Discovery: Preprocessing Technique in Web Server Data log

Supriyadi

STMIK Kharisma Karawang
Jl. Pangkal Perjuangan Km.1 ByPass Karawang 41316, Indonesia
Email:fnfcreator@stmik-kharisma.ac.id

Abstract

Preprocessing is a process to avail a clean and ready data for data mining analysis. Four stages are carried out in this process i.e. data selection, data cleaning, data integration and data transformation. The numerous semi structured pattern of web server log data accumulation has made the data cleaning-up to select the really needed data difficult. Hence, in this research a technique to carry out data cleaning using parser algorithm and query processing is proposed. The parser algorithm is written using PHP programming as web based programming, while the query processing is implemented using relationship database management system (RDBMS) MySQL. The computation system is tested using twenty six different sized trial data, originated from six web server. The final result concluded that in general the data being tested have 80 percent decreasing average with average processing velocity of 9.28 mbps.

Keywords: *query processing, parser algorithm, semi structured data, web log data*

1. Introduction

The preprocessing process is an important stage to invent new knowledge in the chain of Knowledge Discovery in Database (KDD). Basically the KDD is divided into seven (7) stages i.e. data selection, data cleaning, data integrating, transforming, pattern analysis and knowledge presenting [1]. Four out of the first seven stages were the preprocessing stage that required a lot of resources. The main objective of data preprocessing was to enhance the data quality that will be used in the data mining process such as text mining process and web mining [2]. In the web usage mining (WUM) preprocessing process obtaining important information accumulated in the data log access web usually was needed. The data was stored in the log_access file featuring the web server storage pattern. The stored data was still considered heterogen, although it had followed a specific pattern, this was due to the varied wares and the non-exsistence of specific restriction Various processes and algorithms were used to obtain quality data such as algorithym data cleaning manipulation, session identification and path completion [3]. In Fong et.al. [4] the preprocesing process was carried out using IPPM model by making the noise detector process, filthy data disposal, and the learning process took place in one line or frame. The numerous accumulated transaction in the web server had made the web log data a big data, so that data mining ware adaptation need algorithm redesigning and adding it into parallel domain like using Map Reduce [5][6][7].

Log web server access was a semi structured data; the data was arranged according to the web server pattern, in an independent text format for each data group. Unlike the data stored in a database, there was no validation in the stored data access log, such as the use of primary key or string restriction to make it uniform. The data can be considered as filthy data, because various clickstream transactions were recorded every time one line for each web page [8]. In some research on the data log cleaning process was carried out using algorithm manipulation or the available application [3][9][10][11]. However, the research had not presented detailed explanation on the technical implementation on using a specific programming language This research tried to carry out preprocessing by doing the data cleaning algorithm manipulation with structured data in form of array and fields output in the relationship database management system (RDBMS). The designed Algorithm was implemented using PHP Hypertext Preprocessor (PHP) script to control the query in RDBMS MySQL.

Received: 19 Sept 2017; Revised: 5 March 2018; Accepted: 15 March 2018; Published online: 3 May 2018

DOI:http://dx.doi.org/10.14203/j.inkom.574

2. Methodology

This research was carried out following the preprocessing general stages adapted from the four preceding stages of the KDD process i.e. selection data cleaning, data integration, transformation [1] as shown in figure 1

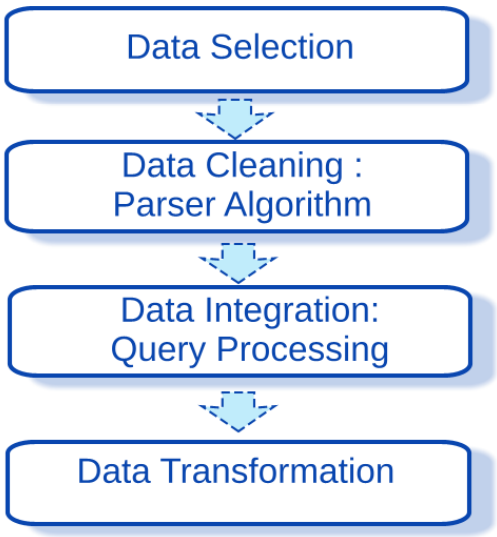

Figure 1. Research process stages

The data that will be processed was a relatively big data text that was formatted based on each web server. This research was using data originated from Apache web server that was developed by The Apache Software Foundation (ASF). The Data log was stored in three different files i.e. access-log, eror-log, and other-vhosts-log files. Of the three files mentioned, access-log file that had the following format will be utilized (ASF, 2016) :

