65

Automatic Data Mining on Internet by Using PERL

AQ1

Nam Pham
Auburn University

Bogdan M. Wilamowski
Auburn University

65.1 Introduction

With the tremendous growth in available information to the masses, the question is how users can search the useful information in the shortest time [PW09], [NGW08]. In other words, making use of consolidated information requires such substantial efforts since the web pages are generated for visualization not for data exchange [KCSG07]. To reach this goal requires methods developed to optimize a user's searching process. This chapter introduces a method known as the data mining robot (DMR) to extract and process data by using PERL scripting language. The DMR can be understood quickly as a software program that serves for mining data automatically. Particularly, with data mining from servers, this method does not use any browser to handle the Web, but does so directly by using PERL modules (software programs are written for a specific function) such as LWP. The use of these modules turns the DMR into an effective solution to extract data with such an accelerated speed. AQ2

The procedure of execution of the DMR can be divided into three steps. (1) Data collection: to extract all information from the data source. (2) Data Filtering: to extract the useful information built in step 1. (3) Data processing: to process and sort the extracted information in a format that is effective for users.

DMR can be written in scripting languages such as PHP, PERL, etc., or C. And PERL is one of the most powerful tools in data manipulation with the powerful regular expression. To run DMR is simple. Programmers only need to install the PERL scripting language, which is an interpreted language, it is parsed and executed at runtime instead of being compiled into binary form and then run. Moreover, this is the open-source software for the users with the standard modules, which also come with PERL. Like the built-in functions, these modules provide the users with hundreds of the prewritten resources. The modules are made up of PERL code written in a way to conform to certain conventions so the users can access that code from their program. The PERL modules provide the users with a great deal of the prewritten code and are stored in the files with extension ".pm." Users can load such the modules into their code by using the "use" statement.

65.1.1 PERL Scripting Language

PERL, which stands for “Practical Extraction and Report Language,” was written by Larry Wall, a linguist working as a systems administrator for NASA in the late 1980s, as a way to make report processing easier. It is a power language for doing data manipulation tasks. It is portable, accessible, and excellent at handling textual information. It has been used with good success for system administration tasks on Unix systems, acting as glue between different computer systems, World Wide Web CGI programming, as well as customizing the output from other programs.

PERL is a prominent Web programming language because of its text processing features. Handling HTML forms is made simple by the CGI.pm module, a part of PERL’s standard distribution.

65.1.2 Regular Expressions

PERL is especially good at handling text, and, in fact, that is what it was originally developed for. Regular expressions are a big part of text handling in PERL. Regular expressions let users work with pattern matching (that is, comparing strings to a test string—or a pattern—that may contain wildcards and other special characters) and text substitution, providing a very powerful way to manipulate text under programmatic control [H99]. Unfortunately, using regular expressions in PERL is one of the areas that programmers find the most daunting. Even relatively straightforward regular expressions can take some time to comprehend.

```
String1=~ m/String2/
```

“`=~ m//`” operator means that `string1` will be compared with the pattern that is `string2`. If the match is true, `string2` will be a part of `string1`, but not vice versa. In other words, the pattern matching operator is always used to compare the pattern on the right hand side with the pattern on the left hand side to see if it fits or not.

Assuming that there are 100 different names in a list and we want to know if John is in this list or not.

```
$a="Peter, Jack and Tom";
$b="Peter";
if ($a=~m/$b/){print ("yes\n");}
elsif($a!~m/$b/) {print("no\n")};
Result:
Yes
```

Instead of using the standard string as a pattern, users can replace it by “wildcards.” The wildcards are just special characters to generalize a string. For example, a user may want to know if there are any numbers in the string. In this case, a user does not know what the number is in the string, so they cannot use a specific number as the pattern to compare, but they can use `(\d+)` instead. `d` is an integer, and + sign means that the integer maybe of length 1 or more.

```
$a="Peter, Jack and Tom";
if ($a=~m/(\d+)/){print ("yes\n");}
elsif($a!~m/(\d+)/) {print("no\n")};
Result:
no
```

