ISSN:0975-9646

Focused and Deep Web Crawling-A Review

Saloni Shah, Siddhi Patel , Prof. Sindhu Nair

Dept of Computer Engineering, D.J.Sanghvi College of Engineering
Plot No.U-15, J.V.P.D. Scheme, Bhaktivedanta Swami Marg, Vile Parle (West),
Mumbai, Maharashtra 400056, India

Abstract— **In the exponentially growing Internet, web search engine companies needed to achieve scalability with large amount of hardware and network resources. For that they gave birth to a technique called focused web crawling to discover topic related information that can be used in online search. Focused crawlers dynamically browse the web by selecting the most favorable links to get maximum number of relevant pages with efficient utilization of bandwidth and time. Deep web is a vast repository in a web that are not always indexed by automated search engines. In this paper we are surveying the available techniques used for focused crawling and deep web crawling.**

Keywords— focused web crawaling, deep web crawling,search engines;

I. INTRODUCTION

Over the last 5-10 years the internet has become one of the main source of information. People use search engines to find any kind of information. Because of this, the need for finding more accurate information is also increasing. Search engines need to be up to date with the new pages that are created everyday. Due to diversity of content, the result becomes irrelevant when a specific query is run on general search indexed pages. Therefore the idea of focused web crawling came into picture.

Before the page is delivered to people as a result of particular search query, web search engines needs to find that pages from the trillions of pages available in world wide web. To find these pages, web search engines employs software robots called spider or crawlers and the process of finding these pages is called web crawling.

In the last several years, some of the more comprehensive search engines have written algorithms to search the deeper portions of the world wide web by attempting to find files such as .pdf, .doc, .xls, ppt, .ps. and others. Searching for this information using deeper search techniques and the latest algorithms allows researchers to obtain a vast amount of corporate information that was previously unavailable or inaccessible. Research has also shown that even deeper information can be obtained from these files by searching and accessing the "properties" information on these files.

1.1 Focused web crawler

Focused web crawling is the process of finding pages that are related to some specific topics or satisfy some particular property. Focused crawler tries to fetch as much relevant page as possible efficiently. The goal is achieved by, precisely prioritizing the already crawled pages and managing the exploration of hyperlinks. The topics could be anything for e.g. crawl pages with '.in' or '.us' domain, crawl pages about sports/news etc.

Focused crawling is a good approach to provide better search results because of the following two reasons:
1. The demand for topic-search engine is rising day by day. Collecting topic-specific information can be done much faster if smart crawling strategies are applied.
2. According to worldwidewebsize.com the current size of indexed web page Google has is approximately 50 billion and increasing at an exponential rate day by day. To crawl such a large web one needs really intelligent techniques.
There are various categories in focused crawlers:

(a) Classic focused crawler
(b) Semantic crawler
(c) Learning crawler

(a) Classic focused crawlers [3] follows the search towards interested pages by taking the user query. The topic which they want to search is user query. They assign priorities to the links based on the topic of query and the pages with high priority are downloaded first. Similarity between the topic and the page containing the links are used to compute the prorities. Similarity of text is computed using an information similarity model such as the Vector Space Model (VSM) [4].
(b) Semantic crawlers [3] are a variation of classic focused crawlers. Downloaded priorities are assigned to pages by applying semantic similarity criteria to compute topic to page relevance: the relevance of a page and the topic is defined by the sharing of conceptual terms. Ontology is used to define the conceptual similarity between the terms.
(c) Learning crawlers [5] uses a training process to guide the crawling process and to assign visit priorities to web pages. A learning crawler supplies a training set which consist of relevant and not relevant Web pages in order to train the learning crawler [5]. Links are extracted from web pages by assigning the higher visit priorities to classify relevant topic. Methods based on context graphs and Hidden Markov Models (HMM) take into account not only the page content but also the link structure of the Web and the probability that a given page (which may be not relevant to the topic) will lead to a relevant page .

The focused crawler starts it's task from few relevant pages and then follows more promising links to find relative pages. It uses the link structure but the order in which they are processed is important. Focused crawler must predict the probability that the page is related to some specific topic before downloading it. There are many algorithms that perform focused crawling. In this paper I

will describe them and also compare the results. It works as follows:

- At the beginning URL Queue is fed with some relevant seed URLs.
- Web page downloader fetches URLs from URL queue and downloads that page from Internet.
- Parser and extractor extracts terms and Hyperlinks from the downloaded pages.
- Relevance calculator finds the relevance of page based on topic and assigned score to parent URL.
- Topic Filter determines the relativity of downloaded page with the topic.
- If the page is found to be relevant then the URLs extracted from that page are added to URL queue, otherwise added to irrelevant table.

