A Semi-Supervised Learning Methodology in Developing a Search Engine

Mohd Noah A. Rahman, Afzaal H. Seyal, Ibrahim Edris, and Siti Aminah Maidin

***Abstract*—**This article reports a generic search engine development using a semi-supervised learning (SSL) methodology which was advocated as a mean to understand better the flow of algorithm in a semi-controlled environment setting. It uses the Breadth-First–Search (BFS) algorithm set of data structures to retrieve web pages and the subsequent indexed links. This technique requires proper approach to avoid flooding of irrelevant links which can constitute a poor design and constructed search engine to crash. The main idea of this report finding is to learn and comprehend the basic concept of web crawling, indexing, searching and ranking the output accordingly in a semi-controlled environment. This is a contrast as compared to an unsupervised and supervised learning conditions which could lead to excessive information.

***Keywords*—** crawler, search engine, indexing, semi-supervised learning.

I. Introduction

THE introduction of search engines such as Bing [1], Google [2] and Yahoo! [3] which have proliferated tremendously over the years, have revolutionized the way people access to information. There are approximately 300 million Google users and 2 billion log-in searches per day [4] and few are valuable [5]. These figures are so stupendous which could easily lead to information overloading [6] and vocabulary differences [7].

The core of the searching capabilities is the availability of search engines which represent a user interface needed to permit users to query data for information [8]. The classification of data into various categories lead to the formulation of various learning techniques (supervised, unsupervised and semi-supervised). There has been tremendous interest between supervised and unsupervised learning [9], [10]. Basically, learning has been studied either in the unsupervised paradigm with clustering and detection of outliers where all the data are unlabeled, or in the supervised paradigm with classification and regression where all the data are labeled.

Mohd Noah A. Rahman is with the Institut Teknologi Brunei, Tungku Link, Gadong, BE 1410 Brunei Darussalam (phone: 6732461020; fax: 6732461036; e-mail: noah.rahman@itb.edu.bn).

Afzaal H. Seyal is with the Institut Teknologi Brunei, Tungku Link, Gadong, BE 1410 Brunei Darussalam (phone: 6732461020; fax: 6732461036; e-mail: afzaal.seyal@itb.edu.bn).

Ibrahim Edris is with the Institut Teknologi Brunei, Tungku Link, Gadong, BE 1410 Brunei Darussalam (phone: 6732461020; fax: 6732461036; e-mail: Ibrahim.edris@itb.edu.bn).

Siti Aminah Maidin was with the Institut Teknologi Brunei, Tungku Link, Gadong, BE 1410 Brunei Darussalam (e-mail: ameena.alhajah@gmail.com.

In a semi-supervised learning (SSL) methodology, a learning classifier uses labeled and unlabeled data [11] – [13] illustrated a complete analysis on the existing semi-supervised learning approaches in his survey. The framework of a SSL is applicable to clustering and classification [14]. The idea of learning unlabeled and labeled data has gain attraction in the field of machine learning [15]. A few proposal and promising techniques have shown improvements in typical classification tasks [16].

A basic search engine comprises of a crawler, indexer and a searcher or query processor which will finally rank the results accordingly. A crawler as the name implies, crawl the web site and follows all the links that are available. It also consumes resources belonging to other organizations as well [17].

This research tends to take full advantage of the SSL and present a novel SSL framework utilizing the search engine architecture with the Breadth-First–Search (BFS) algorithm of data structures to retrieve web pages and the subsequent indexed links.

The organization of this article is sequenced as follow: Firstly, a web page is constructed with a starting seed .Secondly, a crawler is developed to exercise the concept of crawling, indexing, searching and ranking. Thirdly, an experiment is conducted to test all searching activities and subsequently leads to discussions and a conclusion is made.

II. Literature Review and Proposed Work

In a supervised learning methodology, a model is learned using a set of fully labeled items called the training sets [18]. This approach enables novice programmers the ability to apply machine learning paradigm to solve uncontrolled variables faced in an unsupervised learning environment [19]. This learning method has become apparently important due to the rate of growth of data which has increased tremendously and challenge our capacity to write software algorithm around it. This information overloading has become one of the pressing research problem [20].

One of the goals of supervised learning which is synonym to Learning with a Teacher [21] or Inductive Machine Learning [22] is to build a generic search engine which can learn the mapping between the input and a pre-determined set of output. The implementation of this application will enable the next cohort of students to clearly understand the working mechanics of a search engine and proceed to the next stage of their learning process [23].