Below is a list of some widely used wildcards:

. match any character
`\w` match word character
`\W` match non-word character

`\s` match whitespace character
`\S` match non-whitespace character
`\d` match digit character
`\D` match non-digit character
+ match one or more times

65.1.3 Web Browser

The interaction between PERL and the Internet is one of the most special attributes of this language. Using PERL, you can easily create Internet applications that use File Transfer Protocol (FTP), e-mail, Usenet, and Hypertext Transfer Protocol (HTTP), and even browse the Web. There are a lot of modules supporting programmers to do that. Much of this power comes from CPAN modules that can be downloaded and installed from the Web. Since users often download files and data from the Internet, the question is that can PERL emulate a web browser to copy all web sources into local files? The modules as LWP::Simple or LWP::UserAgent have enough capability to do that.

By using modules, the DMR does not use Web. These modules have the capability to download a web page. In other words, they can emulate browsers. The following example illustrates the download of the main FAQ index at CPAN by using the "get" function from the module LWP::Simple and store that web page into file *name.html.*

```
use LWP::Simple;
$add="http://www.cpan.org/doc/FAQs/index.html";
$content=get("$add");
open FILEHANDLE,">name.htm";
print FILEHANDLE $content;
close FILEHANDLE;
```

An address is called into the subroutine and content. This subroutine uses "get" function from LWP::Simple to copy the web source into the variable $content, and then the content of this web page is stored into the file name.html. The final result will be a file having the same content as the web page http://www.cpan.org/doc/FAQs/index.html.

Instead of using LWP::Simple to emulate a browser, we may also use the wget system call. GNU wget is an open source software package for retrieving files using HTTP, HTTPS, and FTP, the most widely-used Internet protocols. Its main function is to link a certain web page with an address input and copy all web sources of that web page into a file. It allows the users to log off from the system and let wget finish its work. This is extremely effective and time-saving when users have to transfer a lot of data from different Web links. In contrast, other web browsers always require the users' constant presence. Generally, wget works almost the same as the browser built from LWP::Simple module. However, it gives the users more options [WSC05].

Below is the basic structure to call wget from a PERL script. "–O" means that the documents will not be written to separate files, but all will be concatenated together and written to one file while "–q" turns off wget output to the prompt screen.

```
$add= ="http://www.cpan.org/doc/FAQs/index.html";
$fname= "filename.htm";
system("wget.exe", "-q", "-O", $fname,$add);
```

By assigning values to the variables $add and $fname as an address of a web page and a name of a file and using the "system" command, the new html file will contain the same content as that web link. Once the new html file contains all data that the users want, they can open and copy this data into an array. From here, the users can extract all information they want by using PERL commands.

65.2 Examples

65.2.1 Extract E-Mail Addresses from Excel Files

An excel file has three columns: the first is the order number, the second contains the e-mail addresses of the authors, and the third displays the categories of the author's interest. In this example, there are seven authors who are interested in three categories: power electronic converters, signal processing, and neural networks. Assume that you want to call for papers on neural networks by sending an e-mail to AQ3 these authors. Our work is to extract the e-mail addresses of these authors from the table (Table 65.1).

