An Architectural Framework of a Crawler for Retrieving Highly Relevant Web Documents by Filtering Replicated Web Collections

Shashi Shekhar, Rohit Agrawal
GLA Institute of Technology & Management
Mathura, India.
shashi.shekhar13@gmail.com ,rohit_agwl@rediff.com

Karm Veer Arya
Indian Institute of Information Technolgy & Management
Gwalior, India.
kvarya@gmail.com

***ABSTRACT*— As the Web continues to grow, it has become a difficult task to search for the relevant information using traditional search engines. There are many index based web search engines to search information in various domains on the Web. By using such search engines the retrieved documents (URLs) related to the searched topic are of poor quality also as the amount of Web pages is growing at a rapid speed, the issue of devising a personalized Web search is of great importance. This paper proposes a method to reduce the time spend on browsing search results by providing a personalized Web Search Agent (MetaCrawler). In the proposed technique of personalized Web searching, Web pages relevant to user interests will be ranked in the front of the result list, thus facilitating the user to get a quick to get access those links ranked in the front of the list. An experiment was designed and conducted to test the performance of proposed Web-Filtering approach. The experimental results suggest substantial improvement in the crawling strategy, especially when the search strings are small.**

Key Words*— Link analysis, Search result ranking, Web IR, Web page classification, Web crawler.

I. INTRODUCTION

As we all know, search is very important to network applications and it looks these days that we cannot live in the Internet without search engines. As the volume of web pages increasing rapidly, users increasingly use search engine to find specific information. The traditional search engines index only surface web whose pages are easily found. The major focus is now on invisible or hidden web. Recent literature shows that large part of the Web is available behind search interfaces and is reachable only when users request for the digital information using an optimized searching techniques.

As the volume of information grows, there is a need for tools and techniques to utilize efficient searching by filtering the replicated links containing same information .It has become very important to automatically search and

filter the Web links through an easy to use interface. The traditional Web Crawlers traverse the web by following hyperlinks and storing downloaded pages in a large database that is later indexed for efficient execution of user queries. They are the essential components of all search engines and becoming increasingly important in data mining and other indexing applications.

There has been some recent academic interest in new types of crawling techniques, such as focused crawling based on semantic web[1,3], cooperative crawling [5], distributed web crawler [2], and intelligent crawling [4],and the significance of soft computing comprising fuzzy logic (FL), artificial neural networks (ANNs), genetic algorithms (GAs), and rough sets (RSs) highlighted [6], but less work on the filtering and processing of the tremendous data in terms of URLs from the crawler

The main objective in this paper is to propose a Metacrawler framework for searching and filtering web collections using a multi agent model approach. The remainder of this paper is organized as follows: Section 2 discusses on the work related to Web crawler. In Section 3, the architectural framework of the Meta crawler is described. In Section 4, experimentation and evaluation details are discussed. Finally in Section 5, conclusion and future directions are presented.

II. RELATED WORK

Usually, agent based Web mining systems can be classified under three categories [7]: intelligent search agents, information filtering / categorization and personalized Web agents. Several intelligent Web agents like Harvest[8], ShopBot[9], iJADE Webminer[10], have been developed to search for relevant information using domain characteristics and user profiles to organize and interpret the discovered information. An adaptive Web search system based on reactive architecture has been presented [11]. An adaptive meta search engine was developed based on neural network based agent [12] which improves search results by computing user's relevance feedback.

The pioneering work on hidden Web crawler design [13] focused on extracting content from searchable electronic

978-0-7695-4058-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ACE.2010.64

databases. They introduced an operational model of HiWE (Hidden Web crawler). The paper discusses on different directions on research in designing the crawler for content extraction. In [14], useful observations and implications are discussed about hidden Web. Their survey reports on locating entry points to hidden Web, coverage of deep Web directories and so on. They also give a clear observation that the crawler strategy for deep Web are likely to be different from surface Web crawlers. In [15], form focused crawler for hidden Web is described which utilizes various classifiers to extract relevant forms. An adaptive strategy based crawler design is discussed in [16].