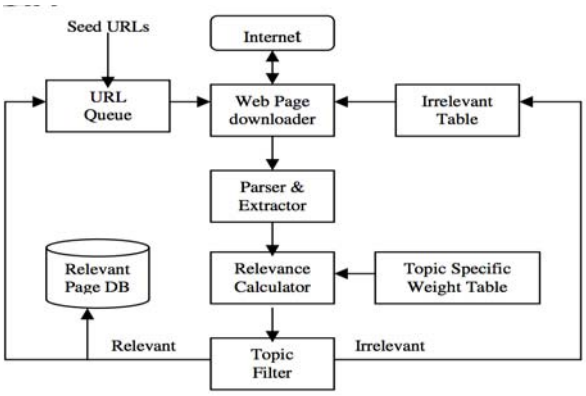

Fig. 1. Architecture of focused crawler

1.2 Deep web crawler

Deep web or invisible web or hidden web is part of World Wide Web that search engines cannot or will not index. Deep web consist of proprietary sites, sites with scripts, dynamic sites, ephemeral sites, sites blocked by search engine policy, sites with special format etc.

Shallow web or surface web is a part of Internet that is indexed by automated search engines. Deep web is 5 to 500 times as vast as the shallow web. So if search engines can only index 20% of the page the remaining 80% of the web is remained un-indexed. Therefore it is more important to crawl and index Deep web along with the shallow web.

Retrieving data from hidden Web sites has two tasks resource discovery and content extraction. The first task deals with automatically finding the relevant Web sites containing the hidden information. The second task deals with obtaining the information from those sites by filling out forms with
relevant keywords. It deals with locating relevant forms that serve as the entry points to the hidden Web data using a multi-agent based Web mining crawler. Finding searchable forms is useful in the following fields [6]
• Entry points for locating the deep Web
• Deriving source descriptions in the databases
• Form matching to find correspondences among attributes

The architecture of deep web crawler is shown:

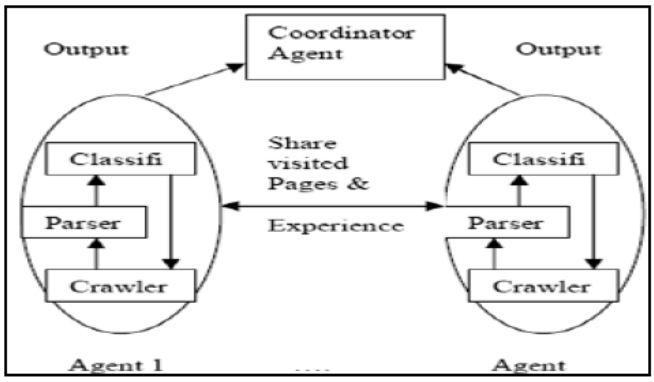

Fig. 2. Architecture of deep crawler

A .Crawler [2]-
Initially the crawler is given a set of URLs to crawl called seed URLs. The pages that are retrieved are given to the parser module. The classifier component gives out list of URLs that are identified as relevant. The links that are identified as promising links are placed in the frontier queue. A specified number of agents are spawned and each agent is positioned at one of the links and given an initial amount of energy to crawl.

B. Parser [2]-
This module gets a page from the crawler, analyzes for its relevancy to the specific subject of retrieval. This task uses similarity measure to calculate the relevance of the page on a particular topic in equation as stated in the paper. The links are extracted and sent to classifier module along with set of keywords, anchor text, and text around the hyperlink.

C. Classifier [2]-
This module deals with identifying a link that leads to searchable forms. There may be links whose single click immediately direct to a form. There may be links which give delayed benefit. To identify the links with immediate and delayed benefit reinforcement learning is employed. The reinforcement learning based multi-agents provide good results in terms of harvest rate and freshness [2].

II. Approaches For Focused And Deep Crawling

2.1 Approaches for focused web crawling

A. Without using background information

Focused Crawler was first proposed by DeBra94[1] which was based on “Fish-Search”. In that algorithm[1] each URL corresponds to a fish whose survival depends on relevance of visited page and speed of remote server. The algorithm marks pages either relevant or not relevant based on simple keyword match. The algorithm works as follows: seed URLs and query are given as input. It then builds a dynamic priority Queue of URLs. At each iteration a URL with higher priority (front of Queue) is popped and then processed. During the evaluation of page it determines if the page is relevant or not using simple keyword matching and based on this score algorithm decides whether to move

in that direction or not. Links extracted from this page are assigned a depth value. If the parent page is relevant depth is assigned some predefined value otherwise one less than the depth of parent's depth. URLs with depth more than 0 are inserted into priority queue. A school of fishes migrates in a direction of food (Here relevant pages). To find information that is not directly available in one hop, A fish dies when it traverses some threshold amount of irrelevant pages. Based on relevance and number of extracted links, fish produces offspring on every document. This algorithm follows Depth-First Search approach.

