Design and Implementation of a Tool for Web Data Extraction and Storage using Java and Uniform Interface

S. Jeyalatha
Department of Computer Science and Engineering
BITS-PILANI, Dubai, U.A.E

B. Vijayakumar
Department of Computer Science and Engineering
BITS-PILANI, Dubai, U.A.E

Munawwar Firoz
BITS-PILANI
Dubai, U.A.E

ABSTRACT
This paper deals with Web Content Mining. While browsing the web, the user has to go through many pages of the Internet, filter the data and download related documents and files. This task of searching and downloading is time consuming. Sometimes the search queries call for specific option, say, limiting search to few links. To reduce the time spent by users, a web extraction and storage tool has been designed and implemented in Java, that automates the downloading task from a given user query. The Test Scenario has been presented with various keywords. The present work can be a useful input to Web Users, Faculty, Students and Web Administrators in a University Environment.

General Terms
Structured Data, Unstructured Data, Web Content Mining, Web Data Extraction.

Keywords
Content Mining, Data Extraction, HTML, Web Data Retrieval, Web Information Extraction.

1. INTRODUCTION
The amount of information available in the Web increases continuously and therefore requires an accurate textual representation of the investigated Web pages. Advertisements, navigational elements complicate the task of extracting the Web page's content. Web content extraction is concerned with extracting the relevant text from Web pages by removing unrelated textual noise like advertisements, navigational elements, contact and copyright notes. Application of data mining techniques to the World Wide Web is referred to as Web Mining [13]. Web Mining can be broadly defined as the automated discovery and analysis of useful information from the web documents and services using data mining techniques [15]. It discovers potentially useful and previously unknown information or knowledge from web data. Web mining employs the technique of data mining into the documents on the World Wide Web [16].

Web Mining tasks can be classified into three types based on which part of the Web to mine [10]. They are Web Content Mining, Web Structure Mining and Web Usage Mining. Web Content Mining deals with discovering useful information or knowledge from web page contents [18]. Web content consists of different types of data such as textual, image, video, audio, metadata and hyperlinks. Web Structure Mining focuses on the hyperlink structure of the web [17]. Web Usage Mining (also known as Web Log Mining), is the process of extracting interesting patterns in Web access logs. Usage information can be used to restructure a web site in order to better serve the needs of users of the site [20]. Data on the web is classified into three types namely Structured, Unstructured and Semi-structured. Structured data is in the form of list, tree, table and Database. An Unstructured data contains loose text and does not follow any particular grammar for representation for a web page [19]. Semi- Structured data do not have a pre defined structure [14]. HTML and XML comes under this category.

The paper is organized into nine sections followed by the References. Section 2 deals with Objectives and Motivation, section 4 describes the problem relating to academic search application, section 5 deals with an algorithm for web data extraction called Search_Assist, section 6 specifies the Implementation, section 7 presents the Test Scenario, section 8 describes the Practical Application and section 9 summarizes the paper.

2. OBJECTIVES AND MOTIVATION
The present work is intended to meet the following objectives:

1. To design and implement an algorithm for Web Data Extraction to download the files and place them in their respective folders like XML, HTML and Text data.
2. To search for a query based on filtering parameters and the number of links to be found should be taken as user input. The links obtained as output must be downloaded onto the local machine and the files must be categorized into folder based on file type.
3. To identify the Academic Search related functions where Web Data Extraction algorithm can be applied effectively.

Most web extraction tools include a web crawler. Web crawling refers to the recursive process of downloading web pages from a set of URL's , extracting the link found within them and adding them to the set of links waiting to undergo the same process. The crawled pages are then indexed according to content. These files are then served to the users based on the user's keywords / query.

Web crawling involves searching a very large solution space which requires a lot of time, hard disk space and lot of usage of resources in general. Hence using existing search engines as the back-end is more practical approach to Web content extraction. It gives the opportunity for the user to store the search results in a local server or public directory. This way searches involving the same keywords can be performed locally rather than

repeated searches and frequent download. This saves time and resources to a great extent.

3. ACRONYMS

DOC : Microsoft Word File.

DOM : Document Object Model.

GUI : Graphical User Interface.

HTML : Hyper Text Markup Language.

HTTP : Hyper Text Transfer Protocol.

Java SE6: Java Platform Standard Edition 6.

MIME : Multipurpose Internet Mail Extension.

PDF : Portable Document Format.

PPT : Power Point File format.

TXT : Text file.

UI : User Interface.

URL : Uniform Resource Locator.

WWW : World Wide Web.

XLS : Microsoft Excel Spreadsheet.