There are many ways to get this job done. The traditional way is to copy this excel file into a text file and use regular expressions to extract the desired data. This work becomes difficult and complicated if the excel file has more columns. However, it can be much simpler with PERL by using the module "Win32::OLE." This package is not included in the standard PERL library but can be downloaded from CPAN. What it does is to open an excel file, but a file has many different sheets, the users have to define which sheets need to be opened by modifying the sheet number or the sheet name. Once this sheet is opened, each column can be saved into different arrays. From these arrays, the e-mail addresses of authors interested in neural networks can be retrieved (Table 65.2).

```
use Win32::OLE qw(in with);
use Win32::OLE::Const 'Microsoft Excel';
$Win32::OLE::Warn = 3;# die on errors…
$mydir='C:/ALL BACK UP/Industrial Electronics on Trans/';
$filename=$mydir."IESdatabase08.xls";
if (-e "$filename")
    {
     # get active Excel application or open Excel application
     my $Excel = Win32::OLE->
     GetActiveObject('Excel.Application')
        || Win32::OLE->new('Excel.Application', 'Quit');
        $Excel->{'Visible'} = 0; #opened file is visible
        # open Excel file
        my $Book = $Excel->Workbooks->Open("$filename");
        # select worksheet number 2 (you can also select a
        worksheet by name)
        my $Sheet = $Book->Worksheets(2);
        # count the number of columns
        my $LastCol = $Sheet->UsedRange->Find({What=>"*",
                       SearchDirection=>xlPrevious,
                       SearchOrder=>xlByColumns})->{Column};
        # count the number of rows
        my $LastRow = $Sheet->UsedRange->Find({What=>"*",
                       SearchDirection=>xlPrevious,
                       SearchOrder=>xlByRows})->{Row};
        $totalrow=$LastRow;
        foreach my $row (1..$LastRow)
            {
             foreach my $col (1..$LastCol)
                 {
                    next unless defined
                    $Sheet-> Cells(1,$col)->{'Value'};
```

```
                        $title_row=$Sheet->Cells(1,$col)->{'Value'};
                        #extract manuscript and save in an array
                        if ($title_col=~ m/Manuscript/)
                        {$area[$row-1]=$Sheet->Cells($row,$col)
                          ->{'Value'};}
                        #extract email and save in an array
                          if ($title_col =~ m/Email Address/)
                        {$email[$row-1]=$Sheet->Cells($row,$col)
                          ->{'Value'};}
                  }
            }
         $Excel->Quit();
    }
for($n=0;$n<$LastRow;++$n) //search data from saved arrays
     {
          if ($area[$n]=~ m/Neural Networks/)
             {print ("$email[$n]\n");}
     }
Result
```

65.2.2 Extract Data from PDF Files

In data mining, we encounter different types of data in different formats; data could be in PDF format, Doc format, etc. The issue here is that these types of data have to be converted into the general text format that can be readable and reusable. There are two solutions to handle this problem. One is done manually and one is done automatically.

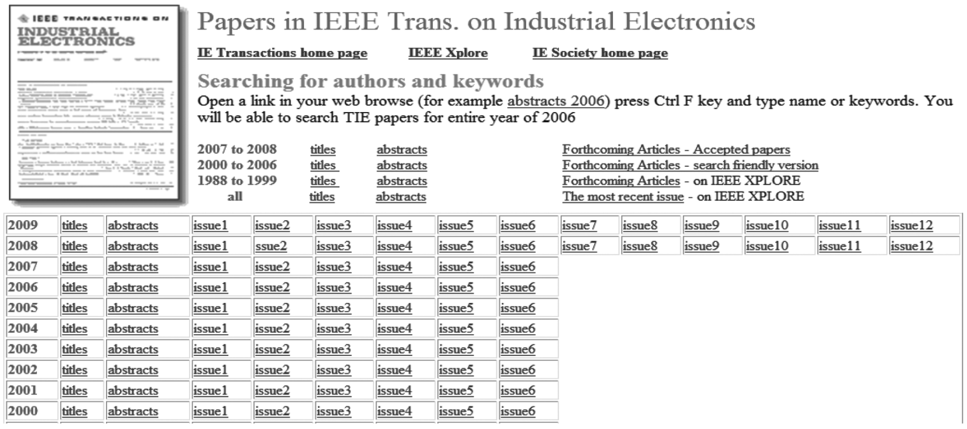
With the manual solution, users have to copy and paste from the PDF format to the text format. However, this solution is time-consuming and impractical as the number of PDF documentations is scaled up. Users can call wget to open the PDF file and copy its content into the text file as they often use to handle text file or html file, but the result file will be corrupted in this way. Because of this problem, the straight solution is not applicable.

