Table 1.

The parameters are set as follows: α =0.6, β =0.8, γ =0, δ =0.5, μ =3. Crawling the pages by using the information in Table 1, the results of the original algorithm and improved one shown in Table 2. The experiment carries on the statistics after crawling every 600 pages.

From the experimental data in Table 2, we can see that the improved Shark-Search algorithm crawled more topical relevance pages than the original algorithm. Although as the number of total page growth, the rate of growth have downward tendency, overall, the performance of the improved algorithm is better than the original one.

Table 1. The Information of URL Seeds

URL Source	URL Seeds	topic	keywords
Anjuke	http://hangzhou.anjuke.com/sale/	Second-hand house	Real estate, purchase a house, second-hand house, property information
SouFun	http://esf.hz.soufun.com/house/		
58TongCheng	http://hz.58.com/ershoufang/		
Ganji	http://hz.ganji.com/fang5/		
Fangtu	http://hangzhou.fangtoo.com/es/		
Tou ming shou fang	http://www.tmsf.com/esf/		

Table 2. Data Comparison between Two Algorithms

Total pages	Relevance pages		Rate of growth (%)
	OSS	SSCT	
600	102	183	79.41
1200	257	434	68.87
1800	594	1121	88.72
2400	853	1498	75.61
3000	1991	2424	21.75
3600	2219	2543	14.60

Figure 3 shows the experimental comparison of the original Shark-Search algorithm and the improved one. From Figure 3, it is easy for us to find that the improved algorithm performed obviously superiority than the original one.

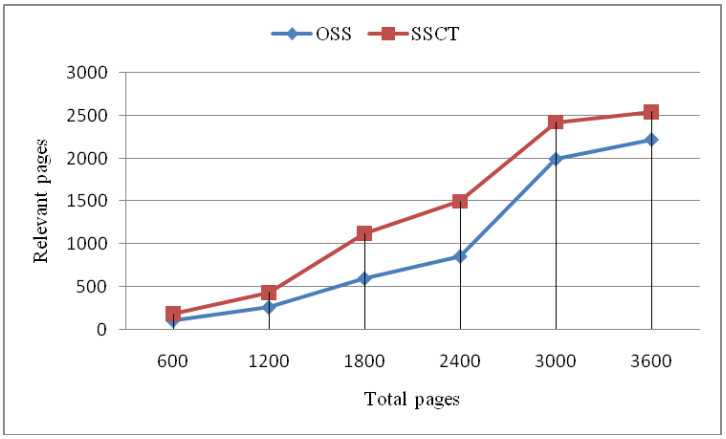


Figure 3. Experimental Comparison of Two Algorithms

Figure 4 is the number of relevant pages comparison between two algorithms every 300 seconds. As seen in Figure 4, the OSS performs not so good, the number of relevant page it crawls grow slowly as time elapse. For the SSCT, it performs not so good before 900 seconds, but after 900 seconds, it performs excellently.

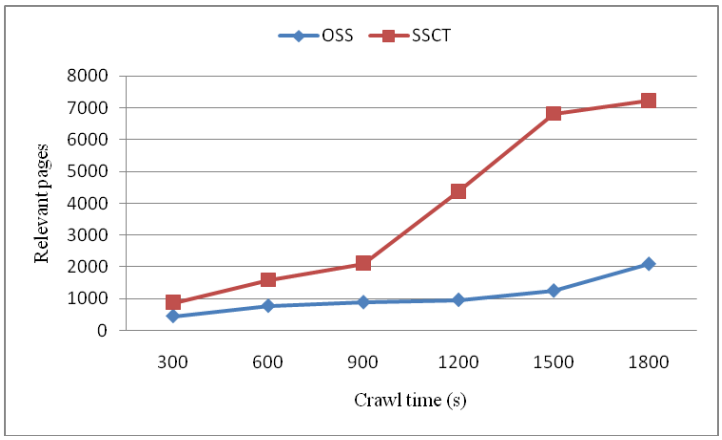


Figure 4. Comparison of Two Algorithms Every 300 Seconds

From the Figure 3 and Figure 4, we can see the impoved Shark Search algorithm performs better than the original algorithm after adding link clustering and tunneling, so we can conclude that the improved Shark-Search algorithm adding link clustering and tunneling is feasible.

