DESIGN AND IMPLEMENTATION OF USER-CENTERED ADAPTIVE SEARCH ENGINE

Shailja Dalmia, Ashwin T S and Ram Mohana Reddy Guddeti

National Institute of Technology Karnataka, Surathkal, Mangalore, India

ABSTRACT

With the ever-growing variety of information, the retrieval demands of different users are so multifarious that the traditional search engine cannot afford such heterogeneous retrieval results of huge magnitudes. Harnessing the advancements in a user-centered adaptive search engine will aid in groundbreaking retrieval results achieved efficiently for high-quality content. Previous work in this field have made using the excessive server load to achieve good retrieval results but with the limited extended ability and ignoring on demand generated content. To address this gap, we propose a novel model of adaptive search engine and describe how this model is realized in a distributed cluster environment. Using an improved current algorithm of topic-oriented web crawler with User Interface based Information Extraction Technique was able to produce a renewed set of user-centered retrieval results with higher efficiency than all existing methods. The proposed method was found to exceed by 1.5 times and two times for crawler and indexer, respectively than all prevailing methods with improved and highly precise results in extracting semantic information from Deep web.

KEYWORDS

Search Engine, WWW, Web Content Mining, Inverted Indexing, Hidden Crawler, Distributed Web Crawler, Precision, Deep Web

1. INTRODUCTION

Over the past few decades, with rapid development in the internet and network technology, we bore witness to an unprecedented growth of web content from which high-quality information needs to be extracted efficiently [9]. A lot of work has been done to draw out information from the invisible or deep web due to its target size [12] which is close to 500 times the size of the existing web structure that can be easily searched and extracted. Also, the deep web attracts a lot of research due to its highly dynamic nature. As a result, the field of traditional search engines could not pace up with increasing user demands of retrieval results with problems like centralized nature [4], limited coverage range, abundance of results to filter from and query ambiguity problems. Of late, there has been growing traction, and a focus shift towards user needs as researchers opine the importance of a user-centric mindset in extracting highly relevant and efficient search results [10].

The centralized architecture of the commercial search engines on a single server is prone to failures and other limitations like expansibility. With all this, it can be said that the centralized search engine has run out of steam and fails to cater to the needs of users in this era of BigData. Moreover, the centralized search engine is mainly keyword based [2], so the results might not be relevant to the user requirements. In addition to this, some content of the web called the hidden web is always hidden from the crawler, which needs to be explored.

Natarajan Meghanathan et al. (Eds) : ACITY, AIAA, DPPR, CNDC, WIMNET, WEST, ICSS - 2019
pp. 21-30, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91702

In order to solve these shortcomings of centralized search engine and general search engine, a user-centered adaptive search engine integrated with semantics, based on Hadoop has been proposed. Through this work, we look to tap into this vast potential of semantic integration and distributed computing and make use of this improved model to achieve relevant results with greater efficiency and paving the way for dynamic content extraction model.

The key contribution here is the ability to adapt the search results with respect to users effectively and efficiently. Users need to get relevant results from the web with high precision incorporating dynamism.

The rest of this paper is ordered in following manner. Section 2 outlines previously related work and compare against the proposed method. Section 3 details the methodology and computation mechanism in a distributed environment. Section 4 presents the results of the implementation. Finally, Section 5 provides conclusion and foresight into the work to eliminate a few downsides of existing methods.

2. RELATED WORK

Even though information plays a pivotal role in the modern world, very few search engines methodologies are in place which is efficient and retrieve relevant results. Since most of the traditional search engines suffer from being less efficient and more time consuming, it is imperative to build stronger and more robust models. To this end, Wei et al. [2] used a content-based crawling technique to search and extract the web content related to the query topic. Lin et al. [4] in their work were able to evolve a distributed search engine using the MapReduce paradigm for crawling, indexing, and searching. However, it succumbed to the problem of theme drift by using link crawling technique. While D.Minnie et al. [5] cover various techniques which are used for indexing and searching modules, it doesn't offer a combined approach of indexing and searching algorithms. Till date, the search engines developed to make use of either content based or link based algorithms to perform the crawling operation. Each of these algorithms has its shortcomings. The link based algorithms lead to the theme drift problem, as stated earlier, due to which irrelevant results would be fetched and displayed to the user. The content-based algorithms restrict the crawling process by limiting the number of links that can be crawled.