With the automatic solution, the work can be done in a much simpler way. In order to do that, users have to use the module CAM::PDF in their code. This module allows users to convert the PDF file into a text file without corruption. In this way, users can preserve the original data as given in the PDF. When the data is converted, it is still reusable.

```
use CAM::PDF;
use CAM::PDF::PageText;
$filename = "C:/ALL BACK UP/Hung Database/filename.pdf";
my $pdf = CAM::PDF->new($filename);
my $pageone_tree = $pdf->getPageContentTree(1);
open TEST, ">", "test.txt" or die $!;
print TEST CAM::PDF::PageText->render($pageone_tree);
close TEST;
```

65.2.3 Extract Paper Information from IEEE XPLORE

PERL is used not only to mine data from files in different formats, but also to extract data directly from webs through a network connection. The name for this functionality is Internet robot (IR). Any IR can be connected to servers remotely by imbedding modules such as LWP::Simple, LWP::UserAgent,



Papers in IEEE Trans. on Industrial Electronics

IE Transactions home page IEEE Xplore IE Society home page

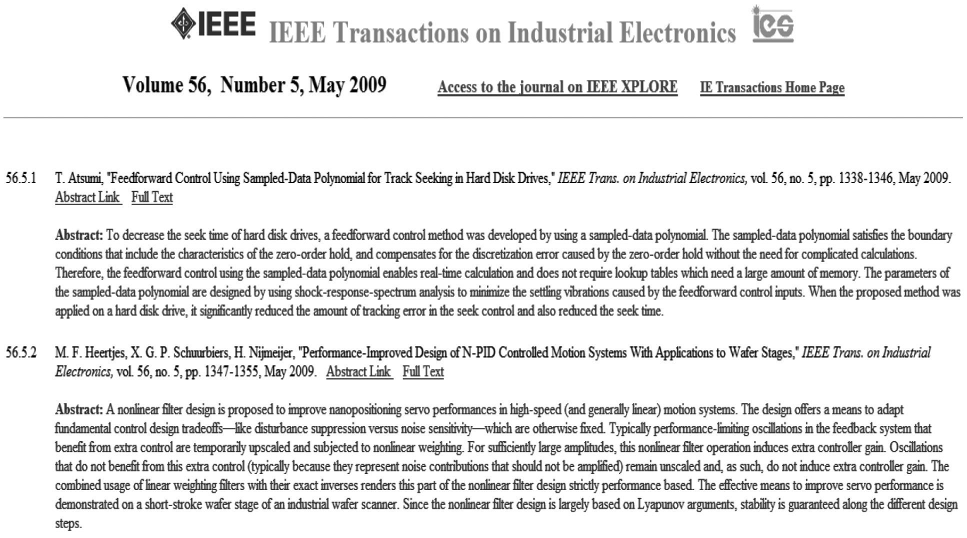
Searching for authors and keywords

Open a link in your web browse (for example abstracts 2006) press Ctrl F key and type name or keywords. You will be able to search TIE papers for entire year of 2006

2007 to 2008	titles	abstracts	Forthcoming Articles - Accepted papers
2000 to 2006	titles	abstracts	Forthcoming Articles - search friendly version
1988 to 1999	titles	abstracts	Forthcoming Articles - on IEEE XPLORE
all	titles	abstracts	The most recent issue - on IEEE XPLORE

2009	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6	issue7	issue8	issue9	issue10	issue11	issue12
2008	titles	abstracts	issue1	ssue2	issue3	issue4	issue5	issue6	issue7	issue8	issue9	issue10	issue11	issue12
2007	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2006	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2005	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2004	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2003	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2002	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2001	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						
2000	titles	abstracts	issue1	issue2	issue3	issue4	issue5	issue6						

FIGURE 65.1 A single link with all extracted information.