A semi-controlled environment to develop a search engine

is proposed to enable search engineers to understand the algorithm of a generic search engine. A typical search engine is comprised of spiders (crawlers), indexers, retrieval and ranking and the user interface [24].

Most of the search engines whether they are used in the industries (remain as trade secrets for their algorithms) or academia employ well established algorithms and techniques [25], [26]. The earliest and the de Facto searching approaches provided by the search engine are based upon the Boolean retrieval model [27] so that the search results exceed the users' abilities to absorb [28], [29]. Furthermore, there exists a plethora of classification algorithms that are used for supervised learning methodology.

Although most of the current retrieval models are still based upon keywords, many online users are inclined to establish searching process according to the high-level semantic concept [30]. This study uses the Breadth-First Search (BFS) algorithm retrieval strategy to retrieve pages from sites which are topic-specific or topical searching. The BFS processes each node of binary tree in level by level increasing fashion and left to right in each level [31]. The generic algorithm of a BFS is illustrated below:

1. Get the URLs seed.
2. Put them in the queue.
3. For every URL in the queue, download page and extract all URLs in the page.
4. Put those URLs in the queue.
5. Iterate step 3.

The BFS is established by following hypertext links leading to pages which also allow for easier modification and enhancement of the codes in future. Using this function written in Python, the crawler can keep crawling continuously or deposit or save a list of un-indexed pages for later crawling avoiding the risk of overflowing the stack.

This function loops through the list of pages, calling addtoindex on each one. It then uses Beautiful Soup, which is a Python library for pulling data out of HTML and XML files to get all links on that page and adds their URLs to a set called newpages. At the end of the loop, newpages becomes pages, and the process rolls over again. The function, is_indexed, will determine if a page has been indexed recently before adding it to newpages.

Now that we have learnt how the indexing process works, next is to decide the best algorithm for the indexing process for this study. We will be using inverted indexing which stores a mapping from content such as words or numbers to its locations in a database file. Theoretically, creating an inverted index for a set of documents D, each with unique ID doc_id involves all codes that are available in [32] written in Python which is concise, easy to extend, multi-platform and it is free.

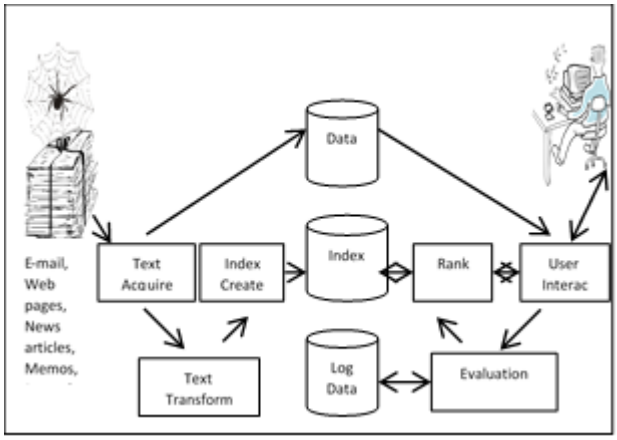


Fig. 1 Generic search engine architecture

III. Experimental Result And Analysis

A web site is constructed and uploaded onto the university's server. This uniform resource locator (url) is often referred as the anchor tag or the initial url seed. The anchor tag is identified and the subsequent activities will commence until it reaches the final destination of displaying all the pre-determined links.

In this experiment, we used a PC with 1.66 GHz Intel processor with 2 GB RAM and Notepad as the editor for the coding of HTML, PHP and a Python language and MySQL as the DBMS to store the links. For the search engine interface, there should be at least three pages; a crawler's page for crawling URLs and saving them into an index file, a search page for submitting queries and the result page for returning results of the search query. In relation to these, we will also need a website with lots of links to allow the search engine to work and of course, these will require browsers to view the web pages. Any supported browsers will do such as Microsoft IE 7 or above, Mozilla FireFox 3.0 or above, Google Chrome among others.

This section presents the results based on several test cases conducted to monitor the crawler's performance. The first testing is to make sure that the Rainbow website is connected to the internet. It then displays all external links and the results from the crawling process can be generated. These results are analyzed through the use of explanations and screenshots of system behavior and capabilities that forms the basis of the derive core results. The evaluation of the result is done by comparing system output with user expectations using a collection of data and queries.