The crawlers do not extract information from web pages that require user input or prior registration. The searching modules usually interpret only the keywords given by the user and do not consider the semantics of the submitted query.

Although it manages to improve the precision in one or the other way by using different mechanisms of crawling, we aim to conceive an improved adaptive model of the search engine to achieve precise retrieval results relevant to users in a distributed environment to overcome above mentioned existing problems with traditional search engines.

3. METHODOLOGY

The proposed model is a user-centered adaptive search engine built in a distributed environment to retrieve high-quality content relevant [15] to users using four different modules. The work by Lin et al. [4] was incorporated along with an evolved topic-oriented and hidden web crawler to retrieve relevant results efficiently with higher precision.

Figure 1 represents the flow diagram of the proposed model. The idea is to get the webpage which is relevant to a user, related to the topic and has high-quality content and can thus adapt to the needs of the users.

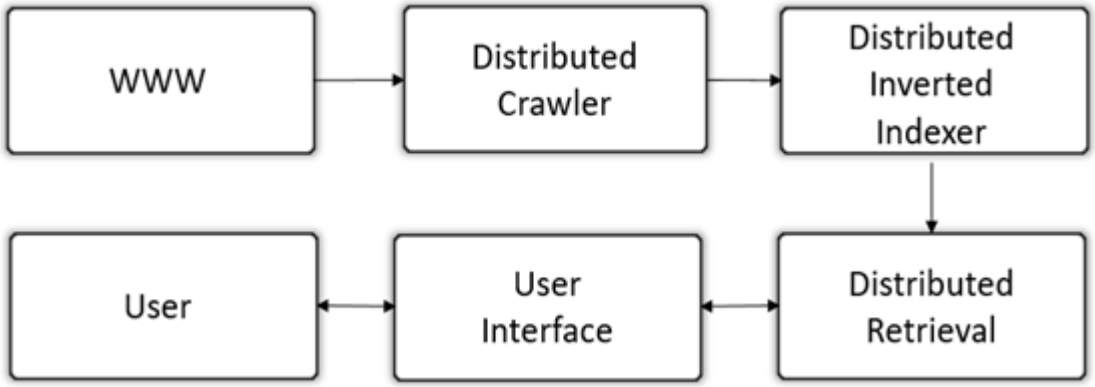

Figure. 1. Work flow of the proposed user-centered adaptive search engine

3.1. Distributed Crawler

3.1.1. Adaptive Crawler

Content-Based Crawling [2] is a simple algorithm which considers web-content relevance but might not be sufficient to fetch results relevant to the user's topic.

Through experimentation, it was found that combining content-based and collaborative-based crawling can get the desired results and adapt to the needs of the user, thus eliminating the theme drift problem and the concept of crawling limited URLs. The computation of the improved algorithm involves the following techniques:

1. **Vector Space Model:** It is an algorithm derived from the relevance of web content. The representation of each document is in the form of a feature vector v= (t_1, t_2, t_3….t_n), where ti is the keyword. Then the similarity between the two documents is calculated using Cosine Similarity measure. The correlation value between two document vectors is calculated using the following formula: If u and v are the two document vectors then,

$$VSM\ (u, v) = (u.\ v)/\ |u|.|v| \qquad (1)$$

2. **Page Rank Calculation:** The Page Rank algorithm calculates the page rank value (PR value) of each page and states that greater the PR value, better the quality of the content in the page. Let PR(X) be the page rank of page X, PR(Lj) be the page rank of page Lj which links to page X, O(Lj) be the number of outgoing links on page j. The variable q is a damping factor which lies between 0 and 1. The formula to calculate the PR value of a web page is as follows:

$$PR(X) = (1\text{-}q) + q * \sum_{j=1}^{N} PR(L_j)/O(L_j) \qquad (2)$$

3. **Improved Algorithm:** We look to retrieve pages that are rich in content quality as well as relevant to the user's topic. Computing hybrid algorithm by combing VSM and PageRank, we were able to achieve the above results. The association between page S and topic page T can be expressed by formula (3) by combining formula (1) and formula (2) ,

$$SIM\ (S,T)=p*\ VSM(S,T) + (1\text{-}p) * PR(S) \qquad (3)$$

where p is a damping factor which lies between 0 and 1, to normalize the percentage of the affect of the web content and quality.

