ISSN: 2350-0328

International Journal of AdvancedResearch in Science, Engineering and Technology

Vol. 6, Issue 2, February 2019

Smart Crawler: Two stage crawler for efficiently harvesting deep web

Kalpna suman , Vyom kulshrestha, Dr. Pankaj Sharma

M.tech scholar, Department of CSE, Sachdeva institute of Technology, Mathura
Assistant Professor, Department of CSE, Sachdeva institute of Technology, Mathura
Professor, Department of CSE, Sachdeva institute of Technology, Mathura

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 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. 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. To eliminate bias on visiting some highly relevant links in hidden web directories, we design a link tree data structure to achieve wider coverage for a website. Our 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

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 [1]. More recent studies estimated that 1.9 zettabytes were reached and 0.3 zettabytes were consumed worldwide in 2007 [2], [3]. An IDC report estimates that the total of all digital data created, replicated, and consumed will reach 6 zettabytes in 2014 [4]. 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 [5], [6]. These data contain a vast amount of valuable information and entities such as Infomine [7], Clusty [8],BooksInPrint [9] 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 [10], [11], [12], [13], [14] fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) [15] and Adaptive Crawler for Hidden-web Entries (ACHE) [16] 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 [17]. 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).

II. SYSTEM ANALYSIS

A.EXISTING SYSTEM

The existing system is a manual or semi automated system, i.e. The Textile Management System is the system that can directly sent to the shop and will purchase clothes whatever you wanted.

The users are purchase dresses for festivals or by their need. They can spend time to purchase this by their choice like color, size, and designs, rate and so on.
They But now in the world everyone is busy. They don't need time to spend for this. Because they can spend whole the day to purchase for their whole family. So we proposed the new system for web crawling.

B. PROPOSED SYSTEM:

The system proposes 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. 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. To eliminate bias on visiting some highly relevant links in hidden web directories, we design a link tree data structure to achieve wider coverage for a website. Our 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. Propose an effective harvesting framework for deep-web interfaces, namely Smart-Crawler. We have shown that our approach achieves both wide coverage for deep web interfaces and maintains highly efficient crawling. Smart Crawler is a focused crawler consisting of two stages: efficient site locating and balanced in-site exploring. Smart Crawler performs site-based locating by reversely searching the known deep web sites for center pages, which can effectively find many data sources for sparse domains. By ranking collected sites and by focusing the crawling on a topic, Smart Crawler achieves more accurate results

III. SOFTWARE REQUIREMENT SPECIFICATION

A.TECHNOLOGIES TO BE USED

- HTML, CSS (Web Presentation)
- JavaScript (Client-side Scripting)
- Java (as programming language)
- JDBC, JNDI, Servlets, JSP (for creating web applications)
- Eclipse with MyEclipse Plug-in (IDE/Workbench)
- Oracle/SQL Server/Access (Database)
- Windows XP/2003 or Linux/Solaris (Operating System)
- BEA WebLogic/JBoss/WebSphere (Server Deployment)

B. **HARDWARE CONFIGURATION**

Processor : Pentium III
Clock : 500 MHZ
Ram : 128 MB

C. HARDWARE REQUIREMENTS:

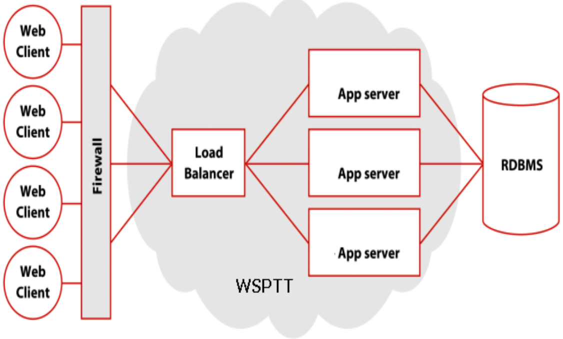
- System :Pentium IV 3.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 14' Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 1 GB.