The 'About' webpage includes another four web pages which forms the basis in understanding the Links, Crawl, Index and Search concepts as shown in Figure 3. The tests results from these two figures are shown in the next section.

Fig. 2 Linking, Crawling, Indexing and Searching buttons

A. Test Results

Link

The (Link) button in Figure 2 is used to show all internal links to pages. Unlike the external link, which requires internet connection to get its contents. This link does not require users to be connected to the internet. Clicking the button will bring user to a page providing internal links, when clicked goes to another page and goes on until the end of link is reached.

Crawl

The (Crawl) button requires users to view the crawled URLs. When the button is clicked, the page will display URL list showing the total number of crawled URLs coming from the internal as well as the external sites. Testing showed that this function worked successfully because the crawler has managed to list out a total of 19 URLs crawled within 10 seconds as shown in Figure 3 below.

All tables and figures you insert in your document are only to help you gauge the size of your paper, for the convenience of the referees, and to make it easy for you to distribute preprints.

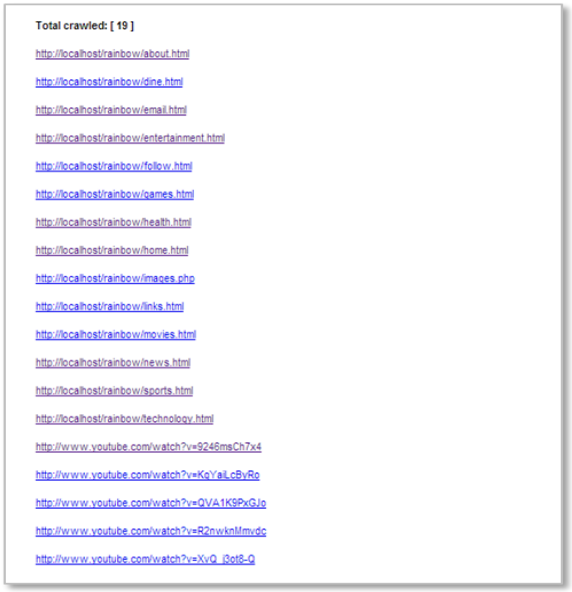


Fig.3 Crawled URLs lists

Index

The (Index) button requires users to crawl and index all internal and external websites from Rainbow. When this button is clicked, it will first prompt user to enter the webpage to be crawled. Having entered the starting page, the crawling and indexing process will start, taking URLs (no words, phrases, images etc.) from every page as shown in Figure 4, because it was designed to index only links. The indexed URLs is then saved into a 'Pages' table in 'Rainbow' database as shown in Figure 9, waiting for users to do search. Depending on how big the website is, there could have hundreds of thousands of web pages indexed and then they are all searchable on the website. Testing shows that crawling and indexing process is functioning because the URLs are added to the index table in the MySql database.

Fig. 4 Crawling and indexing process

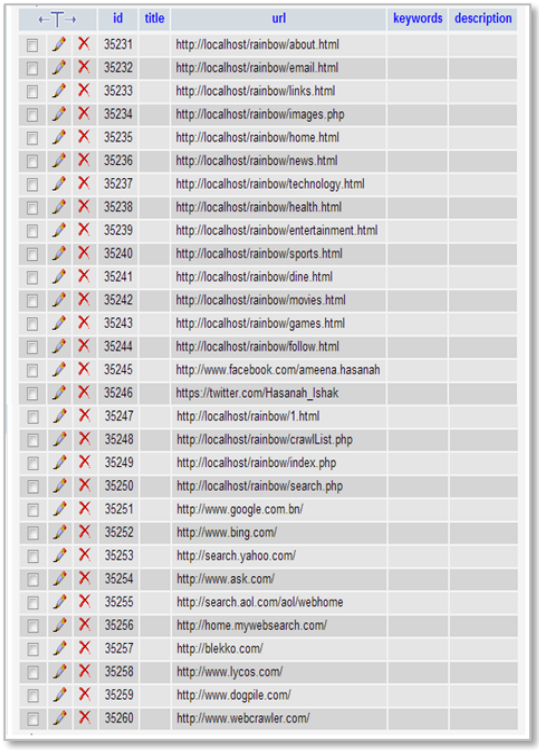
Search

The (Search) button requires users to type in any keyword on the search text box corresponding to the URL name because the indexing process is design to index URLs as keywords. When users perform search, the words they entered in the search box are compared with the indexed URL stored in the database. Results from executing this query will return the following outputs as shown in Figure 5.