III. PROPOSED FRAMEWORK

In the proposed work the crawler invoke underlying search engines (also referred as component search engines) with the user query as a parameter, retrieve information relevant to the query, unify the individual result sets into a single ranked list, and would rank them according to relevance. Different component search engines may accept queries in different formats; the user query may thus need to be translated into an appropriate format for each local system. Result sets are likely to be a list of document identifiers, such as Uniform Resource Locators (URLs) for web pages.

A. Framework Component Architecture

Apart from the underlying search engines, the proposed crawler mechanism has four primary software components: an Engine Selector, Query Dispatcher, Duplicate URL Eliminator and a Result Merger. Proposed Framework Architecture of crawler with its components is illustrated in Figure 1. The numbers on the edges indicate the sequence of actions for a query to be processed.

The process of identifying and selecting potentially useful search engines to search for a given query is known as Engine selection. Engine selector component is responsible for selecting search engine for the Meta crawler. The role of Engine selector is more pronounced when there is a long list of component engines that need to be queried.

The query dispatcher is responsible for establishing a connection with each selected search engine and passing the query to it. The original query may need to be translated according to the search application programming interface (API) used by respective search engines. Accordingly the threads will be created and passed to their respective Search Engine Repository.

The Duplicate URL Eliminator module determines whether an extracted link is already in the URL list (log.lst file) or has recently been fetched. Finally, the URL is checked for duplicate elimination: If the URL is already in the log.lst file, we do not add it to the list file. This checking is being done by a dedicated thread. This thread is generally quiescent except that it wakes up once every few seconds to log crawl progress statistics (URLs crawled, list size, etc.), and thus eliminates the redundant URLs.

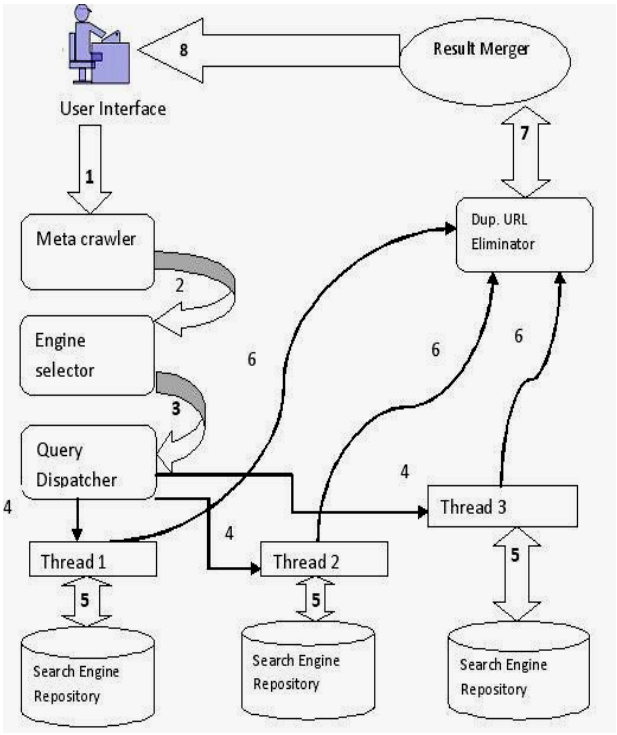

Figure 1. Proposed Web Crawler Component Architecture

The Result merger combines the results into a single ranked list. This component has the most influence on retrieval effectiveness. Here the result merger would rank all the returned documents in descending order of their global similarities with the user query. With context to our paper, we have adopted an iterative algorithm for URL retrieval, checking duplicacy and URL merging. The proposed algorithm is given in pseudo code as follows:

Step 1: Let s is the number of search engines (seed engines) used.
In this experiment value of s is taken 3.
Step 2: Let t is the number of URLs retrieved by the search engine. Set t= 0(initial value)
Step 3: Search the query on crawler framework by creating threads for the engines used:
For each engine s generate thread t
If t ε s **then** retrieve the URL
and increment t by 1; t←t+1
Step 4: Add URLs to the .lst (List)
Step 5: Scan .lst file for duplicacy.
If duplicacy occurs **then** remove link from t and t←t-1
Step 6: Repeat Step 3 through Step 5 until t required number of URLs is Retrieved
Step 7: Extract URLs from web and add to t till t=1000

IV. PERFORMANCE EVALUATIONS

The performance of proposed crawler is evaluated at four different levels. Firstly proposed web crawler retrieval effectiveness is measured using TREC Style Average Precision (TSAP) methodology. Here the precision value for five different query levels is calculated for each specific search engine and that of proposed crawler architecture. The average precision is also calculated for five different queries and is compared with the average precision of the proposed crawler.