%h %l %u %t %r %s %b %Referer %User-agent

Web server log data example:

```
192.168.3.74 - fnfcreator
[01/Dec/2016:01:26:42 +0700] "GET
/owncloud/cron.php HTTP/1.1" 200 2482
"http://prodi-ti/owncloud/" "Mozilla/5.0
(X11; Linux i686; rv:33.0) Gecko/20100101
Firefox/33.0"
```

From the access log_data compilation some important information could be obtained except for username logical data which in general usually has the sign "-", except for a very strict network. The following is the data string log group application to be observed:

The data selection process was usually adjusted to the research requirement; further the data must be cleaned.

Table 1. Data log web server text description

string	Description
%h	Recorded IP address of a remote host name
%l	Logic remote user originated from indented method
%u	User name had authentication
%t	Log time was recorded for every click in the web page
%r	Web page data requested by a client
%s	The status of the HTTP request or communication transaction provided by the server to a client%
%b	The size of the sent data
%Referer	Website reference data so that user enter the web server
%User-Agent	Client specification browser data that access the website page

For example at string [01/Dec/2016:01:26:42 +0700], the data required was the date data or only date and time, while the parentheses ([,]) and +0700 string was categorized as filthy data that must be cleaned or discarded. The expected clean data was 01/Dec/2016 or 01/Dec/2016:01:26:42. Each data line will be converted to array, to make the string search process easier. Usually the conversion process depended on the utilized separator such as quotation mark ("), space, colon (:) etc. In this research double quotation marks (") were used as each row separator, so that the log line data will be divided into five group i.e:

a. %h %l %u %t
b. %r
c. %s %b
d. {Referer} i
e. {User-agent} i

From the five groups there were three groups (a, b and c) that had relatively long data string, especially if the data form delivery method was using GET method in the web page For example in the following data log, group b (request) will be separated using empty space mark separator, then the important data was obtained:

```
GET /fnf/js/messages.php?lang=en&token=d743
d74353 HTTP/1.1 GET
/fnf/themes/img/status.png HTTP/1.1 GET
/fnf/root/log/login.php HTTP/1.1
```

Usually data cleaning depended on the research requirement. At the above example the universal resource locator (url) with the png extension was discarded because what was required was all url with php extension, it was possible that the php extension discarded if the url needed was url with pdf extension