XML : eXtensible Markup Language.

4. PROBLEM DESCRIPTION

The Flow diagram for the proposed system Search Assist is shown in *Figure 1*. The keywords are taken as input from the user either from the file or GUI. It generates URL from the search query given and the search options selected by the user. Using Wget software, the files are downloaded [2]. GNU Wget is a free software package for retrieving files using HTTP, FTP. It is a non-interactive command line tool that is called from other programs [1]. The downloaded search results in HTML form are converted to XML using HTML Cleaner [3]. HTML on the web is not well formed and not suitable for further processing because it contains missing quotes and unclosed tags in the document. HTML Cleaner is an open source HTML parser written in Java which accepts HTML document and corrects individual elements and produces well formed XML [11]. XPath tools are used to find the links in the XML document [4]. A link is chosen and Wget is used to download the document and save the output messages to a file. The file extension of the downloaded file is determined from the MIME type [8]. The file is placed in folder of respective file extension.

5. ALGORITHM: Search_Assist

The present work deals with implementation of an algorithm Search_Assist.

5.1 Reading of Inputs: Keywords, Number of links.

5.2 Generation of Output: Downloaded files in respective folder.

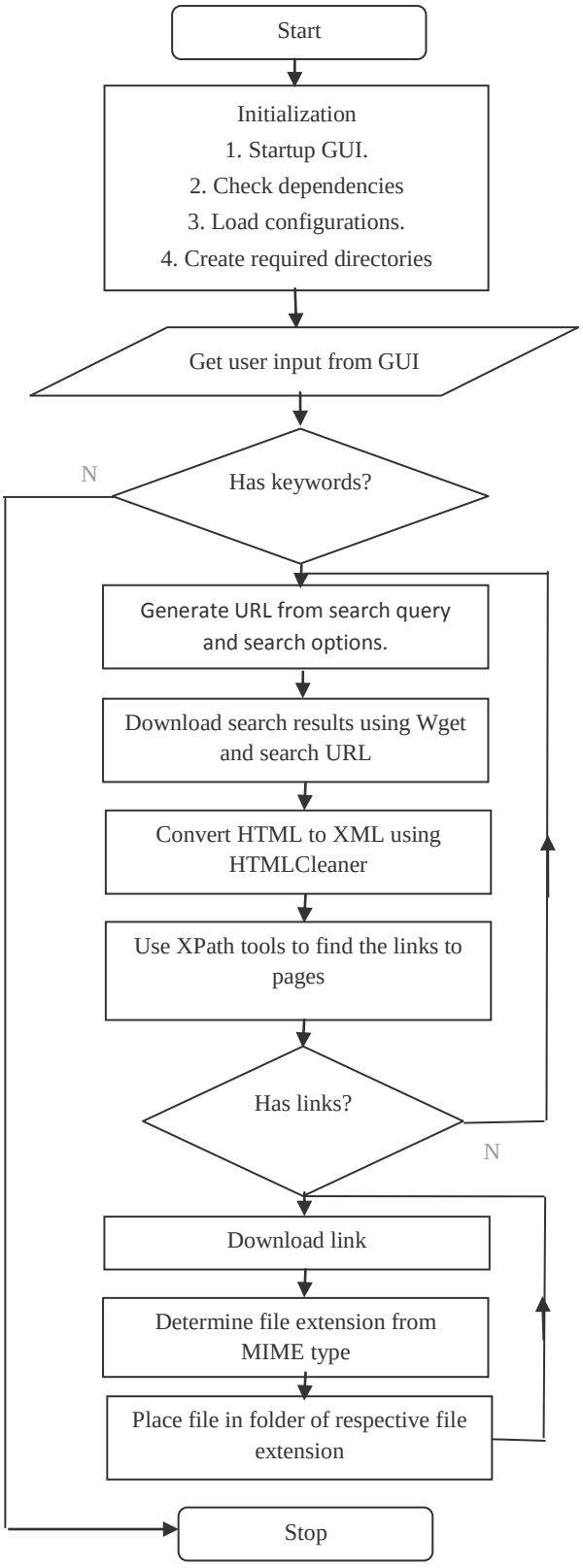

Figure 1: Search Assist Flow Diagram

5.3 Procedure:

1. Generate the search URL from the keywords and various options provided by the user.
2. Use Wget to retrieve the search results as HTML file using the search URL.
3. Use HTMLCleaner (an external party utility), to convert the HTML to a clean XML document.
4. Use XML XPath tools to find the links in the document [5].
5. Choose a link and use Wget to download the documents. Also save the output messages to a file.
6. Determine the MIME type of the downloaded file from the output file. And then check if the type is associated with a file extension from req/mime.txt. (This file contains the mapping between common MIME types and their file extension. Some of them are HTML, PDF, PPT, DOC, TXT, XML, XLS) [9].
7. If file extension is found, create a folder for the extension (if it doesn't exist) and place them in the folder. If file extension wasn't recognized, place the downloaded file into 'Unrecognized' folder.
8. Repeat the same for other links (Step 5)
9. If the keywords have been specified by file input, repeat this process for other keywords (from Step 1).