Secondly the performance of proposed crawler is evaluated on the basis of retrieval relevance. The relevance ratio is calculated for five different query levels on different search engines and is compared with the relevance ratio of the proposed crawler.

In the third, phase the proposed web crawler performance is checked on the basis of different attributes: *Number of Dead Links, Number of Relevant Links* and *Number of Redundant Links.* This is one of the important evaluation criteria to test the performance of the proposed architecture. As these three attributes will not only judge the crawler efficiency but also makes it better than the search engines. Below section describes the evaluation pattern of the crawler according to the above discussed conditions.

Finally the crawler performance is checked on the basis of maximum number of Links retrieved for a given set of query. This is being done to reduce the time used for navigating different web documents for given query. As the searched query is available in thousands of documents, this measurement is helpful in comparing the proposed crawler framework against search engines.

A. Retrieval Effectiveness using TSAP

The TSAP values for three different search engines at five different query levels have been computed and comparison with proposed crawler is given in Figure 2.

$$TSAP \quad @ \quad N \quad = \frac{\sum_{i=1}^{N} r_i}{N} \qquad (1)$$

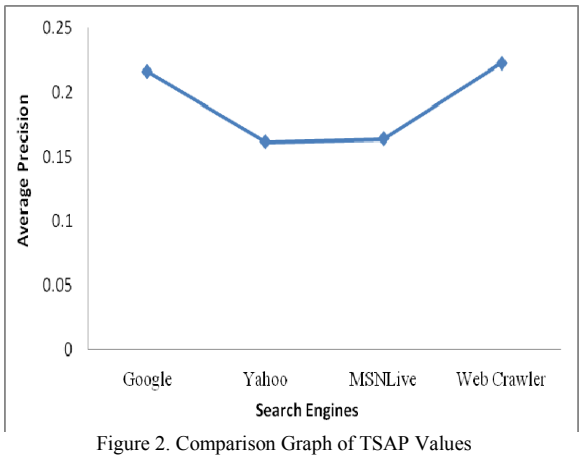

Figure 2. Comparison Graph of TSAP Values

The Figure 2 shows that the average precision value calculated by equation 1 has a higher value for proposed web crawler as compared to three search engines. The proposed web crawler has the highest precision value, where as Yahoo has the minimum value. The proposed web crawler has given the enhanced performance in terms of retrieval effectiveness and outperforms compared to the most popular engine Google.

B. Relevance Ratio

In order to illustrate the relevancy in results obtained by proposed crawler and to those of Google, Yahoo, and MSNLive, The relevance ratio achieved by the experimental setup is defined as follows.

$$\text{Relevance Ratio} = \frac{\text{Number of Relevant URLs}}{\text{Total Number of URLs Retrieved}} * 100 \qquad (2)$$

The relevance ratio is calculated using equation 2 and is illustrated in Figure 3, which shows the relevance ratio of each of the component search engine and our proposed web crawler.

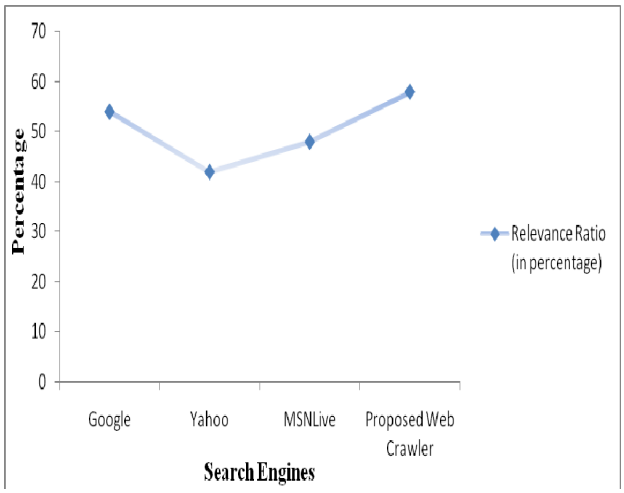

Figure 3. Relevance Ratio

The Figure 3 shows comparative improvement in the relevance ratio by the proposed crawler against the component search engines. The proposed web crawler has better performance and can enhance quality of search results.

