International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 07 | July-2017 www.irjet.net p-ISSN: 2395-0072

Two Stage Smart Crawler with NBSVM classifier

Ashlesha Shirsath

[1] *Ashlesha Shirsath Student, Computer Engineering, P.E.S Modern College of Engineering, Maharashtra, India*

Abstract - *Internet is a huge repository of web pages housing tons of information on various sections of human development and implementation. The dark web refers to that part of Internet where the conventional search engines cannot reach as it cannot be indexed. the huge size of this Internet is itself a hindrance in retrieving efficient and relevant information this is the reason there is a need of a good search engine to bring information as relevant as possible to the user. One of the important and crucial process of searching relevant content is web crawling. A web crawler is a buffer that digs the Internet to gather and create a temporary database to further analyze and arrange data. This project is all about designing an efficient Web Crawler that not only crawls the World Wide Web but also focuses on the topic relevant content. The two stage architecture involves site based searching for home pages, prioritize relevant ones for a topic, second stage involves In-site exploring and adaptive link ranking. The efficiency of a crawler depends on the classification of web pages at the first place before ranking them. Naive Bayes Classifier is used in this paper. Efforts are made to improve this classification process by combining the results of NB and SVM classifier. Research has proved that his combination, popularly known as the NBSVM classifier does yield better results.*

Key Words: *Dark/Deep web, two-stage crawler, feature selection, adaptive learning, etc…*

1. INTRODUCTION

Hidden/deep web cannot be indexed by conventional search engines, this hidden web comprises of content which is 500- 550 times larger than the surface web [3], [4], which houses huge amount of valuable information. Keeping this in mind there is a need of an accurate and quick crawler to dig out the hidden web and extract relevant information. It is a challenge to find relevant content in deep web, this challenge is overcome by the design of a Smart Crawler. Along with the efficiency it is a great challenge for quality and coverage on relevant deep web sources. Thus, a good crawler should return large amount of high-quality results from the most relevant source. As it is found that deep website have less searchable forms, we have designed the crawler in two stages. Site locating stage helps to get wide coverage of sites for a focused crawler, and the in-site exploring stage efficiently searches for web forms within a site. Site locating functionality involves a reverse searching process and incremental two-level site prioritizing functionality for unearthing relevant sites, achieving more data sources. Adaptive learning is used for online feature selection which are in turn used for constructing link rankers. In the site locating stage, closely relevant sites are prioritized and the crawling focuses on a topic using the contents of the home page of sites, returning highly accurate results. During the in site exploring stage, relevant links are prioritized for fast insite searching. This two-stage crawler is a domain specific crawler which classifies the sites in first stage to remove irrelevant websites but also categorizes the searchable forms. The first stage finds a most relevant site for a given topic and the discovers searchable from this site. The whole crawling process starts with a set of candidate sites called seed sites. After a certain threshold, the Smart Crawler performs 'Reverse Searching' process wherein the pages are sent back to the URL database. These are in turn prioritized by the Site Ranker which makes use of the Adaptive Learning technique i.e. to adaptively learn from the features of deep websites. For a given topic, the Site Classifier categorizes URLs as relevant or irrelevant based on the homepage content. Once the most relevant site is obtained in the site locating stage, the second stage does efficient in-site exploration for filtering searchable forms. Links of a site are stored in Link Frontier and respective pages are returned and forms are classed by Form Classifier to locate searchable forms. The classifier used for the above purpose is Naive Bayes classifier. Through various studies and research, results have been proved that when results of Naive Bayes are combined with SVM classifier, better results are obtained. To prioritize links, SmartCrawler ranks them with Link Ranker. Note that site locating stage and in-site exploring stage are mutually dependent on each other. When the crawler discovers a new site, the sites URL is inserted into the Site Database. The Link Ranker's performance is enhanced by an Adaptive Link Learner, which studies the URL path leading to relevant forms.