5.4 Actions of the Algorithm:

***Step 1*:** User enters keyword and presses search button. A thread is spawned to do the search (so that it doesn't block the UI). All the steps here onwards happen within the thread.

***Step 2*:** From the GUI options, a link to Google with the search options is generated. For example the link would be:

http://www.google.com/search?hl=en&safe=active&q=JSON+schema&num=10&start=0
Where

'hl=en' denotes the language preference is English.

'safe=active' sets safe search.

'q' = the keywords.

'num' is the number of search results to show.

'start' indicates the search result to start from. That is we can start from the 10th search results and display the next 20 links by setting num=20. Results 10 to 30 will be shown in that case.

***Step 3*:** Send the link to Wget and Wget will extract the resulting html page which contains the search results.

Step 4: Parsing HTML is a difficult task and doesn't have the flexibility and programming tools required to extract content easily. Hence it is practical to convert HTML to XML. For the purpose a third party utility known as 'HTMLCleaner' is used to convert HTML to XML.

Step 5: Use XPath to extract the links (search results) from the Google search page [6].

Step 6: Begin iterating through the links.

Step 7: Download the link with Wget and retrieve the MIME type of the file from the Wget output (MIME type is given along with the HTTP header while downloading the file. The Wget output is saved onto a log file which is retrieved, after download, by Search Assist).

Step 8: Determine the file extension of the downloaded file from the MIME type. MIME-to-file extension mapping has been prewritten in a file. Only a few, and most important types are currently recognized.

Step 9: Rename the file with the extension and move the file to a folder named after the respective extension. If the folder doesn't exist, it is created on the run.

Step 10: Go to step 6 and iterate through the rest of the links.

Step 11: Go to step 1 and check if any more queries are to be searched. When no more queries remain, quit the thread.

Log messages are shown during the entire search process and the time taken for the process to complete is also shown. When the application is closed, the log messages of the session are saved onto a file for later use. Thus, the complete history of all the searched and downloaded links can be tracked.

6. IMPLEMENTATION

The Search Assist algorithm has been implemented using JavasSE Runtime 6 and GNU Wget 1.12. *Table 1* illustrates the various functions and their actions in Search Assist using Java [7].

Table 1: Search Assist: Functions and their Actions using Java

Functions	Description
makeConfigFileTemplate	This function is invoked to set the default properties and their values before loading the configurations from file or before creating a new configuration file.
saveConfigFile	Save configurations to config.ini (file)
saveData	Invokes saveConfigFile and also takes all log messages from UI and appends on to log file.
search	Search button from UI invokes this function. This function creates a new thread and a new instance of *downloadThread* for initiating the search.
downloadThread	Class that deals with the search.
makeSearchURL	From the options given by user through the UI and the search query specified, this function creates a partial URL to Google search. This function is called from the *run* function and the result of this

	function is saved to downloadThread.url.
downloadLinks	With the given XPath expression, offset and number of results to show as it's parameters, this function does the following: 1. Adds the offset and number of results to the URL in downloadThread.url. 2. Calls Wget to download Google search page. 3. Uses HTMLCleaner to convert html to XML. 4. Build DOM Document from the XML and use the specified XPath expression to find the search results (which are just links) from the page. 5. Traverses through the links and downloads each link, For each link, the following is done: a. Determines their MIME type of the downloaded link. b. Maps the MIME to a file extension from a predefined set of MIME-to-file extension mapping saved in a file. c. Renames the file to required file extension and moves the file to a folder based on the file extension.
searchThisURL	Using downloadThread.url, this function manages the search. It splits and limits the search to 100 links per request. That is, it calls *downloadLinks* for every 100 links with the required parameters.
run	Manages the search one search query at a time. For each search query, it first calls *makeSearchURL* and saves the result to downloadThread.url and then it calls *searchThisURL*.

Figure 2 shows the User Interface with various search options. There are two types of parameters namely Mandatory and Optional. The various GUI options are explained in the following section.