IEEE IEEE Transactions on Industrial Electronics

Volume 56, Number 5, May 2009 Access to the journal on IEEE XPLORE IE Transactions Home Page

56.5.1 T. Atsumi, "Feedforward Control Using Sampled-Data Polynomial for Track Seeking in Hard Disk Drives," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 5, pp. 1338-1346, May 2009. Abstract Link Full Text

Abstract: To decrease the seek time of hard disk drives, a feedforward control method was developed by using a sampled-data polynomial. The sampled-data polynomial satisfies the boundary conditions that include the characteristics of the zero-order hold, and compensates for the discretization error caused by the zero-order hold without the need for complicated calculations. Therefore, the feedforward control using the sampled-data polynomial enables real-time calculation and does not require lookup tables which need a large amount of memory. The parameters of the sampled-data polynomial are designed by using shock-response-spectrum analysis to minimize the settling vibrations caused by the feedforward control inputs. When the proposed method was applied on a hard disk drive, it significantly reduced the amount of tracking error in the seek control and also reduced the seek time.

56.5.2 M. F. Heertjes, X. G. P. Schuurbiers, H. Nijmeijer, "Performance-Improved Design of N-PID Controlled Motion Systems With Applications to Wafer Stages," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 5, pp. 1347-1355, May 2009. Abstract Link Full Text

Abstract: A nonlinear filter design is proposed to improve nanopositioning servo performances in high-speed (and generally linear) motion systems. The design offers a means to adapt fundamental control design tradeoffs—like disturbance suppression versus noise sensitivity—which are otherwise fixed. Typically performance-limiting oscillations in the feedback system that benefit from extra control are temporarily upscaled and subjected to nonlinear weighting. For sufficiently large amplitudes, this nonlinear filter operation induces extra controller gain. Oscillations that do not benefit from this extra control (typically because they represent noise contributions that should not be amplified) remain unscaled and, as such, do not induce extra controller gain. The combined usage of linear weighting filters with their exact inverses renders this part of the nonlinear filter design strictly performance based. The effective means to improve servo performance is demonstrated on a short-stroke wafer stage of an industrial wafer scanner. Since the nonlinear filter design is largely based on Lyapunov arguments, stability is guaranteed along the different design steps.

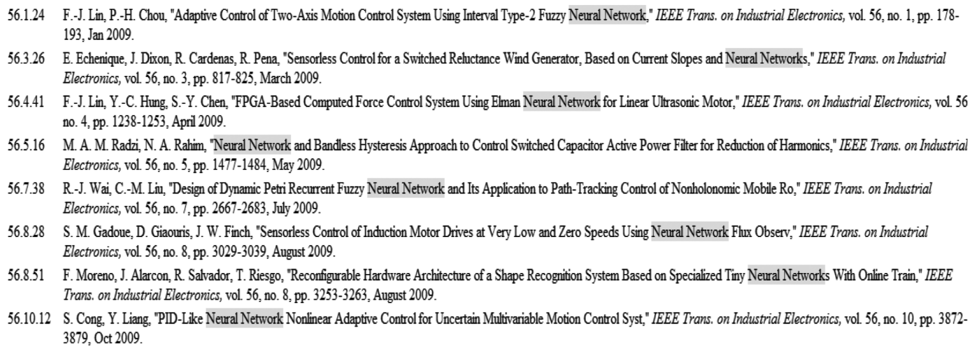
FIGURE 65.2 Web page with links generated by the IR.

or wget into it. The accuracy and speed of mining data make this method become special in processing and extracting data from the Internet.

For this particular application, the IR accesses XPLORE and downloads all the information about the papers stored in the server. Then it extracts all the desired information about authors, titles, page number, etc., and puts them together in an html format that is more human readable than the raw data. To be able to extract all information under different links, wget has to search and extract all the hyperlinks (Figures 65.1 and 65.2). AQ4

65.2.4 List Papers Only in a Specific Subject from TIE Webpage

In this example, the IR will browse the TIE web page and extract papers about "neural network." After downloading data from the server, the IR will save all information in an array, where each line of the web source will be an index of the array. By using regular expressions to modify the



56.1.24 F.-J. Lin, P.-H. Chou, "Adaptive Control of Two-Axis Motion Control System Using Interval Type-2 Fuzzy Neural Network," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 1, pp. 178-193, Jan 2009.