3.1.2. Hidden Web Crawler

As depicted in Figure. 2, Hidden web crawler involves dynamic content extraction [9], which is hidden behind the search forms or page which requires prior registration. Initially, web forms with < form > tag and an input field are detected [14], and user-interface based keyword is selected to submit in the input field to retrieve web page behind the search forms. Two different keywords might fetch the same result, which could decrease the efficiency of crawler. Hence, detection of such duplicate URLs is performed [13], along with automatic processing of the crawler without any manual intervention such that the user-interface based information retrieval is performed to fetch high-quality dynamic content, unlike traditional search engines.

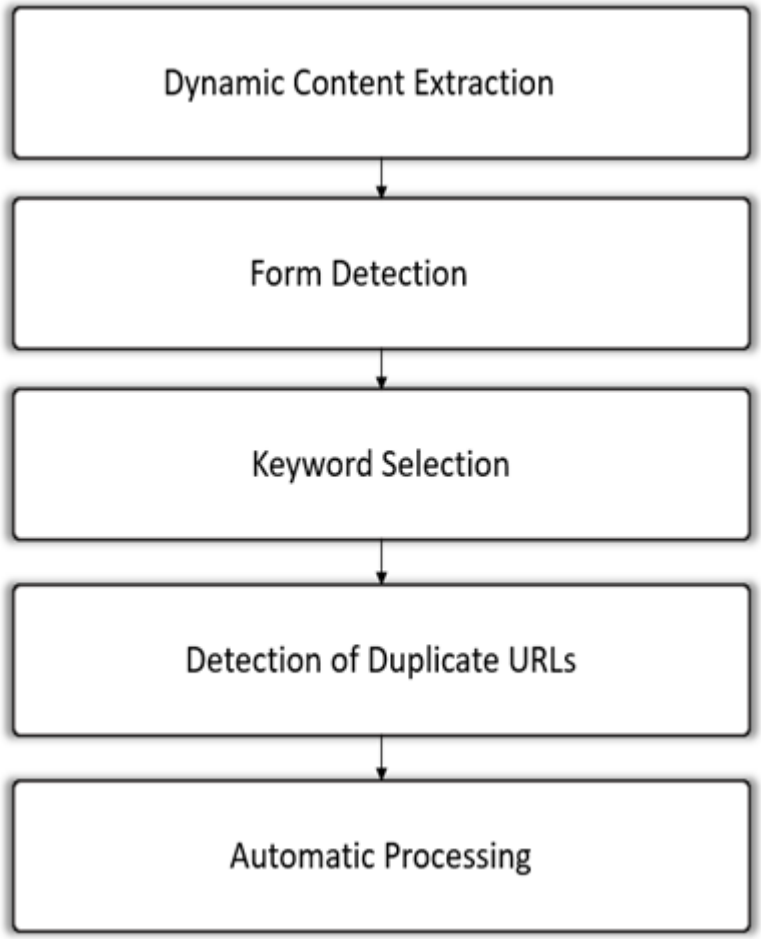

Figure. 2. Work flow of the proposed hidden web crawler

3.1.3. Distributed Module for Crawling

The mapper [4] phase of the crawler starts with tokenizing the file containing the initial URLs. Each mapper is allocated with a URL. Each mapper retrieves the outgoing hyperlinks in the allocated URL and computes the similarity index for each hyperlink using the combined formula (3). If the similarity index > threshold, the url is added to the list of URLs else the URL is discarded. The mapper then extracts the keyword from each of the selected URLs and writes the output to the HDFS, as shown in Figure 3. Each reducer input has a distinct url and aggregates the < url, list(keyword) > into < url, keywords >. The MapReduce paradigm inherently fetches all the < url, keywords > combinations and stores them in HDFS.