Later cho98 proposed an idea to calculate pageRank of a page and using that pageRank as a priority of the extracted URL in the URL Queue. This didn't produce much gain over traditional Breadth-First Search algorithms because the pageRank is calculated on very small non-random subset of web.

Hersovici[1] extended the "Fish-Search" algorithm into "Shark-Search" algorithm. To overcome the limitation of Fish-search, they proposed algorithm, which ranked the pages based on combination of relevance of source page, anchor text, neighboring pages of pre-defined size and inherent relevance score. Inherent relevance score is multiplication of score of parent page and decay factor. Relevance score is any number from 0 to 1 that is calculated based on similarity between topic and document in vector space. The paper Hersovici89 claims that the Shark-Search works 1.5 to 3 times better than the traditional Fish-Search.

The above mentioned algorithms worked without any background information. In the upcoming years improved ideas were proposed that uses some background information inorder to better predict the relevance of a page with respect to the query given.

B. Using background information

Chakrabarti99 built a system with three components: Crawler, Classifier and distiller. A supervised topic classifier[1] called 'learner' controls the priority of unvisited URLs in the URL queue. This classifier is trained on document samples available in topic taxonomy such as Yahoo! and from them it learns to label new documents as relative or not which in turn determines future link expansion. Negative classes are also considered to prevent the crawler shift into undesired topic category. Figure 2 below shows the architecture of focused crawler with classifier.

Further improvement on page relevance and URL priority model. The model for page relevance outputs whether the page is relative or not with the topic. The model for ranking URL called 'apprentice'[1] is an trained online by samples consisting of source page features and relevance of target page that defines the order of unvisited URLs. This approach claims improve 30% to 90% against false positive of chakrabarti99

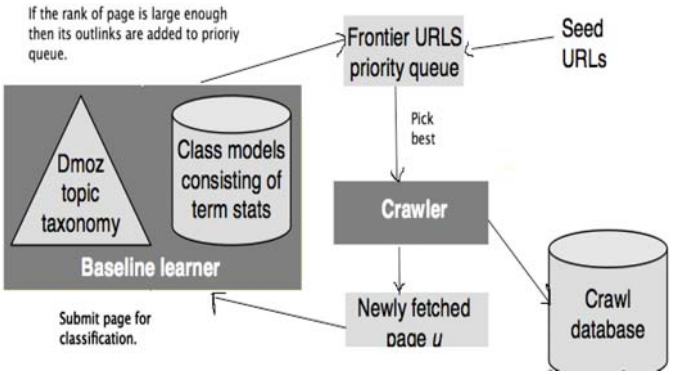

Fig. 3. Focused crawler contralled by classifier

The crawling task comprises of five components:

1) User Interaction: User provides input to crawler along with the ontology knowledge.
2) Web Crawling: Crawler starts with supplied information and pages according to their ranks.
3) Preprocessing: it has several steps. To normalize the text shallow text preprocessing techniques are used.
4) Ontology management: in order to move in right direction crawler relies on the ontological knowledge provided by human engineer.
5) Relevance computation: this is the heart of the crawler. It takes into account the natural language text, hyperlinks etc. to compute overall relevance score.

This kind of crawler showed significant improvement in harvest rate - the ratio of number of webpages crawled satisfying the crawling target to the number of pages retrieved, as compare to baseline focused crawler, which identifies page relevance by simple keyword match.

Later Bergmarc02 came up with an idea of adding tunneling to focused crawling. It believes that only to follow "Best-First" strategy to find relevant page is not necessarily optimal, sometimes it is necessary to go in the direction of series of not relative pages in order to get next relevant pages. Though this strategy may impact on efficiency, it can produce high-precision result in reasonable amount of time. This approach believes that a longer path history can impact on relevance of pages to be retrieved in future and therefore construct a document distance measure that takes considers parent page's distance and so on.

2.2 Approaches for deep web crawling

A. Single threaded crawler

The crawler[2] maintains a list of unvisited URLs called the frontier. The list is initialized with seed URLs which may be provided by a user or another program. Each crawling loop involves picking the next URL to crawl from the frontier, fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extract the URLs and application specific information, and finally

adding the unvisited URLs to the frontier. Before the URLs are added to the frontier they may be assigned a score that represents the estimated benefit of visiting the page corresponding to the URL. The crawling process may be terminated when a certain number of pages have been crawled. If the crawler is ready to crawl another page and the frontier is empty, the situation signals a dead-end for the crawler. The crawler has no new page to fetch and hence it stops.

