Annals of Computer Science and Intelligence Systems

Deep Web Crawling Techniques: An Efficiency Review Focused on Accuracy

C. Raghav Singh*[1] & D. Priya Verma[2]
Student[1], Professor[2]
Computer Science and Engineering, Bapurao Deshmukh College of Engineering, India

ABSTRACT

As deep web grows at a very fast pace, there has been increased interest in techniques that help efficiently locate deep-web interfaces. However, due to the large volume of web resources and the dynamic nature of deep web, achieving wide coverage and high efficiency is a challenging issue. We propose a three-stage framework, for efficient harvesting deep web interfaces. Project experimental results on a set of representative domains show the agility and accuracy of our proposed crawler framework, which efficiently retrieves deep-web interfaces from large-scale sites and achieves higher harvest rates than other crawlers using Naïve Bayes algorithm. In this paper we have made a survey on how web crawler works and what are the methodologies available in existing system from different researchers.

KEYWORDS: Deep web, web mining, feature selection, ranking

I. INTRODUCTION

The deep (or hidden) web refers to the contents lie behind searchable web interfaces that cannot be indexed by searching engines. Based on extrapolations from a study done at University of California, Berkeley, it is estimated that the deep web contains approximately 91,850 terabytes and the surface web is only about 167 terabytes in 2003. More recent studies estimated that 1.9 petabytes were reached and 0.3 petabytes were consumed worldwide in 2007. An IDC report estimates that the total of all digital data created, replicated, and consumed will reach 6 petabytes in 2014. A significant portion of this huge amount of data is estimated to be stored as structured or relational data in web databases deep web makes up about 96% of all the content on the Internet, which is 500-550 times larger than the surface web. These data contain a vast amount of valuable information and entities such as Infomine, Clusty, Books In Print may be interested in building an index of the deep web sources in a given domain (such as book). Because these entities cannot access the proprietary web indices of search engines (e.g., Google and Baidu), there is a need for an efficient crawler that is able to accurately and quickly explore the deep web databases.

It is challenging to locate the deep web databases, because they are not registered with any search engines, are usually sparsely distributed, and keep constantly changing. To address this problem, previous work has proposed two types of crawlers, generic crawlers and focused crawlers. Generic crawlers, fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) and Adaptive Crawler for Hidden-web Entries (ACHE) can automatically search online databases on a specific topic. FFC is designed with link, page, and form classifiers for focused crawling of web forms, and is extended by ACHE with additional components for form filtering and adaptive link learner.

The link classifiers in these crawlers play a pivotal role in achieving higher crawling efficiency than the best-first crawler. However, these link classifiers are used to predict the distance to the page containing searchable forms, which is difficult to estimate, especially for the delayed benefit links (links eventually lead to pages with forms). As a result, the crawler can be inefficiently led to pages without targeted forms. Besides efficiency, quality and coverage on relevant deep web sources are also challenging.

When selecting a relevant subset from the available content sources, FFC and ACHE prioritize links that bring immediate return (links directly point to pages containing searchable forms) and delayed benefit links. But the set of retrieved forms is very heterogeneous. For example, from a set of representative domains, on average only 16% of forms retrieved by FFC are relevant. Furthermore, little work has been done on the source selection problem when crawling more content sources. Thus it is crucial to develop smart crawling strategies that are able to quickly discover relevant content sources from the deep web as much as possible.

The propose work, achieve both wide coverage and high efficiency for a focused crawler. Our main contributions are:We propose a novel three-stage framework to address the problem of searching for hidden-web resources. Our site locating technique employs a reverse searching technique (e.g., using Google's"link:" facility to get pages pointing to a given link) and incremental three-level site prioritizing technique for unearthing relevant sites, achieving more data sources. During the in-site exploring stage, design a link tree for balanced link prioritizing, eliminating bias toward web pages in popular directories.

In the propose work an adaptive learning algorithm that performs online feature selection and uses these features to automatically construct link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on a topic using the contents of the root page of sites, achieving more accurate results. During the in-site exploring stage, relevant links are prioritized for fast in-site searching. Project experimental results on a set of representative domains show the agility and accuracy of our proposed crawler framework, which efficiently retrieves deep-web interfaces from large-scale sites and achieves higher harvest rates than other crawlers using Naïve Bayes algorithm.