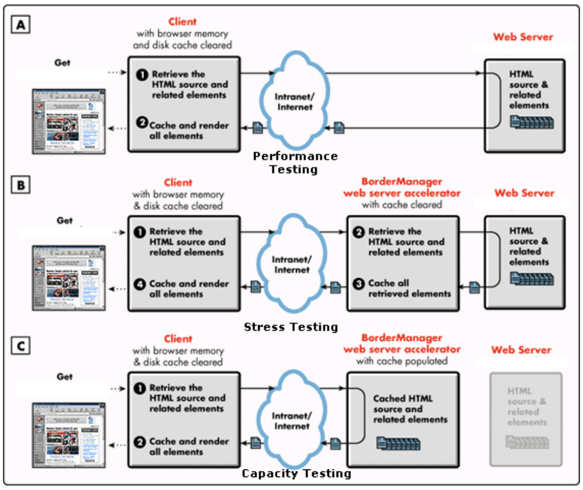
D. SOFTWARE REQUIREMENTS:

- Operating system : Windows Family.
- CodingLanguage:J2EE(JSP,Servlet,Java Bean)
- Data Base : MY Sql Server.
- IDE : Eclipse Juno
- Web Server : Tomcat 6.0

IV. SYSTEM ANALYSIS

A.ARCHITECTURE DIAGRAM

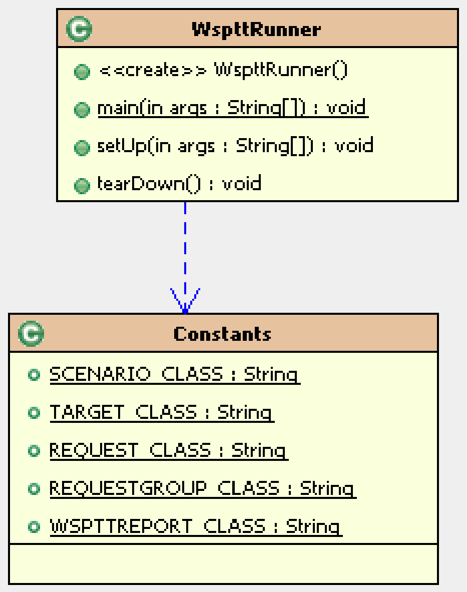



A
Client
with browser memory and disk cache cleared
Web Server
Get
1 Retrieve the HTML source and related elements
2 Cache and render all elements
Intranet/ Internet
HTML source & related elements
Performance Testing
B
Client
with browser memory & disk cache cleared
BorderManager web server accelerator
with cache cleared
Web Server
Get
1 Retrieve the HTML source and related elements
4 Cache and render all elements
Intranet/ Internet
2 Retrieve the HTML source and related elements
3 Cache all retrieved elements
HTML source & related elements
Stress Testing
C
Client
with browser memory & disk cache cleared
BorderManager web server accelerator
with cache populated
Web Server
Get
1 Retrieve the HTML source and related elements
2 Cache and render all elements
Intranet/ Internet
Cached HTML source and related elements
HTML source & related elements
Capacity Testing

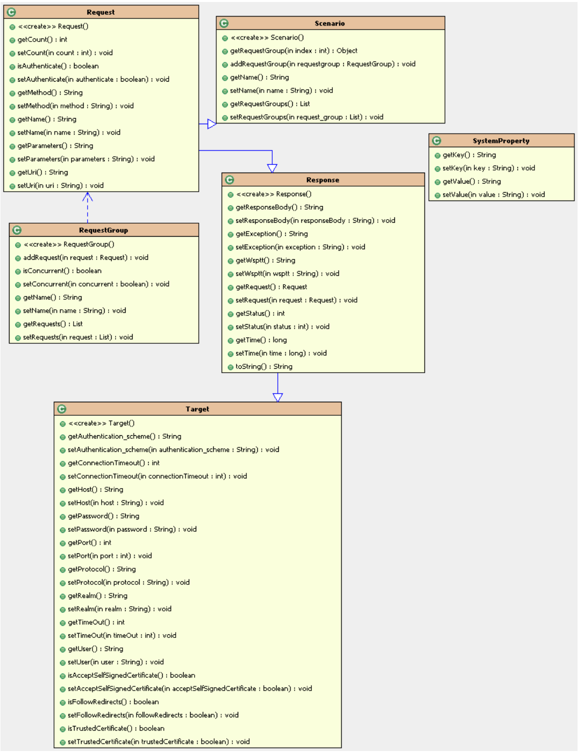
B.CLASS DIAGRAM

Class diagrams describe the structure of the system in terms of classes and objects. The servlet api class diagram will be as follows.