Crawling can be viewed as a graph search problem. The Web is seen as a large graph with pages at its nodes and hyperlinks as its edges. A crawler starts at a few of the nodes (seeds) and then follows the edges to reach other nodes. The process of fetching a page and extracting the links within it is analogous to expanding a node in graph search. A topical crawler tries to follow edges that are expected to lead to portions of the graph that are relevant to a topic.

B. Multi threaded crawler

A sequential crawling loop spends a large amount of time in which either the CPU is idle (during network/disk access) or the network interface is idle (during CPU operations). Note that each thread starts by locking the frontier to pick the next URL to crawl. After picking a URL it unlocks the frontier allowing other threads to access it. The frontier is again locked when new URLs are added to it. The locking steps are necessary in order to synchronize the use of the frontier that is now shared among many crawling loops (threads). The model of multi-threaded crawler[2] follows a standard parallel computing model. Note that a typical crawler would also maintain a shared history data structure for a fast lookup of URLs that have been crawled. Hence, in addition to the frontier it would also need to synchronize access to the history. The multi-threaded crawler model needs to deal with an empty frontier just like a sequential crawler.

However, the issue is less simple now. If a thread finds the frontier empty, it does not automatically mean that the crawler as a whole has reached a dead-end. It is possible that other threads are fetching pages and may add new URLs in the near future. One way to deal with the situation is by sending a thread to a sleep state when it sees an empty frontier. When the thread wakes up, it checks again for URLs. A global monitor keeps track of the number of threads currently sleeping. Only when all the threads are in the sleep state does the crawling process stop. More optimizations can be performed on multi threaded model[2].

C. Gcrawler

The problem which exists in the traditional focused[2] crawler URL analysis model described previously is that the local optimal solution is often easily given in the process of searching the relevant pages according to the predetermined theme, namely only crawling around the related web pages, which results in some related web pages which are linked together through hyperlinks with lower degree of relevance are not crawled, then an effective coverage of the focused crawler reduces. The genetic algorithm is a global random search algorithm that based on the evolutionism and molecular genetics, whose prominent feature is the implicit parallelism and the capacity to make an effective use of the global information, and it can effectively find the global optimal solution jumping local optimum, which is the focused crawler URL analysis model needs. Figure 4 shows the Architecture of GCrawler[2].

But the genetic algorithm also has some disadvantages, for example, it cannot use the feedback in the system and lots of unnecessary redundancy iterations come out when the solutions reach to certain extent; and the capacity of local search is weak, also may not get the optimal solution. For the crawling strategy of the current common focused crawler, the content of the web pages is generally provided by the editors, which results in some information irrelevant to the predetermined theme involved in the web pages. The whole web page documents will be often used in the genetic process, when the genetic algorithm is used in the focused crawler in the past, which results in the theme drift easily comes out in the process. For the problems mentioned above, this paper improves the genetic algorithm. Our proposed crawler by using genetic algorithm is named as Gcrawler.

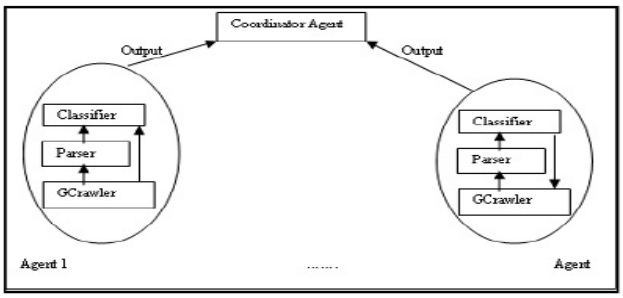

Fig. 4. Gcrawler

III. COMPARISON OF VARIOUS APPROACHES

To better understand how the approaches differ from each other, we have listed down the comparison between the approaches. Each point reviewed is than compared to other approach. The comparison also states the major points of our review. This basically makes the review more meaningful and easier for the reader to understand.