5. Conclusions and Future Work

Different from any other current Web crawlers, in this study we proposed an improved algorithm by adding link clustering and tunneling to overcome the two shortcomings of shark-search algorithm. The former one is to solve the shortage of anchor text context and the latter one is to solve the shortage of local optimum. We can easily extend our experiments with other Linear Regression, Bayes Network classification algorithms. Another simple extension can be achieved by combining the scores for a given URL if the URL appears on several downloaded pages. The experimental shows that the improved algorithm is more feasible and effective compared with the original one. In this paper, it is not efficient for us to crawl Web pages in single computer if the data is extremely massive.

Our optimized algorithm provides future researcher with a feasible and practical algorithm. However, accurate search result also depends mainly on choosing a good Similarity and Confidence value after keyword training related to research topic.

In the future, in order to cater the requirement of crawling massive data, we will take distributed crawling into consideration as well as different heuristics for combining the scores and compare the affects of such heuristics on crawling performance.

Acknowledgment

The work was supported by NSFC (Grant No. 61272032) and also supported by the nature science foundation of Zhejiang province (No. Y6090312).

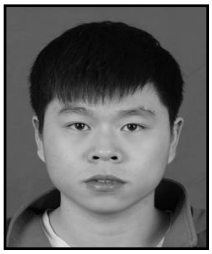
References

[1] CNNIC, The 32th Internet Development Statistic Report of China [EB/OL] [2013-07-17], http://www.cnnic.net/hlwfzyj/hlwxzbg/hlwtjbg/201307/t20130717_40664.htm, **(2013)**.
[2] L. Jing, Q. Yihui and J. Xu, "The Design and Implementation of B2C Web site-Online Mobile Phone Shop", IJUNESST, vol. 6, no. 2, **(2013)**, pp. 107-114.
[3] A. Badia, T. Muezzinoglu and O. Nasraoui, "Focused crawling: experiences in a real world project", Proceedings of the 15th International Conference on World Wide Web, Edinburgh, no. 5, **(2006)**, pp. 1043-1044.
[4] Q. Xu and W. Zuo, "First-order Focused Crawling", Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada, no. 5, **(2007)**, pp. 1159-1160.
[5] G. Pant and P. Srinivasan, "Link contexts in classifier-guided topical crawlers", IEEE Transactions on Knowledge and Data Engineering. vol. 18, no. 1, **(2006)**, pp. 107-122.
[6] J. Eggermont, J. N. Kok, and W. A. Kosters, "Genetic Programming for Data Classification: Refining the Search Space", in Proceedings of the 15th Belgium/Netherlands Conference on Artificial Intelligence, no. 3, **(2003)**, pp. 123-130.
[7] W. Fan, E. A. Fox, P. Pathak and H. Wu, "The effects of fitness functions on genetic programming-based ranking discovery for web search", Journal of the American Society for information Science and Technology, vol. 55, no. 7, **(2004)**, pp. 628-636.
[8] M. Day, "Collecting and preserving the World Wide Web", Tech. rep., UKOLN, University of Bath. February. http://library.wellcome.ac.uk/assets/WTL039229.pdf, **(2003)**.
[9] G. Pant and P. Srinivasan, "Learning to crawl: Comparing classification schemes", ACM Transaction on Information Systems, vol. 23, no. 4, **(2005)**, pp. 430-462.
[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval", Information Processing and Management, vol. 24, no. 5, **(1988)**, pp. 513-523.
[11] M. Chau and H. Chen, "Comparison of Three Vertical Search Spiders", Journal of Computer , vol. 36, no. 5, **(2003)**, pp. 56-62.
[12] R. Shettar and R. Bhuptani, "A Vertical Search Engine – Based on Domain Classifier", International Journal of Computer Science and Security, vol. 2, no. 4, **(2009)**, pp. 18-27.
[13] Y.-t Zhang, L. Gong, Y.-c. Wang, "An improved TF-IDF approach for text classification", Zhejiang university China, vol. 6, no. 8, **(2005)**, pp. 49-55.
[14] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles and M. Gori, "Focused crawling using context graphs", In Proceedings of the 26th International Conference on Very Large Data Bases, no. 9, **(2000)**, pp. 527-534.
[15] K. Bao and Z. Sun, "The Design of Network Support System for Community Management", IJUNESST, vol. 2, no. 6, **(2013)**, pp. 53-62.