56.3.26 E. Echenique, J. Dixon, R. Cardenas, R. Pena, "Sensorless Control for a Switched Reluctance Wind Generator, Based on Current Slopes and Neural Networks," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 3, pp. 817-825, March 2009.

56.4.41 F.-J. Lin, Y.-C. Hung, S.-Y. Chen, "FPGA-Based Computed Force Control System Using Elman Neural Network for Linear Ultrasonic Motor," *IEEE Trans. on Industrial Electronics,* vol. 56 no. 4, pp. 1238-1253, April 2009.

56.5.16 M. A. M. Radzi, N. A. Rahim, "Neural Network and Bandless Hysteresis Approach to Control Switched Capacitor Active Power Filter for Reduction of Harmonics," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 5, pp. 1477-1484, May 2009.

56.7.38 R.-J. Wai, C.-M. Liu, "Design of Dynamic Petri Recurrent Fuzzy Neural Network and Its Application to Path-Tracking Control of Nonholonomic Mobile Ro," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 7, pp. 2667-2683, July 2009.

56.8.28 S. M. Gadoue, D. Giaouris, J. W. Finch, "Sensorless Control of Induction Motor Drives at Very Low and Zero Speeds Using Neural Network Flux Observ," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 8, pp. 3029-3039, August 2009.

56.8.51 F. Moreno, J. Alarcon, R. Salvador, T. Riesgo, "Reconfigurable Hardware Architecture of a Shape Recognition System Based on Specialized Tiny Neural Networks With Online Train," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 8, pp. 3253-3263, August 2009.

56.10.12 S. Cong, Y. Liang, "PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Syst," *IEEE Trans. on Industrial Electronics,* vol. 56, no. 10, pp. 3872-3879, Oct 2009.

FIGURE 65.3 Papers about neural network.

indexes containing the title and compare with the pattern "neural network," the IR will list all papers about "neural network" (Figure 65.3).

```
  $mydir="C:/ALL BACK UP/Hand Book/";
  $add="http://tie.ieee-ies.org/tie/abs/56s.htm";
  //copy website source into a local file
  $fname= "handbook.htm";
  system("wget.exe", "-q", "-O", $fname,$add);
  $file=$mydir."handbook.htm";
  open(r1,"<$file");
  @lines = <r1>;
  close(r1);
  $N=@lines;
  $j=0;
  // search array indexes containing 'neural networks'string

  for($i=0;$i<$N;++$i)
        {
           //transfer uppercase letters into lowercase letters
             $lines[$i] =~ tr/[A-Z]/[a-z]/;
           //search 'neural network'
             if(($lines[$i]=~ m/<td valign/)&&($lines[$i]=~ m/neural
               network/))
               { $a[$j]="<tr>".$lines[$i]."</td>"."</tr>";
                  $j=$j+1;}
        }
Result:
```

65.2.5 Using PERL with Google Scholar in Searching Data

Google Scholar is as easy to use as the normal Google Web search can be, especially with the helpfulness of the "advanced search" option, which can automatically narrow search results to a specific journal or article. The most relevant results for the searched keywords will be listed first, in order of the authors ranking, the amount of references that are linked to it and their relevance to other scholarly literature, and the ranking of the publication that the journal appears in and the citation index. The IR can be incorporated with this search engine to search for information about authors, citations, etc. With this