Mandatory Parameters

1. *Keywords* (either by typing the query in the text field provided or specify a text file containing queries where each query is on a new line): *This query is sent to Google as ' q' URL parameter.*

2. *Save files to*: User specifies a path to a directory on the local machine. All files (log files, downloaded files etc) are saved into this directory.

3. *Use*: User is given an option to choose between Google Web search and Google Scholar.

Optional Parameters

1. *Number of Results*: Number of search results to return. Default is 10. This parameter is sent as 'num' Google URL parameter. Even though this parameter is optional, the program always sends this parameter to Google.

2. *Occurrence*: User can specify where to find occurrences of the search query. There are 4 options, namely, "On Page Title", "On Page Text", "On Links To Page" and "Anywhere". "Anywhere" is the default and it includes occurrences on page title, text and links to pages.

3. *Site:* User can specify the web sites that must be searched. Each web domain is separated by a comma. By default, all and any websites are included in the search.

Options for Google Scholar

1. *Topic*: User can search for a specific field of interest, say mathematics or medical science. The Google URL Parameter for this option is 'as_subj'.

2. *Author*: User can specify the author of the book/article which he is searching for.

3. *Publication*: Publication of the book.

4. *Publication Year*: Search for publications **between** two specified years. Example: 2008 to 2010.

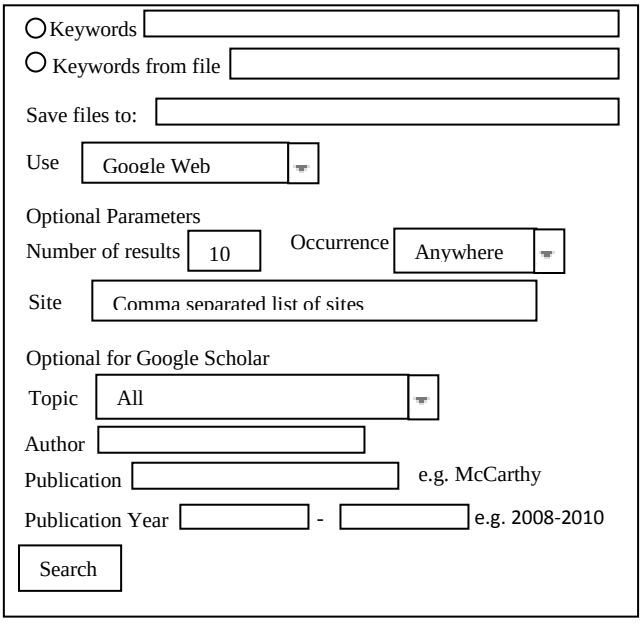

Figure 2: User Interface with various search options

7. TEST SCENARIO

The Test Scenario involved many test inputs involving keywords. The keywords considered here include the following: "Higher Education", "Special Interest Group" and "Conference Alerts".

Figure 3 illustrates the log messages recorded during the entire search process and the total time taken for extraction and storage process. When the application is closed, the log messages of the session are saved onto a file for later use. Thus the complete history of all the searched and downloaded links can be tracked. The downloaded file size is also displayed in log file.

#Wed Mar 16 13:39:08 GST 2011

Current Directory: C:\Documents and Settings\sjeyalatha\Desktop\SearchAssist2

**********Starting download**********

Search URL: http://www.google.com/search?hl=en&safe=active&q=introduction+to+graphic+programming&num=50&start=0

1. Title: Notes for a Computer Graphics Programming Course

Link: http://www.cs.csustan.edu/~rsc/CS3600F00/Notes.pdf

---

mime: application/pdf

File extension: pdf

File moved to 'K:\Users\Mogral\Desktop\search data\pdf\Notes for a Computer Graphics Programming Course.pdf'

File size ~ 3444 KB

2. Title: Introduction to SAS IML Graphics Programming

Link: http://javeeh.net/sasintro/intro110.html

---

mime: text/html

File extension: html

File moved to 'K:\Users\Mogral\Desktop\search data\html\Introduction to SAS IML Graphics Programming.html'

File size ~ 5 KB

3. Title: CS 112 Introduction to Computer Graphics Programming Assignment ...

Link: http://www.ics.uci.edu/~majumder/CG/assn/W11-PA3.pdf

---

mime: application/pdf

File extension: pdf

File moved to 'K:\Users\Mogral\Desktop\search data\pdf\CS 112 Introduction to Computer Graphics Programming Assignment ....pdf'

File size ~ 16 KB

**********Done**********

Total time taken (in milliseconds)=189807

Total download size=17982 KB

Figure 3: Log Messages for the Search Assist Algorithm

8. PRACTICAL APPLICATION

A University consists of different types of users such as Faculty, Students, Staff, Web Administrator and Librarian [12]. Each user has their requirements while browsing information on the Internet. Web Mining helps in extracting information according to user's preferences.