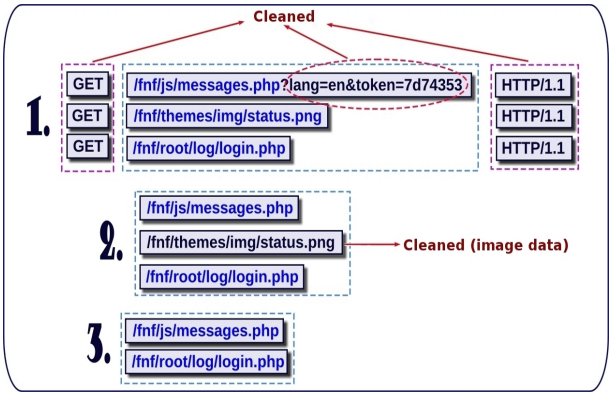

Figure 2. String request analyzing example

or others. After the data was cleaned, then the data was integrated (data integration) into a data base that was made using RDBMS MySQL. Usually the stored preceding data did not use duplicate data validation, it was more to clean and record the entire data without including the primary key. Furthermore to make the data easier to be analyzed and understood, the data stored in the data base should be tabulated (data transformation) that has interconnection based on the key. In this research, data cleaning process will be carried out using parser algorithm and query process to store in the database. The following is the proposed algorithm:

```
Data Selection
  1. Create temporary files (ftmp)
  2. Choose a acces_log file (myFile)
  3. f_read ← open_file ('myFile');
  4. content ← file_get_contens ('myFile');
  5. arr ← convert to array (f_read);

Data Cleaning
  if (arr[request.status] == 200) then
  if (arr[request.ext] == .php ||
      .html || .htm || .pdf || .odt || .doc)
  then
    t1 ← replace_str (" with '');
    t2 ← replace_str ([ with '');
    t3 ← replace_str (+0700 with '');
    content ← str(arr[(t1 & t2 & t3)]);
  ftmp = fopen(ftmp.csv);
  f_tmp ← writeTo(ftmp.csv, contents)


Data Integration
  for(i=0; i<arr.length; i++)
  f_tmp = fopen(ftmp.csv)
  open_db_connection();
    Query_Integ ← "insert into
integration_table set fileds = value"

Data Transformation
  open_db_connection();
  Qinduk ← query ("select * from
integration_table");
```

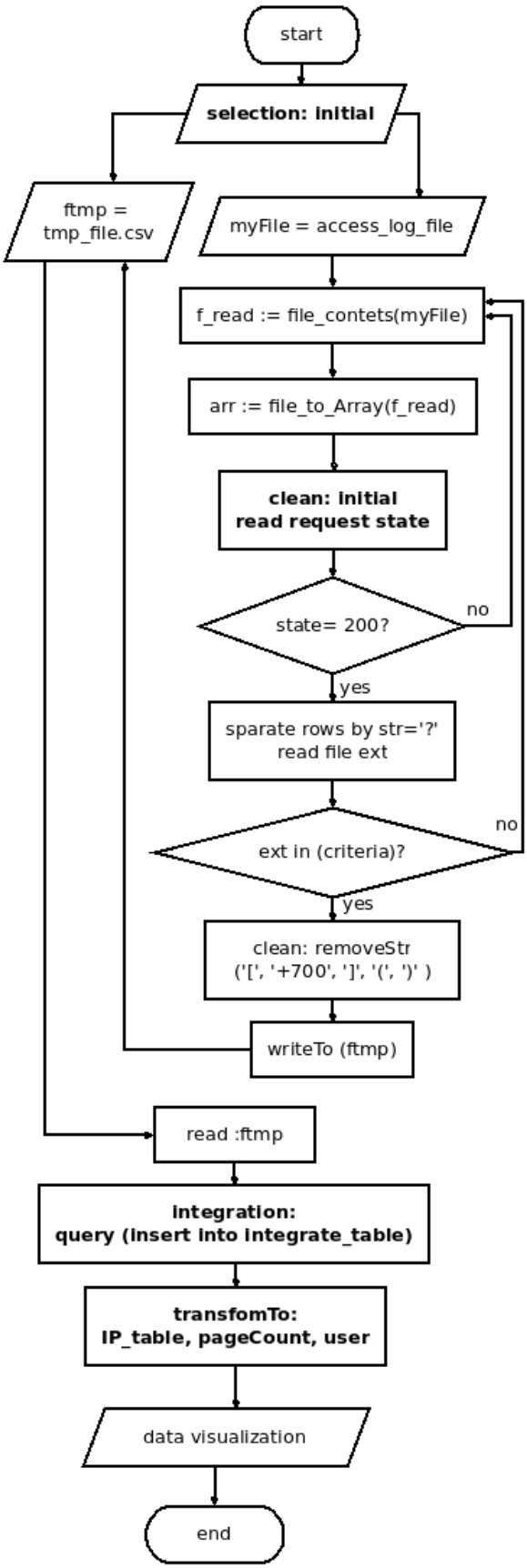

Figure 3. Flowchart proposed algorithm

```
     set IP_table = 'clustering by access
freq',
     page_Count = 'Query("select
count(IP)");
     user_behaviour = 'Query("select
count(IP) group by time_series")'
     ref = 'arr_brk2[3]' ")
```

3. Result and Analysis

3.1. Software Designing

The developed software is a web based application using computer system with the following specification:

Table 2. Computer System Specification

No	Element System	Specification
1	Operating System	Ubuntu 12.04 32-bit
2	Language Programming	PHP 5.6.3
3	Web Server	Apache 2.0
4	DBMS	MySQL
5	Memory	1.9 GiB
6	Proccessor	@2.33GHz 2

The application was designed to make the preprocessing process that consists of four stages as shown in figure 1 easier. The entire stages were integrated into one algorithm channel and was written PHP programming language. The rationale of using web based programming was for the application to be run in various operational system due to its multi-platform characteristic. The process channel flowed in one way and connected to each other so that the dependency relation (include) started from data selection to visualization 1 could be obtained in the use of case diagram.