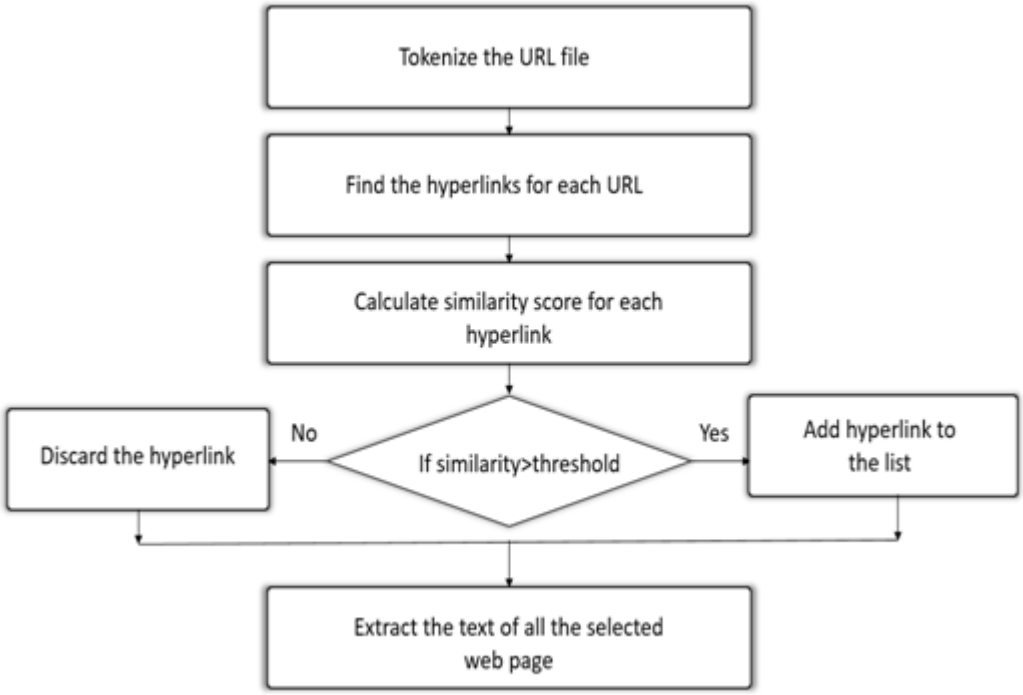

Figure. 3. Distributed Crawler

3.2. Indexing Module

Once the crawler has selected the webpage relevant to user with high-quality dynamism content, it must be stored in an efficient way such that retrieval of results is fast and efficient rather than scanning the entire database of crawled webpages. One such way of storing such webpages is indices [6] which is like a key-value pair.

Depending upon the storage mechanism, indices could be categorized into two categories:

1. **Inverted Index:** We have made use of an inverted index for dynamic web indexing while analyzing a search query to rapidly find web documents that possess the keywords in a query following which we score these web documents by relevance. Table 1. is a simplified representation of an inverted index.

Table 1. Inverted Indexing

The	URL 1, URLS 6, URL 7, URL 2
Goat	URL 3, URL 6, URL 7
Eats	URL 5
Grass	URL 7

2. **Forward Index:** It is typically used to identify recurring URLs that are duplicates because it indexes an array of words for each web document. As proposed in our approach, we look to index a set of web documents with the keywords which are used. Gathering “entire” web documents are performed by scanning the web methodically and meticulously and persisting all visited pages.

This mechanism is shown in Table 2. The above indexing mechanism is used and combined for distributed indexing module which performs efficiently and effectively.

Table 2. Forward Indexing

URL 1	disk, computer, sound
URL 2	fish, ran, hat, fork
URL 3	goat, boat
URL 4	goat, eats, grass

3.3. Distributed Retrieval

The query submitted by the user is tokenized, and the stop words are eliminated, and the keywords are extracted. These keywords will form the basis for searching in the inverted index table built by the indexing module. In order to provide better semantics to the search result, the synonyms of the keywords will also be determined and used to fetch the results. Pages having higher relevance in terms of containing both the words and their synonyms are fetched as a result. The flowchart for implementing the Searching module is, as shown in Figure 4.