Table 2 shows the applicability of Search Assist Algorithm in a University Environment.

Table 2: User Categories and Tasks

User Categories	Tasks
Student	Literature Review.
Faculty	Lecture Notes.
Librarian	Building of Academic Related Data Warehouse.
Web Administrator	Usage Analysis from Log files.
Staff	Search on specific topics.

The experimental results for Web Extraction and Storage functions are illustrated in *Table 3*.

Table 3: Experimental Results for Web Extraction and Storage functions

Keywords	No. of links	Size of downloaded files in KB	Total time taken in milliseconds
Web Mining	48	15094	296688
Web Structure Mining	50	28059	595757
Web Content Management	50	2125	175498
Web Content Mining	50	22410	147736
Masters in Computer Engineering-Europe	10	409	27232
Post Graduate Degrees USA	10	632	23732
IEEE Conference UAE	5	335	19493
IETF Conference	5	264	13100

Time complexity = O(Total no. of links).

9. CONCLUSION

The present work analyzes the ways of extracting web data and storing them using Java and Uniform Interface. The application has been tested successfully using Academic Search keywords. This work can be extended further to handle image, audio, video and data.

10. REFERENCES

[1] Hrvoje Nikšić andGiuseppe Scrivano, GNU WGet, http://www.gnu.org/software/wget/

[2] GNU Wget for windows, http://gnuwin32.sourceforge.net/packages/wget.htm

[3] HTMLCleaner Team, HTMLCleaner, http://htmlcleaner.sourceforge.net/

[4] James Clark and Steve DeRose, W3C XPath Specifications, http://www.w3.org/TR/xpath/

[5] Elliotte Rusty Harold, Java XPath API, http://www.ibm.com/developerworks/library/x-javaxpathapi.html

[6] XPath Tutorial, http://www.w3schools.com/xpath/default.asp

[7] Introduction to Java Programming Y Daniel Liang, Prentice Hall Europe 2007.

[8] N. Freed, N. Borenstein, MIME Format specification, http://www.ietf.org/rfc/rfc2045.txt

[9] List of MIME Types, http://reference.sitepoint.com/html/mime-types-full

[10] R. Kosala., H. Blockeel. Web Mining Research: A Survey, *ACM SIGKDD Explorations*, 2000, 2:1-15.

[11] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, "Extensible Markup Language (XML) 1.0", http://www.w3.org/TR/xml, 2008.

[12] S.Jeyalatha., B. Vijayakumar., E.A. Hazarika. Design of an Interface for Page Rank Calculation using Web Link Attributes Information, Informatica Economica, Vol 14, Issue 3, 2010.

[13] Cooley R., Mobasher B., Srivastava J. Web Mining : Information and Pattern Discovery on the World Wide Web. *ACM SIGKDD Explorations Newsletter*. 2000. 1: 12-23.

[14] K. Pol, N. Patil, S. Patankar, C. Das, "A Survey on Web Content Mining and Extraction of Structured and Semistructured Data", *First International Conference on Emerging Trends in Engineering and Technology*, ICETET, Nagpur, India, pp. 543-546, 2008.

[15] S. Brin., L. Page. The Anatomy of a Large Scale Hypertextual Web Search Engine, Proc 7th International WWW Conference, Brisbane, Australia, pp 107-117, 1998.

[16] L.K. Joshila Grace, V. Maheshwari, D. Nagamalai, Analysis of Web Logs and Web User in Web Mining, International Journal of Network Security & Its Applications,Vol 3. Issue 1, Jan 2011.

[17] S. B. Boddu, V. P. K. Anne; R. R. Kurra, D.K. Mishra, "Knowledge Discovery and Retrieval on World Wide Web Using Web Structure Mining", Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation (AMS), pp 533, June 2010.

[18] B. Singh, H.K. Singh, "Web Data Mining research - A survey" IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Dec 2010.

[19] S.Jeyalatha., B. Vijayakumar., Zainab A. S., Design Considerations for a Data Warehouse in an Academic Environment, World Academy of Science, Engineering and Technology, Issue 71, pp 421-425, Oct 2010.

[20] V. Sathiyamoorthi, V. M. Bhaskaran, "Data Preparation Techniques for Web Usage Mining in World Wide Web – An Approach", International Journal of Recent trends in Engineering, Vol 2, No 4, Nov 2009.