3.1 Comparison of focused crawler approaches

TABLE I
COMPARISON OF FOCUSSED CRAWLER APPROACHES

Without using background information	**Using background information**
System uses the Fish search algorithm to find the relevance of the search	The system consists of three components: Crawler, Classifier and Distiller
No training set provided while performing the search	Initially a training set is provided do decide whether the searched page is relevant or not relevant
The algorithm marks pages either relevant or non relevant based on simple keyword match	The classifier is trained on document samples available in topic taxonomy such as Yahoo and from them it learns to label new documents as relevant or not relevant
The algorithm uses pageRank mechanism to find the expansion of the links to other pages.	The classifier helps in determining future link expansion

3.2 Comparison between approaches of deep web crawler

The experiment shows that the deep crawler URL analysis model based on improved genetic algorithm[2] proposed can improve accuracy rate, recall rate effectively, and avoid getting into the local optimal solution. Table 1 shows the comparison among crawlers

TABLE III
EXPERIMENTAL RESULTS

	Single thread	**Multi thread**	**Gcrawler**
Precision	88.0%	80.0%	90.0%
Recall	22.83%	37.31%	37.31%

The table shows the comparison between the approaches used for deep web crawlers.

TABLE IIIII
COMPARISON OF APPROACHES OF DEEP CRAWLER

	Single threaded	**Multi threaded**	**Gcrawler**
Performance	Low	Medium	High
Pros	1.It maintains frontier 2. gives required information 3. Also added unvisited URLs to the frontier.	1.It maintains a fast lookup of URLs that have been crawled. 2.Optimizations can be performed on the multi-threaded model.	1.It can effectively find the global optimal solution jumping local optimization.
Cons	1.A sequential crawling loop spends a large amount of time in which either the CPU is idle or the network interface is idle.	1.The infrastructure supports at one extreme a very simple breadth-first crawler and at the other end crawler algorithms that may involve very complex URL selection mechanisms.	1.The capacity of local search is weak, 2. It may not get the optimal solution.

IV. CONCLUSIONS

General Crawler has some limitation in terms of precision and efficiency because of its generality, no specialty. Focused Crawler[1] improves the precision and recall of expert search on web. Focused crawler does not collect all pages but select and retrieve relevant page only. There are so many approaches to calculate the relevancy of page. Some base on structured, some used classifier to know the relevancy of page etc. Context based focused crawling give more accurate result to user according to their interest.

Focused crawler is the key technology of vertical search engine, and the relevance analysis of URL topic is the problem faced by focused crawler which must be solved firstly. The two approaches for focused crawling is also compared as shown in the Table 1. Along with the approach the future work done to improve the earlier methods has also being reviewed.

Various approaches for deep web crawling are also discussed. From the discussion it is evident that Gcrawler is most precise followed by single threaded and then multi threaded crawler. It has 90% precision. Also recall factor is better than single threaded approach. Table 3 shows the comparison between these approaches. Thus, from the comparison we see that Gcrawler is better than the other two.

REFERENCES

[1] Meenu et al, A review of focused crawler approaches in International Journal of Advanced Research in Computer Science and Software Engineering, in Volume 4, Issue 7, July 2014.

[2] K.F.Bharti et al, A Framework for deep web crawler using genetic algorithm in International journal of electronics and computer science engineering, ISSN- 2277-1956

[3] Nidhi Jain and Paramjeet Rawal, "A study of focused web crawlers for semantic web" in / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013, 398-402

[4] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613–620

[5] H. Liu, J. Janssen, E. Milios, Using HMM to learn user browsing patterns for focused web crawling, Data & Knowledge Engineering (2006)270–32

[6] Ayoub Mohamed H. Elyasir1, Kalaiarasi Sonai Muthu Anbanan, "Focused Web Crawler", International Conference on Information and Knowledge management,2012.

[7] Hongyu liu et al, "probabilistic models for focused web crawling",Dept. of Mathematics and Statistics, Dalhousie University Halifax, NS, Canada B3H 1W5 .

[8] M. Diligenti† et al, "focused crawler using context graphs", NEC Research Institute, 4 IndependenceWay, Princeton, NJ 08540-6634 USA

[9] Dhiraj Khurana et al, "web crawler:A review", in IJCSMS International Journal of Computer Science & Management Studies, Vol. 12, Issue 01, January 2012

[10] Jaideep Srivatsava, B. Mobasher, R. Cooley, "Web Mining: Information and Pattern Discovery on the World Web", International conference on tools with Artificial Intelligence, pp558-567, Newport beach, 1997.

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

[12] Soumen Chakrabarti, ByronE.Dom, S. Ravikumar, Prabhakar, Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, Jon Kleinberg " Mining Web's Link Structure", IEEE Computer, pp60-67, 1999

[13] J.Akilandeswari, N.P. Gopalan, "A Web Mining System using Reinforcement Learning for Scalable Web Search with Distributed,Fault-tolerant Multiagents", , Issue 11,Vol 4, p1633-1639, November 2005.

.