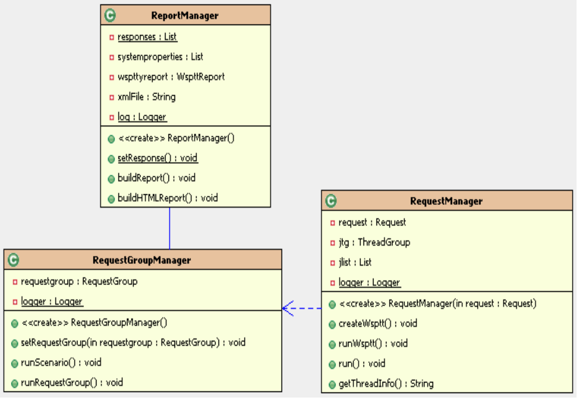
WSPTT



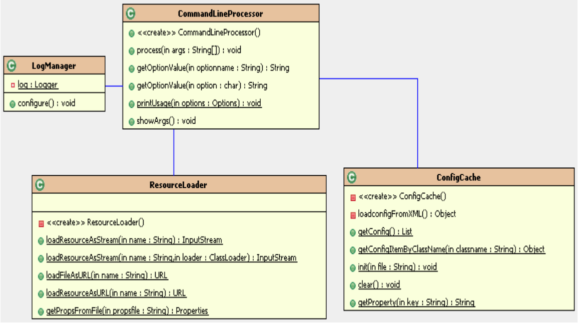
WSPTT Elements

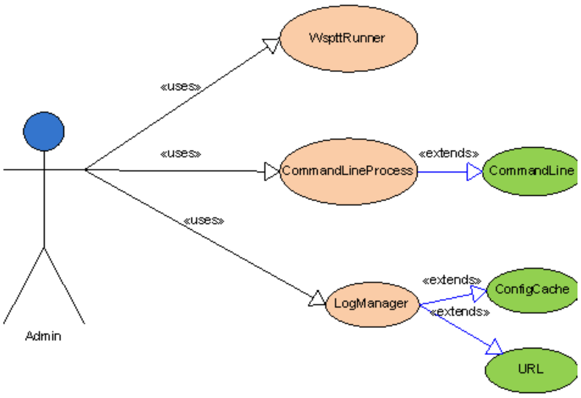

Request
<<create>> Request()
getCount() : int
setCount(in count : int) : void
isAuthenticate() : boolean
setAuthenticate(in authenticate : boolean) : void
getMethod() : String
setMethod(in method : String) : void
getName() : String
setName(in name : String) : void
getParameters() : String
setParameters(in parameters : String) : void
getUri() : String
setUri(in uri : String) : void
Scenario
<<create>> Scenario()
getRequestGroup(in index : int) : Object
addRequestGroup(in requestgroup : RequestGroup) : void
getName() : String
setName(in name : String) : void
getRequestGroups() : List
setRequestGroups(in request_group : List) : void
SystemProperty
getKey() : String
setKey(in key : String) : void
getValue() : String
setValue(in value : String) : void
RequestGroup
<<create>> RequestGroup()
addRequest(in request : Request) : void
isConcurrent() : boolean
setConcurrent(in concurrent : boolean) : void
getName() : String
setName(in name : String) : void
getRequests() : List
setRequests(in request : List) : void
Response
<<create>> Response()
getResponseBody() : String
setResponseBody(in responseBody : String) : void
getException() : String
setException(in exception : String) : void
getWsptt() : String
setWsptt(in wsptt : String) : void
getRequest() : Request
setRequest(in request : Request) : void
getStatus() : int
setStatus(in status : int) : void
getTime() : long
setTime(in time : long) : void
toString() : String
Target
<<create>> Target()
getAuthentication_scheme() : String
setAuthentication_scheme(in authentication_scheme : String) : void
getConnectionTimeout() : int
setConnectionTimeout(in connectionTimeout : int) : void
getHost() : String
setHost(in host : String) : void
getPassword() : String
setPassword(in password : String) : void
getPort() : int
setPort(in port : int) : void
getProtocol() : String
setProtocol(in protocol : String) : void
getRealm() : String
setRealm(in realm : String) : void
getTimeOut() : int
setTimeOut(in timeOut : int) : void
getUser() : String
setUser(in user : String) : void
isAcceptSelfSignedCertificate() : boolean
setAcceptSelfSignedCertificate(in acceptSelfSignedCertificate : boolean) : void
isFollowRedirects() : boolean
setFollowRedirects(in followRedirects : boolean) : void
isTrustedCertificate() : boolean
setTrustedCertificate(in trustedCertificate : boolean) : void