1.1 REVIEW OF LITERATURE

Various papers and sources are studied to perform a thorough literature review. There are basically two types of crawlers: Focused Crawlers and Generic Crawlers, Generic crawlers are mainly developed for featuring deep web and directory construction of deep web resources, that do not limit search on a specific topic, but attempt to fetch all searchable forms [1], [2]. Focused crawler is developed to traverse links to pages of interest and avoid

links to off-topic regions [5],[6]. The classier learns to classify pages as topic-relevant and assigns priority to links in topic relevant pages. The FFC and ACHE are focused crawlers used for deep web interfaces. Focused crawlers attempt to fasten the crawling, increase the retrieval of high quality pages, assign credits to documents [7]. In optimal crawling strategies for a search engine we swathe web crawler as a 2D random walker [8] on the graph whose points are the web pages and line are the hyperlinks between these web pages, Proposed crawler is a two-part scheme improving the crawling process so that the age level of staleness is reduced and quality is increased. In general, several probabilistic models for user browsing in infinite web are proposed and studied to understand how deep and breadth a crawler must go to retrieve a significant portion of web pages that is actually relevant [8]. Experimental results prove that a crawler requires just a few levels of dept. to reach maximum number of web pages because they are widely spread at each level, which also suggests that large should also be included in the crawling process. Classification is a very crucial step before prioritizing the pages, paper in the references [10] has proved that Combined with SVM yields better results as Naive Bayes first classifies on a probability and SVM the further uses declassified data to classify using the vector. SVM works well with larger dataset and ours being a google dataset, it worked well and yielded better result.
here.

2. SYSTEM ARCHITECTURE / SYSTEM OVERVIEW

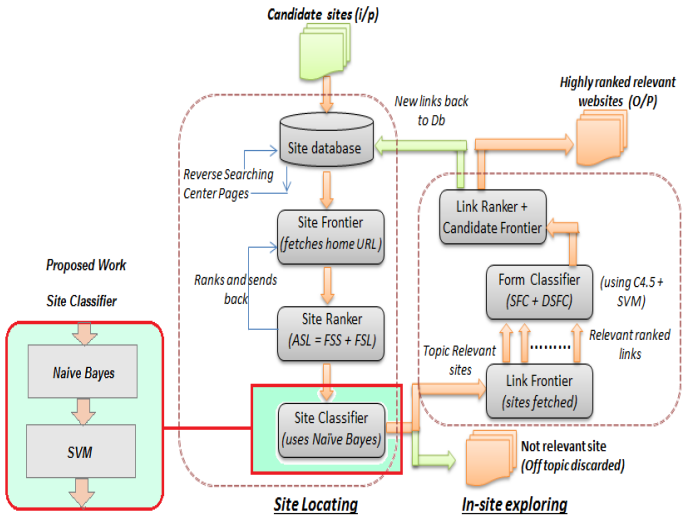

Chart -1: Architecture Diagram of Two Stage Smart Crawler

Processing Steps:
The following scheme consists of two stages:
1) Site Locating
I. Reverse searching
ii. Incremental site prioritizing
2) In-Site Exploring
I. Balanced link prioritizing
II. Adaptive Learning

1. Site Locating- Consists of Site collecting, Site ranking and Site classification. Reverse searching means randomly picking a known deep website or a seed site and use a search engines facility to find center pages and other relevant sites. The result page from the search engine is rest traversed to extract links. Then these pages are downloaded and analyzed to decide whether the links are relevant or not. If the page contains related searchable forms, it is relevant. 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. To obtain broad coverage on websites, an incremental site prioritizing strategy is used. Wherein we record learned patterns of deep web sites and make paths for incremental crawling. Then, unvisited sites are assigned to Site Frontier and are prioritized by Site Ranker, and visited sites are added to fetched site list. While crawling, SmartCrawler follows the out-offsite links of relevant sites. To perfectly classify out of sitelinks, Site Frontier utilizes two queues to save unvisited sites. The high priority queue is for out-of-site links that reclaimed as relevant by Site Classifier and are decided by Form Classifier to have searchable forms. The low priority queue is for out-offsite links that only decided as relevant by Site Classifier. As seen in the diagram, classifier has been replaced by a combination of NB and SVM classifier. After implementing this concept, results have proved that there is a improvement in the efficiency of the crawler.2. In-site exploring is done to find searchable forms. The motive is to quickly find searchable forms and to scan web directories of the site as much as possible. To obtain these goals, in-site exploring uses two crawling strategies for high efficiency and coverage. Links within a site are prioritized with Link Ranker and Form Classifier to classify searchable forms. New crawling strategies like stop-early and balanced link prioritizing, are used to improve crawling efficiency and coverage. Link Ranker numbers links so that SmartCrawler can quickly find searchable forms. A high score is given to a link t hat is close to the links that directly point to pages with searchable forms. Classifying forms is to keep form focused crawling, which liters out non-searchable and irrelevant forms. Online feature construction of feature space and adaptive learning process improves the efficiency and wide coverage. Ranking involves Site and Link ranking. Site frequency and Site similarity is used for Site ranking and Link ranking is based on feature space of links with searchable forms. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes

content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here.

3. Mathematical Model

Let Crawler System C = {I, M, O} where

- Input Data Set I = {I1, I2, ……, In} ……..(1)

where
I1 = First seed site
I2 = Second seed site
In = Nth seed site

- Module M = {M1, M2} ……..(2)

Where
M1 = Site locating module
M1 = {f11, f12, f13}
where
f11 = Reverse searching function,
f12 = Ranking websites,
f13 = Classifying the web sites, here we are combining the Naive Bayes classifier with the SVM classifier
f13 = NB + SVM

- M2 = Insite exploring module

M2 ={f21, f22, f23} ……..(3)
where
f21 = Storing relevant links
f22 = Fetching ranking and reloading link frontier
f23 = Classifying web forms

- O = {O1, O2, ……..,O3}…… ……..(4)

where O1, O2, ……..On are highly relevant ranked websites

- Feature context is represented as vector of Weight W of a term t and is defined as:

W t, d = 1 + log t ft, d…… (5)
where t f , d = frequency of term t appears in document d can be U, P, A or T
where U - URL, P - path of URL, A - anchor, T - text around URL
_ Given the homepage URL of a new site s
_ Given the homepage URL of a new site s = {Us;As; Ts}, the site similarity to known deep web sites FSS, can be defined as follows:
ST(s) = Sim(U, Us)+sim(A, As)+sim(T, Ts)……..(6)

_ The function Sim(:) is computed as the cosine similarity between two vectors V1andV2 :
Sim(V1, V2) = V1.V2 / |V1| .|V2| ……..(7)

- Link ranking for link l = {Pl, Al, Tl}, the link similarity to the feature space of known links with searchable forms FSL is defined as:

LT(l) = Sim(P, Pl)+sim(A, Al)+sim(T, Tl)……..(8)

4. SYSTEM ANALYSIS

A. Dataset

The dataset used is TEL-8 dataset as mentioned in the paper from the UIUC repository. The TEL-8 dataset contains 447 deep web sources with 477 query interfaces, because a source may contain multiple interfaces.

B. Hardware

1) Memory: 8GB
2) Processor: Intel (R) Core (TM) i5 4300M CPU @ 2.60GHz

C. Software

1) Operating System: Microsoft Windows 10
2) Microsoft Visual Studio 2010
3) SQL server 2008

D. Performance Parameters

The performance of the system is measured using different parameters amongst which time Similarity score is the most important factor.

Efficiency will be gauged on following performance parameters:

1) Similarity : Number of similar websites found in a given threshold.

2) Distinct : Number of unique web-sites found for the given search keyword in the given threshold.

3) Total : Total number of websites found for the given search keyword.