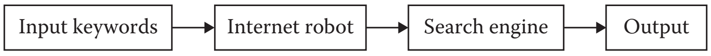


FIGURE 65.4 Searching process model.

```
Cites,Authors,
71,"J.M. Carrasco, L.G. Franquelo, J.T. Bialasiewicz,"
60,"F. Blaabjerg, R. Teodorescu, M. Liserre,"
48,"J. Holtz,"
38,"J. Moreno, M.E. Ortuzar, J.W. Dixon,"
35,"S. Katsura, Y. Matsumoto, K. Ohnishi,"
35,"P.P. Acarnley, J.F. Watson,"
33,"S. Alepuz, S. Busquets-Monge, J. Bordonau,"
32,"Y. Cheng, C. Qian, M.L. Crow,"
32,"J.H. Jung, J.J. Lee, B.H. Kwon,"|
```

FIGURE 65.5 Output file.

type of searching, the IR can take advantage of the search engine Google Scholar, which is relatively quick and easy to use. The searching process can be modeled as the following (Figure 65.4): input keywords from users, which can be journal name, year of published article, etc. When these keywords are defined, the IR will activate the search engine as Google Scholar, Web of Knowledge, etc., and search for selected information, then generate the output file. For example, if information about authors, citations of *Journal IEEE on Transactions Industrial Electronics* in the year 2006 is required, the users can define the keywords by using this journal name and the given year to activate the search engine Google Scholar. The IR will copy all information about papers on this journal in this year through Google Scholar. When extracting process is done, the text file output will be generated (Figure 65.5). With this type of concept, there are many applications where users can benefit from a Robot. The searching process is optimized and time-saving.

65.3 Summary and Conclusion

Current tools that enable data extraction or data mining are both expensive to maintain and complex to design and use due to several potholes such as difference in data formats, varying attributes, and typographical errors in input documents. One such tool is an Extractor or Wrapper, which can perform the data extraction and processing tasks [CKGS06]. Wrapper induction based on inductive machine learning is the leading technique available nowadays. The user is asked to label or mark the target items in a set of training pages or a list of data records in one page. The system then learns extraction rules from these training pages. Inductive learning poses a major problem—the initial set of labeled training pages may not be fully depictive of the templates of all other pages. Poor performance of learnt rules is experienced for pages that follow templates uncovered by the labeled pages. This problem can be solved by labeling more pages, because more pages cover more templates. Despite, manual labeling requires a AQ5
large supply of labor and is time consuming with an unsatisfied coverage of all possible templates.

This method of data extraction, DMR, optimizes data mining process and makes it become a popular tool in extracting data from web pages. PERL script with regular expressions and modules increases the speed of data extracting as well as the accuracy. DMR is customized according to the required data and the format of data that users desire.

References

[PW09] N. Pham and B.M. Wilamowski, IEEE article data extraction from Internet, *13th IEEE Intelligent Engineering Systems Conference* (*INES 2009*), Barbados, April 16–18, 2009.

[NGW08] S. Neeli, K. Govindasamy, B.M. Wilamowski, and A. Malinowski, Auto data mining from web servers using PERL script, *International Conference on Intelligent Engineering System 2008* (*INES 2008*), Miami, FL, pp 191–196, February 25–29, 2008.

[KCSG07] M. Keyed, C.-H. Chang, K, Shaalan, and M.R. Girgis, FiVa tech: Page-level data extraction from template pages, *Seventh IEEE International Conference on Data Mining Workshops 2007 (ICDM Workshops 2007)*, Omaha, NE, pp. 15–20, October 28–31, 2007.

[H99] S. Holzner, *PERL Black Book*, CoriolisOpen Press, Scottsdale, AZ, Edition 1999. AQ6

[WSC05] I.-C. Wu, J.-Y. Su, and L.-B. Chen, A web data extraction description language and its implementation, *29th Annual International Conference on Computer Software and Applications Conference 2005* (*COMPSAC 2005*), Vol. 2, Edinburgh, U.K., pp. 293–298, July 25–28, 2005.

[CKGS06] C.H. Chang, M. Kayed, R. Girgis, and K.F. Shaalan, A survey of web information extraction systems, *IEEE Transactions on Knowledge and Data Engineering*, 18(10), 1411–1428, October 2006.

AQ7 **TABLE 65.1** Excel File
TABLE 65.2 Result