Based on several outputs from the system, this crawler has proved to be functioning successfully generating outputs matching queries submitted by users with data stored in the index tables. Although this is just a simple web crawler with few basic functions, the crawling, indexing and searching process have produced information to user when searching process is called. In some situations, during searching process, some pages failed to open file contents deriving from the external sites. These problems have affects the effectiveness of the search engine in which the quality of output is not up to the standard of any search engine. A more detailed evaluation on these is explained in the discussion section of this paper.

IV. DISCUSSION

Based on few test cases, the results show that the quality of output was not up to the standard of effectiveness due to the output quality. This is because the current version of this crawler was designed to index only URLs (links or pages) no words, titles, images, videos or anything else to be used as keywords for the search process. Using URL's resource name alone is not good enough as they are poor descriptions of actual information needs. This could be improved and there are many ways for improvement to produce better results and one way is to have the document indexed on its contents, titles, metadata, images, links etc. so that user can have a broad view of the keywords to be searched. Since this current version of crawler is only used for learning process, in future it can be utilized further to enable document to be indexed on title, images or words, links etc. on the page.



id	title	url	keywords	description
35231		http://localhost/rainbow/about.html		
35232		http://localhost/rainbow/email.html		
35233		http://localhost/rainbow/links.html		
35234		http://localhost/rainbow/images.php		
35235		http://localhost/rainbow/home.html		
35236		http://localhost/rainbow/news.html		
35237		http://localhost/rainbow/technology.html		
35238		http://localhost/rainbow/health.html		
35239		http://localhost/rainbow/entertainment.html		
35240		http://localhost/rainbow/sports.html		
35241		http://localhost/rainbow/dine.html		
35242		http://localhost/rainbow/movies.html		
35243		http://localhost/rainbow/games.html		
35244		http://localhost/rainbow/follow.html		
35245		http://www.facebook.com/ameena.hasanah		
35246		https://twitter.com/Hasanah_Ishak		
35247		http://localhost/rainbow/1.html		
35248		http://localhost/rainbow/crawlList.php		
35249		http://localhost/rainbow/index.php		
35250		http://localhost/rainbow/search.php		
35251		http://www.google.com.bn/		
35252		http://www.bing.com/		
35253		http://search.yahoo.com/		
35254		http://www.ask.com/		
35255		http://search.aol.com/aol/webhome		
35256		http://home.mywebsearch.com/		
35257		http://blekko.com/		
35258		http://www.lycos.com/		
35259		http://www.dogpile.com/		
35260		http://www.webcrawler.com/		

Fig. 5 Crawled and indexed URLs saved in database

Processing time is the time taken to crawl and index websites whereas response time is a delay between a user submitting a query and the time when results are received, usually measured in milliseconds (ms). Processing and response time are again determined by few things such as the speed of the processor and how huge the index file is. For this project, I have tested the crawler using a 1.66 GHz Intel processor with 4 GB RAM and it came out that the processing time for the crawling and indexing processes are not good enough. This is because the indexing algorithms for this crawler's version are made simple by indexing only one field which is the URL field. If the indexing is designed to do full-text search or other types of indexing such as titles, images, videos, links etc. this would take longer time which again depends on the size of the site contents.

With this simple crawler, it is able to crawl the web pages for 15 minutes and downloaded 9853 links from the Rainbow website. In order to maintain a search engine corpus of say, ten billion web pages, in a reasonable state of freshness, say with pages being refreshed every 4 weeks on average, the crawler must download over 4,000 pages per second [33]. This can be achieved using distributed crawlers over multiple computers and each crawling machine must pursue multiple downloads in parallel and this would likelihood overload and crash that web server. Therefore, politeness policies should be implemented to limit the amount of traffic directed to the web server.

So in this case, the performance of the Rainbow crawler is far slow because it was not designed for distributed crawling. If it is to crawl continuously until the end, it would probably take hours or even days to finish crawling because it involves crawling to external sites as well. Performance will only be slow in areas where indexing images, videos or full-text are involved but this crawler only use URLs as the index field. Therefore, this crawler should perform faster than expected. On the other hand, the response time taken when submitting queries and receiving results is quite fast because the table has been indexed which makes it easy to retrieve specified URLs.