C. Retrieval Attribute Comparison

The performance is compared on the basis of certain attributes and top 100 documents retrieved by each of the search engines and proposed web crawler are collected. The retrieval accuracy is tested on the basis of three attributes:

- Number of Dead Links Retrieved
- Number of Redundant Links Retrieved
- Number of Relevant Links Retrieved

The below section illustrates the query average comparison in Figure 4 for four different queries executed on the three search engines and on our proposed web crawling Framework.

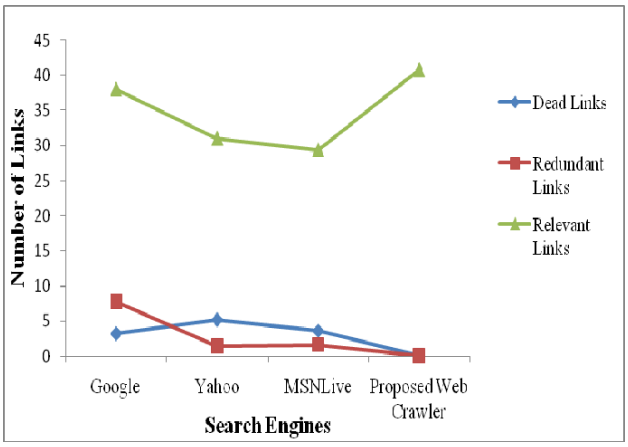

Figure 4. Attribute Comparison

The above result graph shows the comparative analysis on the basis of attributes of the web documents retrieved by the three engines and by our proposed web crawler on the average of four different queries. The proposed web crawler has the minimum value for Number of Dead Links and Number of Redundant Links as compared to three popular search engines. The graph also describes the Number of Relevant Links retrieved by the search engines and by the proposed web crawler. The web crawler has the maximum relevancy as compared to other three search engines.

D. Links Retrieval Comparison

We have evaluated the performance of the proposed crawler using maximum number of links retrieved for a query. Five different queries are passed to the three popular search engines (Google, Yahoo, and MSNLive) and on proposed web crawler. Here we are comparing the maximum number of real links retrieved by the search engines against proposed web crawler. The comparison graph for maximum number of links retrieved is shown in Figure 5.

The Figure 5 shows the comparative results based on five different queries issued on proposed web crawler and on three search engines. The graph represents the maximum number of links retrieved for the five different queries.

Comparing the performance of link retrieval, it is clear from the Figure 5 that the proposed web crawler and Yahoo have the maximum number of link retrieval as compared to Google and MSNLive. The proposed web crawler has the linear performance as shown by a straight line having maximum number of links for all the five different queries.

We have evaluated the performance of the proposed web crawler by conducting four different types of experiments. The above experimental results illustrates that the proposed crawling architecture improves the retrieval relevancy by retrieving fresh and non replicated web links early in the crawling process.