E. Comparative Results

By ranking the retrieved sites and by focusing the crawling on a topic we have achieved large number of relevant pages and wide coverage. Results have been graphed to better understand the performance of Smart Crawler using only Naïve Bayes Classifier and using NBSVM classifier. A keyword is searched and crawl results of the searched keyword are plotted with the comparison as below:

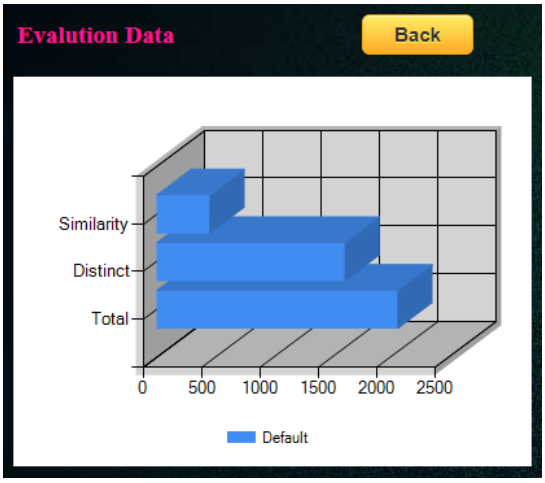

Fig. 2. Results in form of a bar graph for a searched keyword using NB Classifier

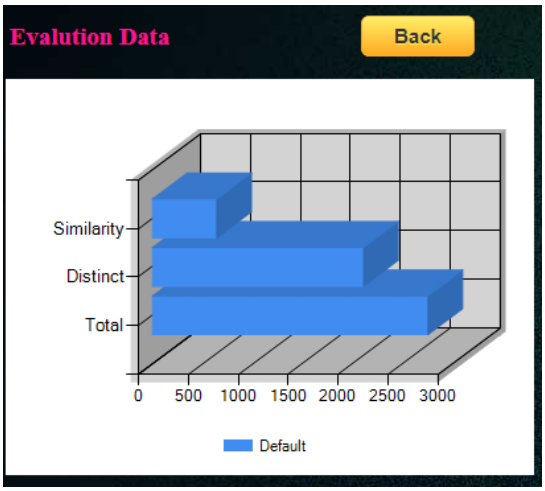

Fig. 3. Results in form of a bar graph for a searched keyword using NBSVM classifier.

3. CONCLUSIONS

Form results it can be concluded that Smart Crawler achieves both wide coverage for deep web pages and efficient crawling with NBSVM classifier compared with the NB classifier only. This has happened so because SVM is used for larger data sets and what better than online dataset would be a better choice for SVM. Thus results obtained by implementing my proposed system prove to have improved efficiency and results by 0.8 %.

ACKNOWLEDGEMENT

I take this opportunity to express my special thanks of gratitude and deepest appreciation to my project teacher, advisor Prof. Deipali V. Gore for her advice, inspiring guidance and encouragement through my research for this work. This research could not have been completed without her assistance. Finally, I would also like to thank my family and friends for their unending support.

REFERENCES

[1] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang Toward large scale integration: Building a metaquerier over databases on the web, In CIDR, pages 4455, 2005.

[2] Denis Shestakov, Databases on the web: national web domain survey, In Proceedings of the 15th Symposium on International Database Engineering Applications, pages179184.ACM,2011.

[3] Michael K. Bergman, "White paper: The deep web: Surfacing hidden value, "Journal of electronic publishing, 7(1), 2001

[4] 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 355364. ACM, 2013.

[5] Luciano Barbosa and Juliana Freire, Searching for hidden-web databases, In WebDB, pages 16, 2005.

[6] 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 441450. ACM, 2007.

[7] Balakrishnan Raju and Kambhampati Subbarao, Sourcerank: Relevance and trustassessment for deep web sourcesbased on inter-source agreement, In Proceedings of the 20th international conference on World Wide Web, pages 227236, 2011.

[8] Mustafa Emmre Dincturk, Guy vincent Jourdan, Gregor V.Bochmann, and Iosif Viorel Onut, A model-based approach for crawling rich internet applications, ACM Transactions on the Web, 8(3):Article 19, 139, 2014.

[9] Andre Bergholz and Boris Childlovskii, Crawling for domain specific hidden web resources, In Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on, pages 125133. IEEE, 2003.

[10] Sida Wang and Christopher D. Manning, Baselines and Bigrams: Simple, Good Sentiment and Topic Classification, Department of Computer Science, Stanford University.