The strength of this current version of search engine can be seen in its capability to accomplish the crawling, indexing and searching process for information. Based on previous discussions, there are rooms for improvement in order to make the search engine more powerful and functional by first implementing the ranking algorithm which was not included in the system. One of the most important steps in improving the website ranking in any search engine is to ensure that it contains plenty of rich information that includes relevant keywords that indicate the subject matter of the page's content. From here, then we can suggest to improve the performance of the search engine by implementing the followings.

For system effectiveness, that is improving the quality of output, use headings to summarize page contents and include accurate description of page content in metadata. Apart from these, full-text search can also be used that is breaking a block of unstructured text into individual words, phrases, and special tokens. With these, a wide category of indexed keywords are stored in the index file to enable users to find relevant searches which are far better than using only descriptive URLs as keyword as in this crawler. Besides splitting documents into keywords, ranking can also improve the search quality. This can be done in many ways and the most common ones are using the location and frequency of keywords on the web page. Those with higher frequency are deemed to be more relevant than those with lower frequency.

On the other hand, in order to improve the efficiency of the crawler in the searching process, we can include a stopping function which is removing common words such as 'and', 'or', 'the', 'in' to reduce the size of the database. These words can be a problem for some queries as these slows down indexing process if full-text search is used. Apart from stopping, a stemming function should also be implemented which is removing words that are derived from a common stem such as 'computer', 'computing', 'compute'. Using a morphological analysis in stemming is also useful which is collapsing variations of a given word into one index term; for example, treating 'mice' and 'mouse', or 'electrification' and 'electric' as the same word.

Both word-level transformation such as the stopping and stemming, can be included in the crawling and searching algorithms. Although it is not quite effective for some queries but it can improve the search performance in some situations. Other than these, the crawling algorithms can also be improved to enable politeness policies that is obeying robot.txt and controlling access to servers by following the 'Allow' and 'Disallow' rules that dictates which resources the crawler is allowed to access. Also try to avoid using Frames, Flash or Javascript as these can slow down the crawler's

performance. Lastly, we could include spam pages detection as spamming can affect the efficiency of search engines and more seriously, the effectiveness of the results.

V. CONCLUSION

This study has demonstrated a generic search engine which forms the core of applications that facilitate information retrieval process in a SSL environment. It employed a BFS algorithm of data structures for retrieval strategy which allows future code modifications and enhancements. This proposed methodology uses all available computing resources efficiently as compared to most tasks performed by high end servers.

The goal of the SSL is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. SSL is of great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data are scarce or expensive. SSL also demonstrates potential as a quantitative tool to understand human category learning, where most of the input is self-evidently unlabeled.

The development for this prototype will allow students to develop and enhance algorithms to the next level of their learning process. This will further enhance their abilities to produce new algorithms and laid a ground-work in innovation and creativity. The SSL method performed very well utilising the search engine capabilities for fine sub-classification. Future direction of this study can be further explored in other learning paradigm such as generative semi-supervised method, discriminative semi-supervised method or other hybrid learning methodologies.