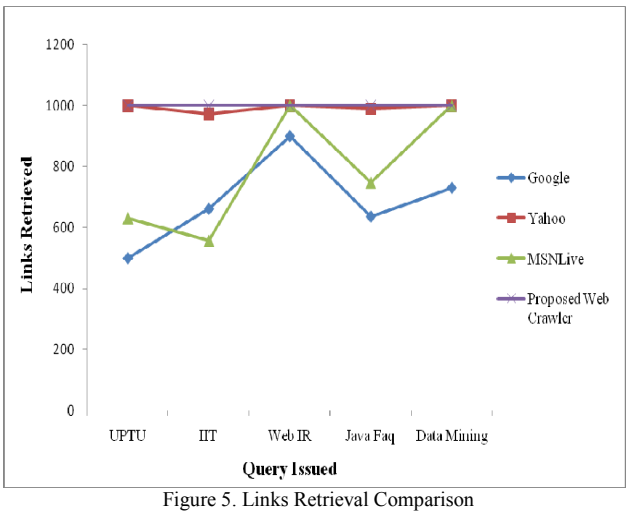

Figure 5. Links Retrieval Comparison

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a modular framework of crawler based on service oriented architecture. The framework supports dynamic addition and removal of participant search providers. A reference implementation of the framework has been done, to demonstrate flexibility of the architecture. Individual components like engine selector, query dispatcher, result merger and duplicacy eliminator can be seamlessly reconfigured to produce highly customized search systems without significant effort. A prototype crawler has been developed to demonstrate the flexibility of the framework with three commercial search providers as component search engines.

Next generation search technologies are increasingly focusing on the personalization of search, to handle the explosive growth of information in all walks of human knowledge. The crawler framework can be a good choice for testing new approaches in providing highly relevant personalized search results. The framework itself can be augmented and extended in several ways. New crawling algorithms that take structured queries for processing can be formulated, to provide highly relevant personalized clustered search results for every user.

REFERENCES

[1] V. Loia, W. Pedrycz, and S. Senatore. "Semantic Web Content Analysis: A Study in Proximity-Based Collaborative Clustering." Proceedings of the 18th International Conference on Data Engineering (ICDE'02).

[2] V. Shkapenyuk, T. Suel. "Design and Implementation of a High-Performance Distributed Web Crawler." International World Wide Web Web Conference,2001

[3] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu. "Intelligent crawling on the world wide web with arbitrary predicates." Proceedings of the 10th International World Wide Web Conference, May 2001.

[4] S. Altingovde and O. Ulusoy. "Exploiting interclass rules for focused crawling." IEEE Intelligent System, 2004

[5] M. Buzzi. "Cooperative crawling." Proceedings of the First Latin American Web Congress (LA- WEB 2003), 2003

[6] S. K. Pal. V. Talwar. "Web Mining in Soft Computing Framework: Relevance, State of the Art and Future Directions". Proceedings of IEEE Transactions on Neural Networks, Vol. 13, No. 5, 2002.

[7] J. Srivatsava, B. Mobasher, R. Cooley, "Web mining: Information and pattern discovery on the world wide Web." International conference on tools with Artificial Intelligence, pp558-567, Newport beach, 1997.

[8] C.M. Bowman, P.B. Danzig, U. Manber, M.F.Schwartz," Scalable internet resource discovery: Research problems and approaches" Communications of the ACM, 37(8), 98-107,1994.

[9] R.B. Doorenbos, O.Etzioni and D.S. Weld, "A scalable comparison shopping agent for the world wide Web" Technical report 96-01-03, University of Washington, Department of CSE,1996.

[10] S. Chakrabarti, B..Dom, S.Ravikumar, P. aghavan, S. Rajagopalan, A. Tomkins, D.Gibson, J. Kleinberg, " Mining Web's Link Structure", IEEE Computer, pp60- 67, 1999.

[11] F. Gasparetti, A. Micarelli, " Swarm Intelligence :Agents for Adaptive Web Search", Technical Report, Dept. of Information,University of ROMA, Rome, Italy, 2000.

[12] Y. Xie, D. Mundlura, V.V. Ragahvan, "Incorporating Agent based Neural Network Model for Adaptive Meta Search", The centerfor Advanced Computer studies, University of Louisiana, 2004.

[13] S. Raghavan, H. G.Molina, "Crawling the Hidden Web", Proc. Of the 27th VLDB Conference, 2001.

[14] K Chen, B. He, C. Li, M. Patel, Z. Zhang, "Structured databases on the Web: Observations and Implications", Technical Report, UIUC.

[15] L. Barbosa, J. Freire, " Searching for hidden-Web databases", Eighth Intl. workshop on the Web and Databases, 2005.

[16] L. Barbosa, J. Freire, "An Adaptive Crawler for Locating Hidden-Web Entry points ,Proc of Intl WWW conf,p441-450, 2007.