IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN : 2278-0661, p-ISSN : 2278-8727
PP 91-94
www.iosrjournals.org

Extracting the Web Data through Deep Web Interfaces

Monika Bhide[1], M. A. Shaikh[2], Amruta Patil[3], Sunita Kerure[4]
[1,2,3,4]Department of Information Technology, JSPM'S Rajarshi Shahu College of Engineering ,Savitribai Phule Pune University, Pune, India

Abstract: *The web stores huge amount of data on different topics. The users accessing web data vastly in now days. The main goal of this paper is to locating deep web interfaces. To locating deep web interfaces uses techniques and methods. This paper is focus on accessing relevant web data and represents significant algorithm i.e. adaptive learning algorithm, reverse searching and classifier. 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.*

Keywords *- adaptive learning, Deep web, feature selection, ranking, two- stage crawler*

I. INTRODUCTION

The Deep Web consists of data that is present on the web but is not accessible by text search engine through traditional crawling and indexing. Hidden web data, stored in structured or unstructured databases [4], is inherently hidden behind search forms. It is qualitatively and quantitatively different from the surface Web. The quality content of the deep Web is 1,000 to 2,000 times greater than that of the surface Web whereas overall the hidden web contains approximately 7,500 terabytes of data and 550 billion individual documents in contrast to the surface Web, which is reported to about 167 terabytes [4]. Crawler is program that visits web sites and reads their pages and other information in order to make entries for search engine address list. It maintains a list of URLs of the document that showing and fetches indexes in inside URL queue.

A web crawler is a system, a program that traverses the web for the purpose of bulk downloading of web pages in an automated manner. The crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. A related use is web archiving (a service provided by e.g., the Internet archive [3]) where large sets of web pages are periodically collected and archived for posterity.

II. RELATED WORK

Existing strategies were dealing with making of a single profile per user, but conflict occurs when user's interest varies for the same query example when a user is interested in banking exams in query "bank" may be slightly interested in accounts of money bank where not at all interested in blood bank. At such time conflict occurs so we are dealing with negative preferences to obtain the fine grain between the interested results and not interested. Consider following two aspects:

2.1 Document-Based method

These methods aim at capturing users' clicking and browsing behavior. It deals with click through data from the user i.e. the documents user has clicked on. Click through data in search engines can be thought of as triplets (q, r, c)

Where,

q = query
r = ranking
c = set of links clicked by user.

2.2 Concept-based methods

These methods aim at capturing users' conceptual needs. Users' browsed documents and search histories. User profiles are used to represent users' interests and to infer their intentions for new queries.

III. PROPOSED WORK

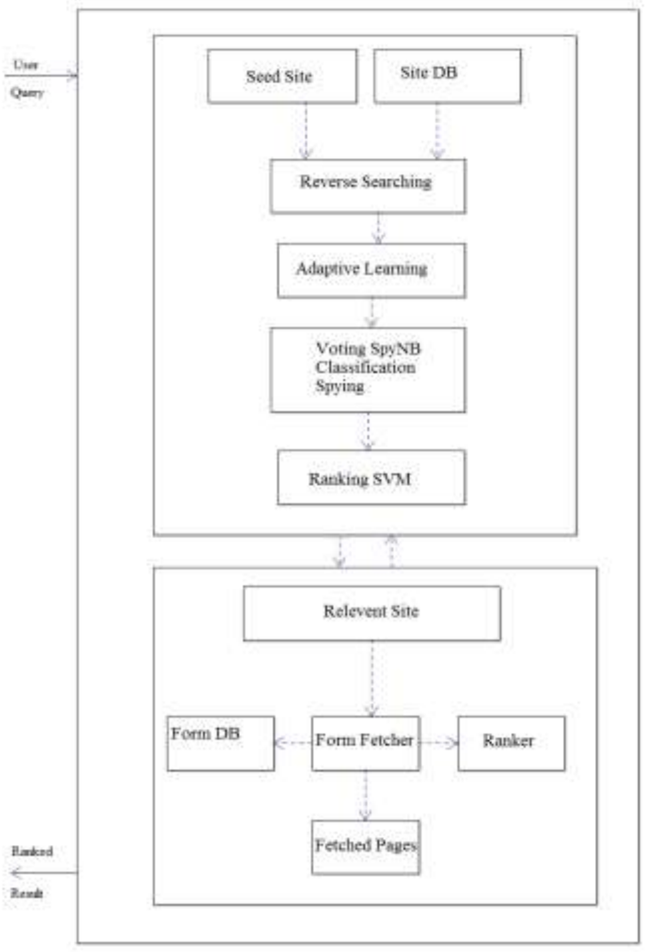

Figure 1: System Architecture

Figure.1 describes the two stage architecture , in the first stage starts with a seed set of sites in a site database. Seeds sites are sites given to start crawling, which begins by next URLs from select seed sites to explore other pages and other domains. When the number of unvisited URLs in the database is less than a threshold during the crawling process, then system perform ”reverse searching” of well-known deep web sites for center pages and feeds these pages back to the site database. In second stage fetches homepage URLs from the site database, we going to rank the relevant information.

1.1 Site Locating

Site locating use for searching relevant sites for particular topic.

1.2 Site Classifier

When a new site comes, the homepage content of the site is extracted and parsed by removing stop words and stemming. Then we construct a feature vector for the site and the resulting vector is fed into a Naive Bayes classifier to determine if the page is topic-relevant or not.