REFERENCES

[1] Bing Inc., http://www.bing.com.
[2] Google Inc., http://www.google.com.
[3] Yahoo! Inc., http://www.yahoo.com.
[4] A.A. Mishra and C. Kamat, "Migration of Search Engine Process into the Cloud," *International Journal of Computer Application*, vol. 19, no. 1, April 2011.
[5] E.J.Glover, S. Lawrence, M.D. Gordon, W.P. Birmingham, and C.L. Giiles, "Web Search Your Way", *Communications of the ACM*, vol. 44, no. 12, pp. 97-102, 2001.
[6] H. Chen, P. Buntin, L. She, S. Sutjahjo, C. Sommer and D. Neely, "Expert, prediction, symbolic learning and neural networks: An experiment on Greyhound racing", *IEEE Expert*, vol. 9, no. 2), pp. 21-27, 1994.
[7] G.W. Furnas, T.K. Landauer, T.K., L.M. Gomez, and S.T. Dumais, "The Vocabulary Problem in Human-System Communication", *Communications of the ACM*, vol. 30, no. 11, pp. 964-971, November 1987.
[8] G. Kumar, N. Duhan and A.K.Sharma, "Page Ranking Based on Number of Visits of Links of Web Pages", *in Proc. of Computer & Communication Technology (ICCT)*,2011, pp. 11-14.
[9] K. Nigam, A. McCallum, S. Thrun and T. Mitchell, *Text classification from labelled and unlabeled documents using EM*, Machine Learning, Boston:Kluwer Academic Publisher, 2000, pp. 1-34.
[10] M.R. Amini, and P. Gallinari, "The use of unlabeled data to improve supervised learning for text summarization", *SIGIR*, Tampere, Finland, Aug. 2002.
[11] X.H. and W.S. Lee, "Hyperparameter learning for graph based semi-supervised learning algorithms", *in Proc.of the NIPS*, 2006.
[12] Z.H. Zhou and M. Li, "Semisupervised regression with co-training style algorithms*", in Proc. of the TKDE,* 2007.
[13] X.J. Zhu, *Semi-supervised learning literature survey* Technical Report (1530), Dept. Comp. Sci., Univ. Wisconsin-Madison, USA, 2006.
[14] S. Basu, M. Bilenko, and R.J. Mooney, "A probabilistic framework for semi-supervised clustering", *in Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.
[15] M. Seeger, *Learning with labeled and unlabeled data*, Technical Report, University of Edinburgh, 2001.
[16] X. Zhu, J, Lafferty and Z. Ghahramani, "Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions", *ICML 2003 Workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, 2003.
[17] M. Koster, "Robots in the web: threat or treat", *ConneXions*, vol. 4, no. 4, Apr. 1995.
[18] W.B. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*, Pearson Education Inc., 2010, pp. 344.
[19] M.N. Rahman,A.H., Seyal, M.S. Omar, and S.A. Maidin, "A search engine development utilizing unsupervised learning approach", in Proc. of the 4th International Conference on Soft Computing, Intelligent Systems & Information Technology (ICSST), Bali, Indonesia, 11-15 Mar. 2015.
[20] C.M. Bowman, P.B. Danzig, U. Manber and F. Schwartz, "Scalable Internet Resource Discovery: Research Problems and Approaches", *Communications of the ACM*, vol. 37, no. 8, pp. 98-107, Aug. 1994.
[21] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall PTR, 2nd Edition, 1998.
[22] S. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", *Informatica Journal*, vol. 31, pp. 249-268, 2007.
[23] M.N. Rahman, A.H. Seyal, S.A. and Maidin, S.A. (2014) "Search Engine Development: Adaptation From Supervised Learning Methodology", *In Electronic Proc. of The Fourth International Conference on Digital Information Processing and Communications (ICDIPC2014), as part of The Second World Congress on Computing and Information Technology (WCIT2014*), Kuala Lumpur, Malaysia,8 – 20 Mar. 2014, pp. 35-42.
[24] M. Chau, H. Chen, J. Qin, J., Y. Zhou, Y. Qin, Y., W.K Sung and D. McDonald, "Comparison of two approaches to building a vertical search tool: a case study in the nanotechnology domain", *in Proc. of the 2nd ACM/IEEE-CS joint conference on digital libraries*, ACM, Jul. 2002, pp. 135-144.
[25] G. Salton, *Automatic Text Processing*, Reading, MA: Addison-Wesley, 1989.
[26] C. Faloutsos., "Access Methods for Text", *ACM Computing Surveys*, vol. 17, no. 1, pp. 48-74, Mar. 1985.
[27] V.I. Frants, J. Shapiro, L. Taksa, I. and V.G. Voiskunskii, "Boolean Search: Current State and Perspectives", *Journal of the American Society of Information Science*, vol. 50, no. 1, pp. 86-95, 1999.
[28] M. Chau and H. Chen, "A Machine Learning Approach to Web Page Filtering Using Content and Structure Analysis", *Decision Support 9ystems*, Elsevier, vol. 44, no. 2, pp. 482-494, Jan. 2008.
[29] S. Jung, J.L. Herlocker and J. Webster, "Click Data as Implicit Relevance Feedback in Web Search", *Information Processing & Management*, Elsevier, vol. 33, pp. 791-807, Mar. 2007.
[30] J. Yi, "The Research of Search Engine Based on Semantic Web", Intelligent Information Technology Workshop (IITAW), International Symposium, pp. 360-363, 2008.
[31] G. Brassard and P. Bratley, *Fundamentals of Algorithms*, New Delhi, India:PHI Publications, 1st Edition, 2008, pp. 303-305.
[32] T. Segaran, *Programming Collective Intelligence: Building Smart Web 2.0 Applications*, O'Reilly Media Inc., First Edition, 2007
[33] Najork, M. (2009). *Web Crawler Architecture*. Encyclopedia of Database Systems