II. LITERATURE SURVEY

There is a rich literature, here we discuss the most related work.

Feng Zhao et al. [1] presents a two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces. Deep web grows at a very fast pace, there has been increased interest in techniques that help efficiently locate deep-web interfaces. However, due to the large volume of web resources and the dynamic nature of deep web, achieving wide coverage and high efficiency is a challenging issue. Here propose a two-stage framework, namely Smart Crawler, for efficient harvesting deep web interfaces. In the first stage, Smart Crawler performs site-based searching for center pages with the help of search engines, avoiding visiting a large number of pages. In the second stage, Smart Crawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking. To achieve more accurate results for a focused crawl, Smart Crawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, Smart Crawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking.

Jianxiao Liu et al.[2] proposed an Approach of Semantic Web Service Classification Based on Naive Bayes ,proposed a method to classify and organize the semantic Web services to help users find the services to meet their needs quickly and accurately is a key issue to be solved in the era of service-oriented software engineering. This paper makes full use the characteristics of solid mathematical foundation and stable classification efficiency of Naive Bayes classification method. It proposes a semantic Web service classification method based on the theory of naive bayes. It elaborates the concrete process of how to use the three stages of bayesian classification to classify the semantic Web services in the consideration of service interface and execution capacity.

Bo Tang, presents an approach toward Optimal Feature Selection In Naive Bayes For Text Categorization [3].
Author proposed that, automated feature selection is important for text categorization to reduce the feature size and to speed up the learning process of classifiers. In this paper, author present a novel and efficient feature selection framework based on the Information Theory, which aims to rank the features with their discriminative capacity for classification. Author first revisit two information measures: Kullback-Leibler divergence and Jeffrey's divergence for binary hypothesis testing, and analyze their asymptotic properties relating to type I and type II errors of a Bayesian classifier.
Here author introduced new feature selection approaches based on the information measures for naive Bayes classifiers, aiming to select the features that offer the maximum discriminative capacity for text classification. They also derived the asymptotic distributions of these measures, which leads to the other version of the Chi-square statistic approach for feature
Selection.

Amruta Pandit and Prof. Manisha Naoghare[4], proposed work for Efficiently Harvesting Deep Web Interface with Reranking and Clustering.The rapid growth of the deep web poses predefine scaling challenges for general purpose crawler and search engines. There are increasing numbers of data sources now become available on the web, but often their contents are only accessible through query interface. Here author proposed a framework to deal with this problem, for harvesting deep web interface. Here Parsing process takes place. To achieve more accurate result crawler calculate page rank and Binary vector of pages which is extracted from the crawler to achieve more accurate result for a focused crawler give most relevant links with an ranking. This experimental result on a set of representative domain show the agility and accuracy of this proposed crawler framework which efficiently retrieves web interface from large scale sites.

Anand Kumar et al. [5] presents a two-phase system, to be specific Smart Crawler, for productive gathering profound web interfaces.
In this paper, author proposed, web develops at a quick pace, there has been expanded enthusiasm for procedures that assistance effectively find profound web interfaces. Be that as it may, because of the expansive volume of web assets and the dynamic way of profound web, accomplishing wide scope and high proficiency is a testing issue. In the primary stage, Smart Crawler performs site-based hunting down focus pages with the assistance of web crawlers, abstaining from going to a substantial number of pages.

Akshaya Kubba[7] mentioned that, Web mining is an important concept of data mining that works on both structured and unstructured data. Search engine initiates a search by starting a crawler to search the World Wide Web (WWW) for documents .Web crawler works in a ordered way to mine the data from the huge repository. The data on which the crawlers were working was written in HTML tags, that data lags the meaning. It was a technique of text mapping. Semantic web is not a normal text written in HTML tags that are mapped to the search result, these are written in Resource description language. The Meta tags associated with the text are extracted and the meaning of content is find for the updated information and give us the efficient result in no time.

Monika Bhide et al. focus on accessing relevant web data and represents significant algorithm i.e. adaptive learning algorithm, reverse searching and classifier[8].the web stores huge amount of data on different topics.The main goal is to locating deep web interfaces. To locating deep web interfaces uses techniques and methods.The locating deep web interfaces system works in two stages. In the first stage apply reverse search engine algorithm and classifies the sites and the second stage ranking mechanism use to rank the relevant sites and display different ranking pages. The work is based on crawl ordering that reveals the incremental crawler performance, better and is more powerful because it allows revisitation of pages at different rates and efficiently accessing web data. It can eliminate bias toward certain web site directories for wider coverage. This will provide the most relevant sites and accurate results to the users.