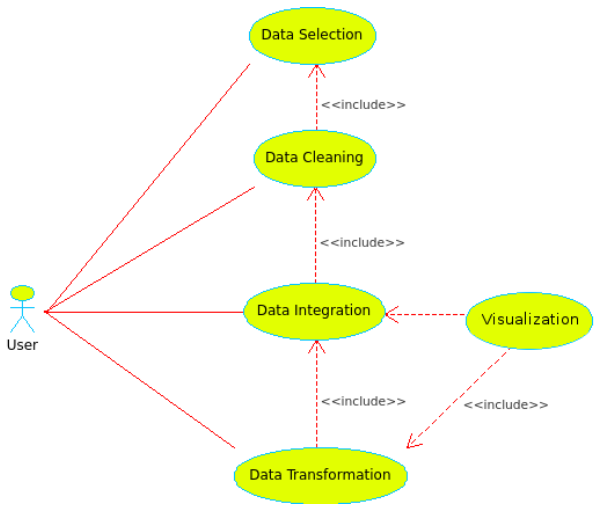

Figure 4. Use case diagram data preprocessing

3.2. Data Selection

To have more varied data and accuracy on the proposed algorithm test outcome, the log data access was taken from some web server. Three groups of data were obtained from the local computer web server in the Local area Network (LAN) and two groups from the active internet network web server. To have the calculation process more precise, the utilized data size was byte. The data had the size between 1 1.000.000.000 byte, with the interval distribution as followed (Tabel 3):

Table 3. To be tested data size grouping

Num	Size interval (byte)	amount
1	data $<$ 10.000	2
2	10.000 $<$ data $<$ 100.000	5
3	100.000 $<$ data $<$ 1.000.000	5
4	1.000.000 $<$ data $<$ 10.000.000	5
5	10.000.000 $<$ data $<$ 100.000.000	5
6	data $>$ 100.000.000	4

3.3. Data Cleaning

All the transaction process carried out by the client that was successful, half success or failed would be recorded by the web server. The cleaning stage would be as followed:

1. Choose a row that has 200 statuses that is a successful transaction (selection A)
2. Choose log row that has extended string request according to the need, e.g. php, pdf, or any other (selection B).
3. Discard all the strings behind the sign ?, existing at the request string (cleaning 1)
4. Convert the character [with an empty space (cleaning 2)
5. Convert the character +0700] with an empty space (cleaning 3

In general the stage was an inseparable process with the data cleaning process because it was in one group order or in algorithm it was written as followed:

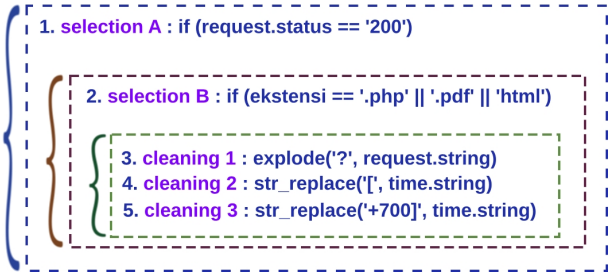

Figure 5. Data cleaning process illustration

The graphic in figure 5, showed some conditions where the depreciating data size value was high i.e. D, H, K, and O data. Similar to the high value of the data depreciating value, the process velocity was also above the average

```php
<?php
$tmpFile = fopen("tmp.csv", "w") or die("failed to open!");

$myFile = $_FILES['txtberkas']['tmp_name'];
$contentfile = file_get_contents($myFile);

$arr = explode("\n",$contentfile);
$jbrs = 0;
for($k=0; $k<count($arr); $k++)
{ $arr_rows = explode('"', $arr[$k]);
  $arr_rows_untuk_content = explode('"', $arr[$k]);
  if(substr($arr_rows[2],0,4) == ' 200') //if state 200
  {$arr_rows_1 = explode(" ",$arr_rows[1]);
    $arr_rows_1_1 = explode("?",$arr_rows_1[1]);

    $txt_req =  strlen($arr_rows_1_1[0])-3;
    $fileExt = substr($arr_rows_1_1[0],$tiga_teks_req,3);
    if(($fileExt == 'php')
        or ($fileExt == 'htm')
        or ($fileExt == 'pdf')
        or ($fileExt == 'tml')
        or ($fileExt == 'xls'))
    {
    for($m=0; $m<=count($arr_rows)-2; $m++)
    {$no = $m+1;
     if($m==0)
     {
        $arr_rows[$m] = str_replace("[","",$arr_rows[$m]);
        $arr_rows[$m] = str_replace(" +0700]","",$arr_rows[$m]);
     }
     if($m==1){
        $arr_rows_1 = explode(" ",$arr_rows[$m]);
        $arr_rows_1_1 = explode("?",$arr_rows_1[1]); }
        if($m==2)
        {$arr_rows_2 = explode(" ",$arr_rows[$m]);
         $arr_rows[$m] = " ".$arr_rows_2[1];
        }
        if($m==3)
        {$arr_rows_3 = explode("?",$arr_rows[$m]);
         $arr_rows[$m] = $arr_rows_3[0];
         $arr_rows_untuk_content[$m] =" ". $arr_rows_3[0];
        }
       }
      if($m==5)
      { $arr_rows_5   = explode(" ",$arr_rows[$m]);
        $arr_rows_5_1 = explode(";",$arr_rows_5[1]);
        $arr_rows_5_1_g = str_replace("(","", $arr_rows_5_1[0]);
       fwrite($myFile, $content);
      } }
    }} ?>
```

Figure 6. Data cleaning code in PHP scripting

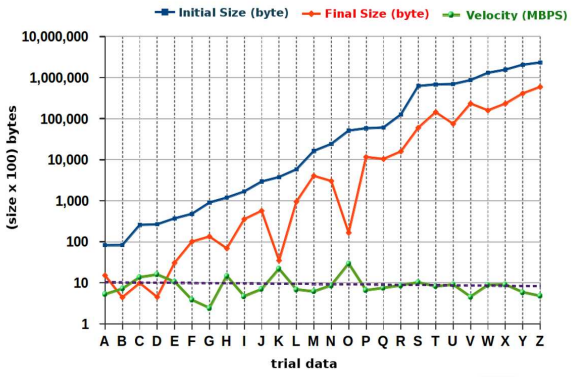

Figure 7. Data cleaning process trial result graphic

line (trend line) This was due to the preceding separation that for data log with the request status except 200 and the existence except .php., .html, .htm, .pdf, .odt, .odp, .ods, .doc, .xls, .docx, .xlsx and .pptx will be discarded as they were considered filthy data. In other words, there was a failure in transaction process and data communication between client and server, or server often redirect to present data image or page style repeatedly in D,H,K, and O data. In general the data cleaning process had the average data test depreciating at 86% with average process velocity at 9.28 MBPS. Looking at the graphic, the data cleaning process velocity was relatively stable

Table 4. Data test

Code	File name	Size (byte)
A	access.log.3-1-simak	8,211
B	access.log-1-simak	8,277
C	stmik-kharisma_log	25,698
D	pmb.stmik-kharisma._log	26,540
E	access.log.2-prodi-ti	36,913
F	access.log.1-prodi-ti	47,694
G	access.log.4-simak	89,609
H	access.log.3-akademik	118,923
I	wiki.kharisma._log	167,225
J	kuesioner.kharisma_log	292,163
K	pmb.kharisma-Feb-2017	375,619
L	access_log_uji-simak	578,376
M	warungkopi.kharisma_log	1,627,722
N	access_log_uji-simak	2,410,679
O	stmik-kharisma_Feb-2017	5,097,320
P	kuesioner.kharisma_log	5,799,623
Q	access.log.4-simak	6,059,311
R	access_log_uji1-deptan	12,580,531
S	access.log2.-deptan	62,306,216
T	access.log1.-deptan	68,113,110
U	access_log-deptan	69,416,017
V	access.log4.-deptan	86,543,821
W	access.log3.-deptan	130,360,103
X	access.log.4-deptan	155,365,873
Y	access.log.2-deptan	205,330,526
Z	access.log.4-deptan	232,221,052