1.3 Adaptive learning

This algorithm performs online feature selection and uses these features to automatically construct link rankers. The adaptive learning algorithm chooses selective features of links and uses these features to automatically create a link classifier. In this relevant sites are prioritized for fast searching and the crawling is focused on a topic using the contents of the root page of sites, achieving more accurate results.

IV. Mathematical Model

Let us consider S as a system for Concept Based User Profile.
S= {……
Input: Identify the inputs
F= {f_1, f_2, f_3 ….., f_n| ‘F’ as set of functions to execute commands.}
I= {i_1, i_2, i_3…|’I’ sets of inputs to the function set}
O= {o_1, o_2, o_3….|’O’ Set of outputs from the function sets}
S= {I, F, O}
I = {Query submitted by the user, ...}
O = {Output of desired query,...}
F = {Functions implemented to get the output, SPY-NB algorithm, Clustering algorithm}

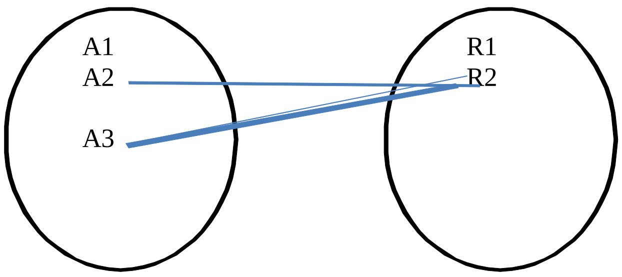

Figure 2: Set Theory

Figure 2 describes A1: Query provided by the user. Example: Apple iphone
A2: Query provided by user. Example: Orange fruit
R1: Resulted web snippets provided by the search engine
A3: Wrong or incorrect query submitted
R2: Error routine in the search engine

4.1 Algorithms & Techniques Used

4.1.1 Reverse searching

This method describes the result page from the search engine is first parsed to extract links. Then these pages are downloaded and analyzed to decide whether the links are relevant or not using the rules. The first rule is if the page contains related searchable forms, it is relevant and other rule is if the number of seed sites or fetched deep web sites in the page is larger than a user defined Threshold, the page is relevant.

```
Input: seed sites and harvested deep websites
1 while # of candidate sites less than a threshold do
2 // pick a deep website
3 site = getDeepWebSite(siteDatabase,
seedSites)
4 resultPage = reverseSearch(site)
5 links = extractLinks(resultPage)
6 foreach link in links do
7 page = downloadPage(link)
8 relevant = classify(page)
9 if relevant then
10 relevantSites =
extractUnvisitedSite(page)
11  relevantSites
12 end
     13   nd
     14   nd
```

4.1.2. Incremental Site Prioritizing

This method follows the out-of site links of relevant sites. To accurately classify out-of-site links, Site Frontier utilizes two queues to save unvisited sites. The high priority queue is for out-of-site links that are classified as relevant by Site Classifier and are judged by Form Classifier to contain searchable forms. The low priority queue is for out of site links that only judged as relevant by Site Classifier. For each level, Site Ranker assigns relevant scores for prioritizing sites. The low priority queue is used to provide more candidate sites. Once the high priority queue is empty, sites in the low priority queue are pushed into it progressively.

```
Input: siteFrontier
1HQueue=SiteFrontier.CreateQueue(HighPriority)
2 LQueue=SiteFrontier.CreateQueue(LowPriority)
3 while siteFrontier is not empty do
4   if HQueue is empty then
5     HQueue.addAll(LQueue)
6     Lqueue.Clear()
7   end
8   site = HQueue.poll()
9   relevant = classifySite(site)
10  if relevant then
11    performInSiteExploring(site)
12    Output forms and OutOfSiteLinks
13    siteRanker.rank(OutOfSiteLinks)
14    if forms is not empty then
15      HQueue.add (OutOfSiteLinks)
16    end
17    else
18      LQueue.add(OutOfSiteLinks)
19    end
20  end
21 end
```

V. Conclusion

This paper proposes the extracting web data through deep web interfaces. This paper is based on crawl ordering that reveals the incremental crawler performance, better and is more powerful because it allows re-visitation 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.

REFERENCES

[1] Peter Lyman and Hal R. Varian. How much information? 2003. Technical report, UC Berkeley, 2003.
[2] Roger E. Bohn and James E. Short. How much information? 2009 report on american consumers. Technical report, University of California, San Diego, 2009.
[3] Martin Hilbert. How much information is there in the"information society"? Significance, 2012.
[4] Idc worldwide predictions 2014: Battles for dominance – and survival on the 3rd platform. http://www.idc.com/research/Predictions14/index.jsp, 2014.
[5] Michael K. Bergman. White paper: The deep web: Surfacing hidden value. Journal of electronic publishing, 2001.
[6] 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, 2013.
[7] Searching for HiddenWeb Databases Luciano Barbosa University of Utah and Juliana Freire University of Utah
[8] Luciano Barbosa and Juliana Freire. An adaptive crawler for locating hidden-web entry points. In Proceedings of the 16th international conference on World Wide Web, pages 441–450. ACM, 2007.
[9] Luciano Barbosa and Juliana Freire. Searching for hidden-web databases. In WebDB, pages 1–6, 2005.
[10] Michael K. Bergman. White paper: The deep web: Surfacing hidden value. Journal of electronic publishing, 7(1), 2001.