Raju Balakrishnan et al.[10] proposed, selecting the most relevant web databases for answering a given query. The existing database selection methods (both text and relational) assess the source quality based on the query-similarity-based relevance assessment. When applied to the deep web these methods have two deficiencies. First is that the methods are agnostic to the correctness (trustworthiness) of the sources. Secondly, the query based relevance does not consider the importance of the results. These two considerations are essential for the open collections like the deep web. Since a number of sources provide answers to any query, author conjuncture that the agreements between these answers are likely to be helpful in assessing the importance and the trustworthiness of the sources. While computing the agreement, we measure

and try to adjust for any collusion between the sources. This adjusted agreement is modeled as a graph with sources at the vertices. On this agreement graph, quality score of a source that we call SourceRank, is calculated as the stationary visit probability of a random walk. We evaluate SourceRank in multiple domains, including sources derived from Google Base, with sizes up to 675 sources. They demonstrate that the SourceRank tracks source corruption. Our relevance evaluations show that SourceRank improves precision by 22-60% over the the Google Base and the existing methods.

D. Shestakov, address the accurate estimation of deep web by sampling one national web domain[11].here author report some of their results obtained when surveying Russian web.The Host-IP clustering sampling technique addresses the drawback of previous deep web surveys and allow to characterize a national segment of deep web.He aiso got an insights on the sight of Russian deep Web by calculating upper bound estimate for the total number of entity in online database.

Suryakant Chouthary et al. [12] worked for model-based rich internet applications crawling in which they they design methods based on menu and probability models. Strategies for crawling Web sites efficiently have been described more than a decade ago. Since then, Web applications have come a long way both in terms of adoption to provide information and services and in terms of technologies to develop them. With the emergence of richer and more advanced technologies such as AJAX, ―Rich Internet Applications (RIAs) have become more interactive, more responsive and generally more user friendly. Unfortunately, we have also lost our ability to crawl them. Building models of applications automatically is important not only for indexing content, but also to do automated testing, automated security assessments, automated accessibility assessment and in general to use software engineering tools. We must regain our ability to efficiently construct models for these RIAs. In this, authors present two methods, based on Model-Based Crawling (MBC) first introduced: the menu model and the probability model. These two methods are shown to be more effective at extracting models than previously published methods, and are much simpler to implement than previous models for MBC. A distributed implementation of the probability model is also discussed. We compare these methods and others against a set of experimental and real RIAs, showing that in our experiments, these methods find the set of client states faster than other approaches, and often finish the crawl faster as well.

Eduard C. Dragut et al. [13] describe a Web query interface extraction algorithm, which combines HTML tokens and the geometric layout of these tokens within a Web page. Tokens are classified into several classes out of which the most significant ones are text tokens and field tokens. A tree structure is derived for text tokens using their geometric layout. Another tree structure is derived for the field tokens. The hierarchical representation of a query interface is obtained by iteratively merging these two trees. Thus, convert the extraction problem into an integration problem. Their experiments show the promise of our algorithm: it outperforms the previous approaches on extracting query interfaces on about 6.5% in accuracy as evaluated over three corpora with more than 500 Deep Web interfaces from 15 different domains. Author presented a technique for extracting hierarchical schema trees from Deep Web interfaces. This representation is richer and thus easier to be used for Deep Web integration than previous, at models. Our extraction technique is based on a small set of general design rules which, together with a proper exploitation of visual layout of HTML pages, allow to extract schema trees with high accuracy. We showed experimentally that our method outperforms previous approaches even if its capabilities for extracting structure are disregarded.

Luciano Barbosa, Juliana Freire [14] describe new adaptive crawling strategies to efficiently locate the entry points to hidden-Web sources. The fact that hidden-Web sources are very sparsely distributed makes the problem of locating them especially challenging. Author deal with this problem by using the contents of pages to focus the crawl on a topic; by prioritizing promising links within the topic; and by also following links that may not lead to immediate benefit. Author propose a new framework whereby crawlers automatically learn patterns of promising links and adapt their focus as the crawl progresses, thus greatly reducing the amount of required manual setup and tuning. This strategy effectively balances the exploitation of acquired knowledge with the exploration of links with previously unknown patterns, making it robust and able to correct biases introduced in the learning process. We have shown, through a detailed experimental evaluation that substantial increases in harvest rates are obtained as crawlers learn from new experiences. Since crawlers that learn from scratch are able to obtain harvest rates that are comparable to, and sometimes higher than manually configured crawlers, this framework can greatly reduce the effort to configure a crawler.