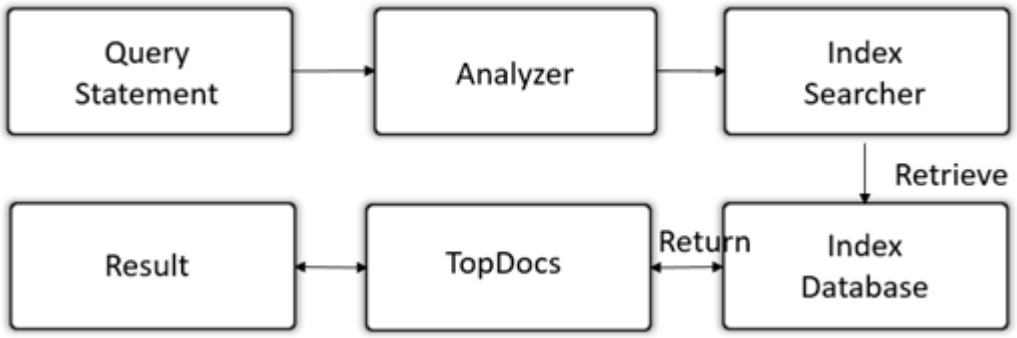

Figure. 4. Framework for Searching

4. RESULTS AND ANALYSIS

The configuration of the system used with one master and two slave node is Operating system as Ubuntu 14.04, Processor as Intel Core i3, Number of Cores as 4, Threads per Core 8, Hadoop Version as Hadoop 2.7.0. The proposed methodology was applied, and precision combined with the working efficiency was used to assess the performance of search-engine. Working efficiency is determined by the pace at which the results are retrieved while precision evaluates the capability of the system in retrieving high-quality, relevant results.

1. **Efficiency Comparison Between Distributed and Centralized Search Engine:** Time is taken to perform crawling, indexing, and searching phase is compared between User-Centered Adaptive Search Engine and Traditional Search Engine. We look to benchmark the performance numbers of our model on the following number of websites to be crawled: 50, 100, 150, 200, 250. Table 3 depicts the total time cost comparison between centralized and distributed crawler.

Table 3. Execution Time of Centralized and Distributed Crawler

Number of URLs	50	100	150	200	250
Centralized Execution Time (sec)	81.34	147.79	229.72	286.42	409.56
Distributed Execution Time (sec)	53.99	105.23	145.42	187.51	252.74

As shown in Figure 5, when the count of URLs crawled are 50, the existing centralized approach takes 81.34(s) while our Distributed Crawler takes 53.99(s). It can be noticed that the efficiency of the Distributed is better than Centralized when the count of URLs is increased, thus substantiating the use of the Map-Reduce Model for large data-sets.

The Distributed Crawler performed much better compared to the Centralized Crawler. The former was found to be 1.5 times faster than the latter. Hence it can be concluded that the time taken to crawl the URLs from the vast Internet required for a search engine reduces to a considerable extent when harnessing the parallel processing capabilities of the Hadoop MapReduce framework.

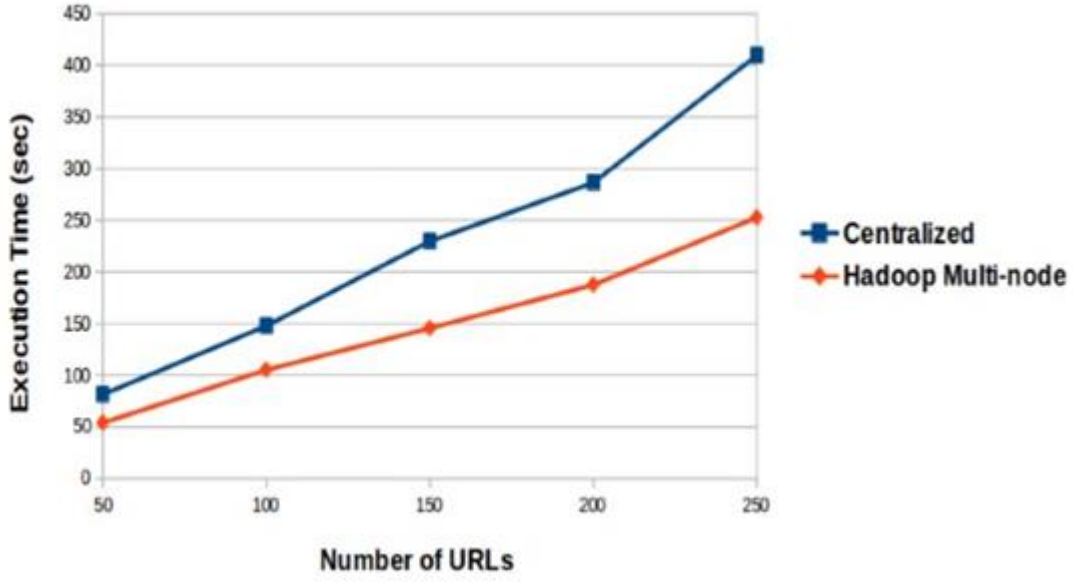

Figure. 5. Centralized Crawler versus Distributed Crawler

The inverted indexing module has been implemented on both Centralized and Distributed Hadoop platform. The tabulations are as shown in Table 4.