however the data trend line velocity course tended to decrease in line with the increase of trialed data size. Data binning process needed to be carried out to discard noisy, to facilitate the observation on the data velocity relation to depreciating data. The process was carried out based on data velocity by observing the minimum and maximum data. Looking into the data velocity process, minimum data of 2.4 mbps and the maximum of 29.5 mbps were obtained or an extended value between 0 until 30. If i=5 interval is taken, thus the interval total (j) is:

j = (max - min) / i
= (30 - 0) / 5
= 6 interval

Based on figure 8, the velocity mean always directly proportional with data depreciating mean. The higher the depreciating the higher the velocity was, and so likewise. This was due the numerous filthy data, and then more data was skipped and not read, so that the process became faster.

3.4. Data Integration

After the data were sorted out and each part was cleaned adjusted to the need, then the data was reintegrated by rewriting it in a new file as followed in figure 9.

After the data were sorted out and each part was cleaned adjusted to the need, then the data was reintegrated by rewriting it in a new file. Every time the reading process and the new trial bundle were cleaned,

Table 5. Data test result

Code	Final size (byte)	Velocity (MBPS)	Reduct- ion	Time (second)
A	1,511	5.2	82	0.0015
B	440	7.2	95	0.0011
C	982	13.6	96	0.0018
D	448	16.0	98	0.0016
E	3,046	10.7	92	0.0033
F	10,050	3.8	79	0.0119
G	13,367	2.4	85	0.0352
H	6,867	14.7	94	0.0077
I	35,266	4.7	79	0.0339
J	56,539	7.0	81	0.0400
K	3,482	21.8	99	0.0164
L	94,205	6.9	84	0.0804
M	402,093	6.1	75	0.2535
N	300,835	8.6	88	0.2663
O	16,541	29.5	99	0.1650
P	1,157,915	6.6	80	0.8436
Q	1,035,439	7.5	83	0.7714
R	1,581,069	8.7	87	1.3864
S	5,982,336	10.2	90	5.8207
T	14,362,727	8.2	79	7.9410
U	7,429,705	9.1	89	7.3112
V	23,243,546	4.6	73	18.0437
W	15,848,228	8.8	88	14.1295
X	23,136,799	9.1	85	16.3501
Y	41,037,419	5.9	80	33.2438
Z	59,155,859	4.8	75	46.3805

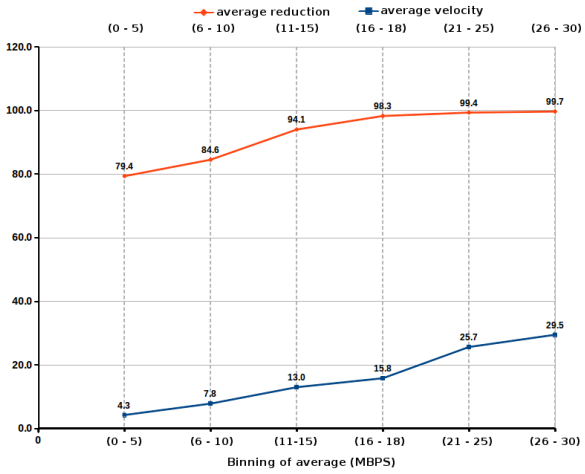

Figure 8. Data binning process result graphic

then the new_file.csv, the new data will be presented, so that the new file will be affected by the newest data. To produce a data ready to be analyzed with the technique in the data mining, it is suggested to continue to proses data transformation process.