Yeye He et al.[15] describe a prototype system, that specializes in crawling entity-oriented deep-web sites. They propose techniques tailored to tackle important sub problems including query generation, empty page filtering and URL deduplication in the specific context of entity oriented deep-web sites. These techniques are experimentally evaluated and shown to be effective.

Deep-web crawl is concerned with the problem of surfacing hidden content behind search interfaces on the Web. While many deep-web sites maintain document-oriented textual content (e.g., Wikipedia, PubMed, Twitter, etc.), which has traditionally been the focus of the deep-web literature, we observe that a significant portion of deep-web sites, including almost all online shopping sites, curate structured entities as opposed to text documents. Although crawling such entity-oriented content is clearly useful for a variety of purposes, existing crawling techniques optimized for document oriented content are not best suited for entity-oriented sites.
As the size of web is growing, in order to complete the downloading of pages in fewer amounts of time and increase the coverage of crawlers it is necessary to distribute the crawling process.

Sawroop Kaur Bal G. Geetha[16] present client server architecture based smart distributed crawler for crawling web. In this architecture load between the crawlers is managed by server and each time a crawler is loaded, load is distributed to others by dynamically distributing the URLs. Focused crawlers makes efficient usage of network bandwidth and storage capacity, when distributed can enhance the performance As search engines fail to fulfill the user's requirement for complete and recently updated information, it turns out to be profoundly attractive to utilize distributed web crawlers.

III. PROPOSED APPROACH

To efficiently and effectively discover deep web data sources, proposed Crawler is designed with a three-stage architecture. The first site locating stage finds the most relevant site for a given topic, the second in-site exploring stage uncovers searchable forms from the site and then the third stage apply naïve base classification ranked the result.

First Phase: Fetching Results from Google:
In first phase the proposed system fetches results from Google search engine with the help of Google developer API and JSON (Java Script Object Notation).

Second Phase: Fetching the Word count from HTML Pages:
In second phase the proposed system opens the web pages internally in application with the help of Jsoup API and preprocess it. Then it performs the word count of query in web pages.

Third Phase: Frequency Analysis:
In third phase the proposed system performs frequency analysis based on TF and IDF. It also uses a combination of TF*IDF for ranking web pages

The above fig 1 show project flow diagram in which first user search query. The query get search in Google. The URL/web will display through JSON API. It is commonly used for transmitting data in web applications e.g., sending some data from server to the client so it can display on web pages and vice versa. In our proposed work the only top five URL will display. After the Result is fetch, the next step is Non-HTML data extraction. Jsoup is use as parser for Non-HTML .data extraction. It take only text data and avoid opening and closing brackets, images, and all noisy data.

In next step the word count is perform on fetch result. after the word count is perform the frequency get calculated and rerank result will get. At the end the comparative analysis is done.

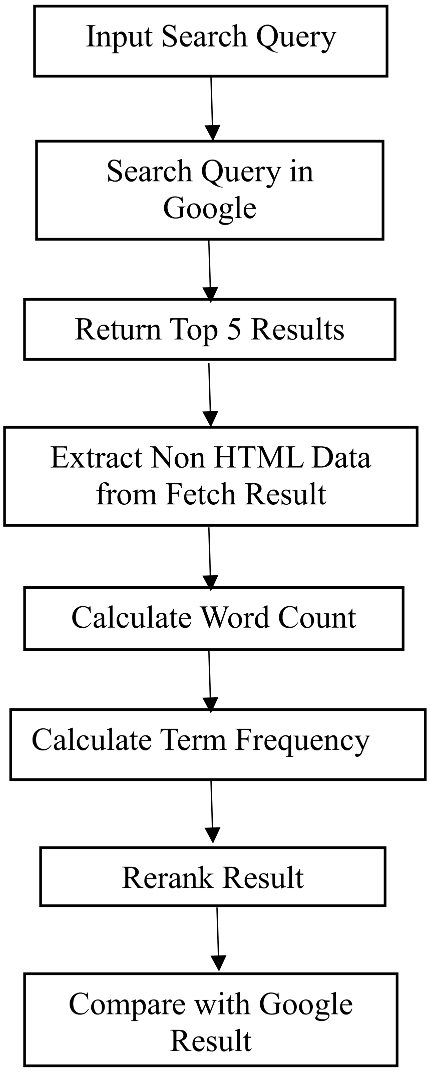

Fig 1 : Flow of Proposed Approach

IV. OBJECTIVES OF THE PRESENT WORK

The objectives of the proposed approach are best described as below:

1. To access data from Google for Top Results, perform word count on top results with pre-processing as well.
2. To Perform HTML Extraction.
3. To Perform Frequency Calculation.
4. To Perform Comparative Analysis.

V. CONCLUSION

A brief survey of different crawling methods available in the literature including analysis and comparative study of different techniques used for crawling is done. To improve the efficiency and to overcome the problem of existing system, KNN perform on top results with preprocessing as well as Frequency Calculation performs using Naïve Bayes algorithm is discussed.

VI. REFERENCES

[1] Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang, Hai Jin "Smart Crawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces" in IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 9, NO. 4, JULY/AUGUST 2016.

[2] Jianxiao Liu, Zonglin Tian, Panbiao Liu, Jiawei Jiang, "An Approach of Semantic Web Service Classification Based on Naive Bayes" in 2016 IEEE International Conference on Services Computing, SEPTEMBER 2016.

[3] Bo Tang, Student Member, IEEE, Steven Kay, Fellow, IEEE, and Haibo He, Senior Member, IEEE "Toward Optimal Feature Selection in Naive Bayes for Text Categorization" in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 9 Feb 2016.

[4] Amruta Pandit ,Prof. Manisha Naoghare, "Efficiently Harvesting Deep Web Interface with Reranking and Clustering", in International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 1, January 2016.

[5] Anand Kumar , Rahul Kumar, Sachin Nigle, Minal Shahakar, "Review on Extracting the Web Data through Deep Web Interfaces, Mechanism", in International Journal of Innovative Research in Computer and Communication Engineering, Vol. 4, Issue 1, January 2016

[6] Sayali D. Jadhav, H. P. Channe "Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques" in International Journal of Science and Research, Volume 5 Issue 1, January 2016.

[7] Akshaya Kubba, "Web Crawlers for Semantic Web" in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 5, May 2015.

[8] Monika Bhide, M. A. Shaikh, Amruta Patil, Sunita Kerure,"Extracting the Web Data Through Deep Web Interfaces" in INCIEST-2015.

[9] Y. He, D. Xin, V. Ganti, S. Rajaraman, and N. Shah, "Crawling deep web entity pages," in Proc. 6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 355–364.

[10] Raju Balakrishnan, Subbarao Kambhampati, "SourceRank: Relevance and Trust Assessment for Deep Web Sources Based on Inter-Source Agreement" in WWW 2011, March 28–April 1, 2011.

[11] D. Shestakov, "Databases on the web: National web domain survey," in Proc. 15th Symp. Int. Database Eng. Appl., 2011, pp. 179–184. [12] D. Shestakov and T. Salakoski, "Host-ip clustering technique for deep web characterization," in Proc. 12th Int. Asia-Pacific Web Conf., 2010, pp. 378–380.

[12] Suryakant Chouthary, Emre Dincturk, Seyed Mirtaheri, Ggregor V. Bochmann, Guy-Vincent Jourdan and Iosif Viorel onut"model-based rich internet applications crawling: ―menu‖ and ―probability‖ models "in journal of Web Engineering, Vol.0, No.2003.

[13] Eduard C. Dragut, Thomas Kabisch, Clement Yu, and Ulf Leser. A hierarchical approach to model web query interfaces for web source integration. Proc. VLDB Endow. 2(1):325–336, August 2009.

[14] Luciano Barbosa, Juliana Freire "An Adaptive Crawler for Locating Hidden Web Entry Points" in WWW 2007

[15] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In Proceedings of the sixth ACM international conference on Web search and data mining, pages 355–364. ACM, 2013.

[16] Sawroop Kaur Bal G.Geetha"Smart distributed web crawler"in International Conference On Information Communication And Embedded System(ICICES 2016)