Table 4. Execution Time of Centralized and Distributed Indexer

Number of URLs	50	100	150	200	250
Centralized Execution Time (sec)	23.08	116.58	187.34	338.9	600.81
Distributed Execution Time (sec)	24.9	64.96	106.56	160.88	253.18

The Distributed Indexer performed much better compared to the Centralized Indexer. The former was found to be approximately two times faster than the latter. Hence it can be concluded that the time is taken to index the contents of the URLs and the URLs reduces to a considerable extent when executed on Hadoop using the map-reduce framework. A plot of the number of URLs to be indexed versus the time taken is as shown in Figure 6.

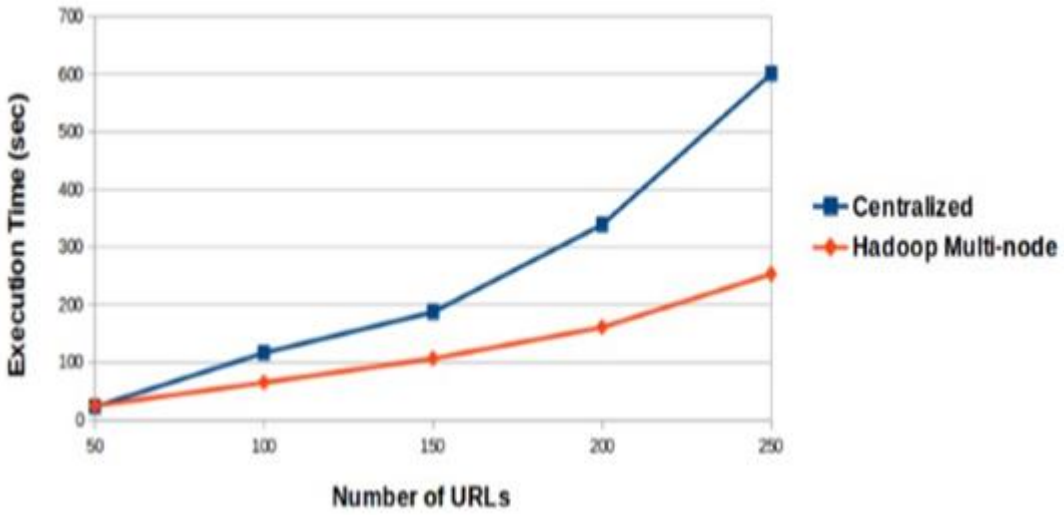

Figure. 6. Centralized Crawler versus Distributed Indexer

2. P**recision Comparison Between Distributed and Centralized Search Engine:** Adaptive Crawler was successful in retrieving the relevant web pages with high-quality content. Here "www.naukri.com" is considered as starting URL, number of URLs set to 500 and "data-scientist" specified as a query. The precision is measured for the retrieval results between Traditional and Proposed Search Engine, which is tabulated in Table 5.

Search Engine	Precision
Proposed	94%
General	53%

It shows that 94% of the search results of the proposed method are highly relevant to input query while only 53% is relevant in case of the traditional approach and the rest are unassociated to the query. Hence, it is the result of the improved adaptive algorithm which factors in both visible and hidden web as well as makes use of the combined approach for web content relevance and link authority.

5. CONCLUSIONS AND FUTURE WORK

The proposed model was found to outperform all preceding works in this area of search engine models. It was a tedious and nerve-wracking challenge to work with huge magnitudes of data factoring in both efficiency and precision as key performance indicators. Observing the above results, the proposed methodology explored in this work is touted to be a very important and efficient step towards search-engine models encompassing both visible and hidden web. This method also ascertains the use of distributed computing to retrieve results effectively. Semantic query eliminated the drawback of keyword-based search engine.

In future, we would like to extend this work in the following directions. First, we would like to compare the performance of the proposed algorithm with the most widely used search engines like Google and DuckDuckGo. Second, we would like to investigate ways to extend the capabilities of existing search engines with above proposed algorithm to achieve higher efficiency and better precision results.