Report Manager

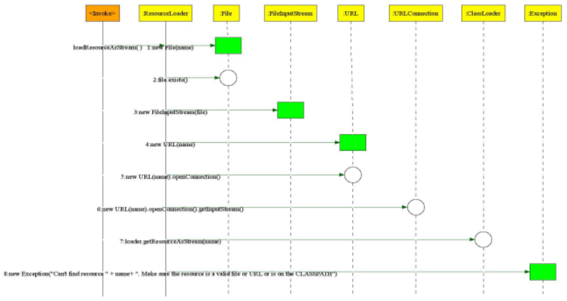


WSPTT Utility


CommandLineProcessor
<<create>> CommandLineProcessor()
process(in args : String[]) : void
getOptionValue(in optionname : String) : String
getOptionValue(in option : char) : String
printUsage(in options : Options) : void
showArgs() : void
LogManager
log : Logger
configure() : void
ResourceLoader
<<create>> ResourceLoader()
loadResourceAsStream(in name : String) : InputStream
loadResourceAsStream(in name : String,in loader : ClassLoader) : InputStream
loadFileAsURL(in name : String) : URL
loadResourceAsURL(in name : String) : URL
getPropsFromFile(in propsfile : String) : Properties
ConfigCache
<<create>> ConfigCache()
loadconfigFromXML() : Object
getConfig() : List
getConfigItemByClassName(in classname : String) : Object
init(in file : String) : void
clear() : void
getProperty(in key : String) : String



SEQUENCE DIAGRAM
RESOURCE LOADER


<Invoke>
:ResourceLoader
:File
:FileInputStream
:URL
:URLConnection
:ClassLoader
:Exception
loadResourceAsStream( )
1:new File(name)
2:file.exists()
3:new FileInputStream(file)
4:new URL(name)
5:new URL(name).openConnection()
6:new URL(name).openConnection().getInputStream()
7:loader.getResourceAsStream(name)
8:new Exception("Can't find resource " + name+ ". Make sure the resource is a valid file or URL or is on the CLASSPATH")

V.CODING & IMPLEMENTATION

A. TECHNOLOGIES USED
JAVA TECHNOLOGY

Initially the language was called as "oak" but it was renamed as "Java" in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.
- Finally, Java is to Internet programming where C was to system programming.

IMPORTANCE OF JAVA TO THE INTERNET

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

OVERALL DESCRIPTION

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The .Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

VI. CONCLUSION

In this paper, we propose an effective harvesting framework for deep-web interfaces, namely Smart- Crawler. We have shown that our approach achieves both wide coverage for deep web interfaces and maintains highly efficient crawling. SmartCrawler is a focused crawler consisting of two stages: efficient site locating and balanced in-site exploring. SmartCrawler performs site-based locating by reversely searching the known deep web sites for center pages, which can effectively find many data sources for sparse domains. By ranking collected sites and by focusing the crawling on a topic, SmartCrawler achieves more accurate results. The in-site exploring stage uses adaptive link-ranking to search within a site; and we design a link tree for eliminating bias toward certain directories of a website for wider coverage of web directories. Our experimental results on a representative set of domains show the effectiveness of the proposed two-stage crawler, which achieves higher harvest rates than other crawlers. In future work, we plan to combine pre-query and post-query approaches for classifying deep-web forms to further improve the accuracy of the form classifier.

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, 9(4):8–12, 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, 7(1), 2001.
[6] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In Proceedingsof the sixth ACM international conference on Web search and data mining, pages 355–364. ACM, 2013.
[7] Infomine. UC Riverside library. http://lib-www.ucr.edu/, 2014.
[8] Clusty’s searchable database dirctory. http://www.clusty. com/, 2009.
[9] Booksinprint. Books in print and global books in print access. http://booksinprint.com/, 2015.
[10] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In CIDR, pages 44–55, 2005.
[11] Denis Shestakov. Databases on the web: national web domain survey. In Proceedings of the 15th Symposium on International Database Engineering & Applications, pages 179–184. ACM, 2011.
[12] Denis Shestakov and Tapio Salakoski. Host-ip clustering technique for deep web characterization. In Proceedings of the 12th International Asia-Pacific Web Conference (APWEB), pages 378–380. IEEE, 2010.
[13] Denis Shestakov and Tapio Salakoski. On estimating the scale of national deep web. In Database and Expert Systems Applications, pages 780–789. Springer, 2007.