[16] C.-Y. Tseng and K.-Y. Liu, "A Modified Priority Based CPU Scheduling Scheme for Virtualized Environment", IJHIT, vol. 6, no. 3, **(2013)**, pp. 39-50.
[17] C.-y. Wang and Y.-f. Li, "Study of Information Filtering Technology in a Vertical Search Engine", Information Science, vol. 32, no. 3, **(2014)**, pp. 93-97.
[18] H.-y. Song, X.-r. Liu and H.-j. Qian, "A Novel Crawling Strategy of Focused Web Crawler", Computer Applications and Software, vol. 28, no. 11, **(2011)**, pp. 264-267.
[19] M. Herseovici, M. Jacov and Y. S. Maarek, "The Shark-Search algorithm an application: Tailored Web Site Mapping", Computer Networks and ISDN Systems, vol. 30, no. 1, **(1998)**, pp. 317-326.
[20] G. Salton and M. J. McGill, "Introduction to Modern Information Retrieval", Computer Series, New York, NY, **(1983)**.
[21] P. De Bra, G.-J. Houben, Y. Kornatzky and R. Post, "Information retrieval in distributed hypertexts", Proceedings of RIAO'94. Intelligent Multimedia. Information Retrieval Systems and Management, New York. NY, no. 10, (**1994)**, pp. 41-43.
[22] J. Chen and Z.-m. Chen, "Improved Shark-Search algorithm based on page segmentation", Journal of Shandong University (Natural Science), vol. 42, no. 9, **(2007)**, pp. 62-65.
[23] R.-g. Yang, Y. Song, X.-z. Meng, "Multimedia topic search algorithm based on improved Shark-Search", Computer Engineering and Applications, vol. 14, no. 4, **(2010)**, pp. 152-154.
[24] L. Luo, R. Wang and X. Huang, *et al.* "A novel shark-search algorithm for theme crawler", Web Information Systems and Mining, Springer Berlin Heidelberg, vol. 7529, no. 10, **(2012)**, pp. 603-609.
[25] Z. Ju, "Image Edge Detection Based on the Improved K-means Clustering Algorithm", Bulletin of Science and Technology, vol. 28, no. 6, **(2012)**, pp. 47-48.
[26] Y.-z. Bai and J.-z. Liang, "Research and implementation for focused crawler based on probabilistic model", Computer Engineering and Science, vol. 35, no. 1, **(2013)**, pp. 160-165.

Authors

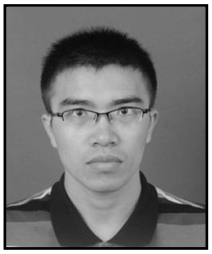
Youwei Yuan was born in 1966, received his doctor degree in computer science from Wuhan University of Technology, China, in 2007. He is currently a professor of computer science, Hangzhou Dianzi University (Hangzhou, China). His research interests include artificial intelligence, data mining and distributed parallel processing, He has published over 50 technical papers in prestigious journals and conferences, 20 papers been indexed by SCI and EI.



Dou Chen was born in 1990, is a postgraduate student of computer science, Hangzhou Dianzi University (Hangzhou, China).His research interests include data mining and search engine.

Yong Li was born in 1987, is a postgraduate student of computer science, Hangzhou Dianzi University (Hangzhou, China).His research interests include data mining and search engine.

Dongjin Yu received his BS and MS in Computer Applications from Zhejiang University in China, and PhD in Management from Zhejiang Gongshang University in China. He is currently a professor at Hangzhou Dianzi University, China. His research efforts include service computing, program comprehension and cloud computing.



Lamei Yan is currently a professor of school of digital media and design. She is a visiting professor of RMIT University, Australia from September 2012 to April 2013. Her research interests include image processing, computer aided design and finite element analysis.

Zhixiang Zhu, She received his master's degree in Software Engineering from Hangzhou Dianzi University in 2011. His primary research area focuses on knowledge discovery and data engineering, middleware technology and software system architecture.