REFERENCES

[1] Fu Yuan, Li Ling, Zhang Hairong, MA Xiaozhen, Zhang Yi ”The Realization of the Distributed Search Engine on Hybrid Cloud Platform”, Second International Conference on Measurement, Information and Control, 2013.

[2] Wang-wei, ”Comparative analysis of vertical search engine”, Seventh International Conference on Measuring Technology and Mechatronics Automation, 2015.

[3] K. Duraiswamy, S. Prabha, J. Indhumathi, ”Comparative Analysis of Various PageRanking Algorithms”, World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Control and Information Engineering Vol No:8, 2014,65-69.

[4] Ma Yajie, Cheng Lin, ”Design and implementation of a vertical search engine using MapReduce”, Eighth International Conference on Measuring Technology and Mechatronics Automation, 2016.

[5] S.Srinivasan, D. Minnie, “Intelligent Search Engine Algorithms on Indexing and Retrieval of Text using Text Representation” International Conference on Recent Trends in Information Systems, 2011.

[6] Yi ping, Wu wenzhong, ”Applications of Distributed Search Engine using Hadoop”, Computer System and Applications, 2012.

[7] Tom White, ”Hadoop The Definitive Guide”, International Journal of Computer Applications Volume 55– No.1, October 2012, 975-8887.

[8] Hadoop Set-Up: Configuration of Hadoop Multi-Node Cluster Environment, https://www.tutorialspoint.com/hadoop/hadoop-multi-node-cluster.html, visited on December 2018.

[9] H. Garcia-Molina, S. Raghavan. Crawling and Indexing the Hidden Web, in: Proc. of the 27th International Conference on Very Large Databases, 2001.

[10] Hui Chen, King-Ip Lin, ”Automatic Information Detection from the Hidden Web”, Information Technology: Coding and Computing, IEEE, 2002.

[11] Hao shukui, ”Architecture Analysis of Hadoop, HDFS and MapReduce Paradigm”. Designing Techniques of Posts and Telecommunications, 2012.

[12] The Deep Web: Measuring Hidden Value, http://www.completeplanet.com/Tutorials/DeepWeb/, visited on January 2018.

[13] Wang Hui-chang, Ruan, Shu-hua, Tang,Qi-jie. "The Implementation of a Web Crawler URL Filter Algorithm Based on Caching”. Second International Workshop on Computer Science and Engineering, IEEE, 2009.

[14] Jayant Madhavan, David Ko, Luc jaKot, Vignesh Ganapathy, Alex Rasmussen, Alon Halevy. “Google's Deep-Web Crawl”, Proceedings of the International Conference on Very Large Databases (VLDB), 2008.

[15] Li guangli, Liu juefu. Research and Implement of Vertical Search Engine[J]. Journal of Intelligence,2009,28(10):144-147.

AUTHORS

Shailja Dalmia works as Data Scientist in Merchant and Acquirer Processing Analytics Team in Visa. Her current research interests include machine learning, Big data analytics and Real time streaming analytics varying from Payment Analytics to Early Warning Detection system for merchant attrition rate.

Ashwin T S, currently, he is pursuing his full-time Ph.D. at National Institute of Technology Karnataka, Surathkal, Mangalore, India. His research interest includes Affective Computing, User-Centered Design and Adaptive Systems, Human Computer Interaction, and IOT. He is the student member of IEEE and has more than 25 research publications in reputed and peer-reviewed International Conference and Journal publications.

Ram Mohana Reddy Guddeti received his B.Tech from S.V. University, Tirupati, Andhra Pradesh, India in 1987; M.Tech from Indian Institute of Technology, Khargpur, India in 1993 and Ph.D. from The University of Edinburgh, U.K in 2005. Currently, he is the Professor and Head, Department of Information Technology, National Institute of Technology Karnataka Surathkal, Mangalore, India. His research interests include Affective Computing, Big Data and Cognitive Analytics, Bio-Inspired Cloud and Green Computing, Internet of Things and Smart Sensor Networks, Social Multimedia and Social Network Analysis. He is a Senior Member of both IEEE and ACM; Life Fellow of IETE (India); Life Member of ISTE (India) and Life Member of Computer Society of India. He has more than 200 research publications in reputed and peer reviewed International Journals, Conference Proceedings and Book Chapters.