Actually integration process is a process to make data processing easier, because the data was stored in a clean file. The data was still in the form of text with more structured and orderly compilation. To have the data legible using the structured query language (SQL) the

Table 6. Trial data sequencing based on velocity

Code	velocity (MBPS)	Reduct- ion
G	2.4	85.1
F	3.8	78.9
V	4.6	73.1
I	4.7	78.9
Z	4.8	74.5
A	5.2	85.1
Y	5.9	80.0
M	6.1	75.3
P	6.6	80.0
L	6.9	83.7
J	7.0	80.6
B	7.2	94.7
Q	7.5	82.9
T	8.2	78.9
N	8.6	87.5
R	8.7	87.4
W	8.8	87.8
U	9.1	89.3
X	9.1	85.1
S	10.2	90.4
E	10.7	91.7
C	13.6	96.2
H	14.7	94.2
D	16.0	98.3
K	21.8	99.1
O	29.5	99.7

Table 7. Binning proccess result

Binning (velocity)	Average of velocity	Average of Reduction	Amount of data
0 - 5	4.3	79.4	6
6 - 10	7.8	84.6	14
11 - 15	13.0	94.1	3
16 - 20	15.8	98.3	1
21 - 25	25.7	99.4	1
26 - 30	29.5	99.7	1

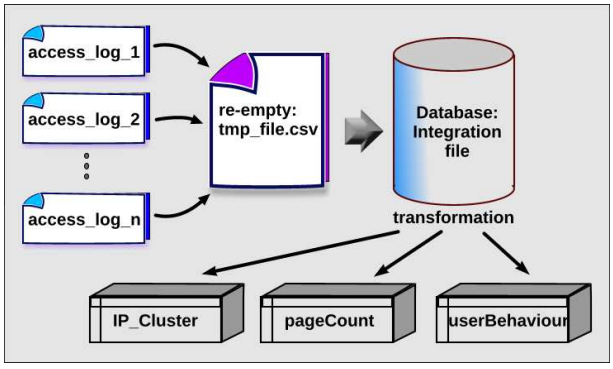

Figure 9. Data integration process

data text should be reformed into database format.

Figure 10. Example of Trial data presentation before being cleaned and integrated

Figure 11. Example of Trial data presentation after being integrated into a new file

3.5. Data Transformation

Data transformation was a conversion in the form of general data text like comma separated value (csv) to become data text in the database and table. The conversion was carried out to make the data analysis process easier in accordance to the following needs:

a. Website users clustering and mapping using IP address data.
b. Traffic analysis based on the users access time (time-series)
c. Users conduct analysis based on the accessed file (url) and the data reference
d. Users clustering and browser classification
e. Web page access statistic

Although it was converted into table, the preceding table collector or the integration table had not been given the primary key. The objective of eliminating the primary key was to have all data can enter although with a same IP address. It was suggested to use primary key or primary index based on IP address and time series, for further table like the IP_cluster table.

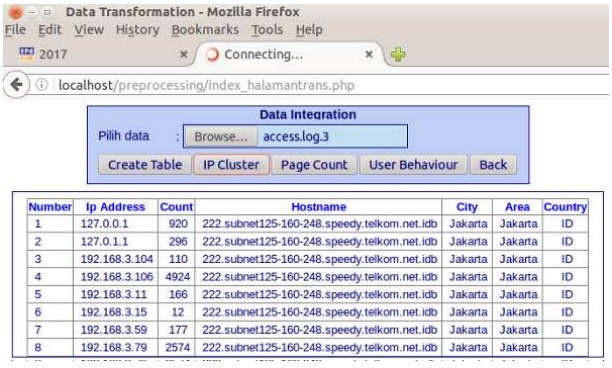

Figure 12. Clustering IP Result

4. Conclusion

After carrying out the preprocessing process (data selection, data cleaning, data integration and data transformation) it can be concluded that data cleaning was the stage that needed the most resource. This was due to the numerous filthy data that had to be cleaned with around 80% from the trial data. New software system design was meant for data log with ASF standard format. While for data log format outside the standard format it still needed adjsutment, moreover for data log besides Apache web server.

5. Ackowledgement

The writer convey her heartfelt gratitude to the STMIK Kharisma Karawang (2017) and the Department of Agriculture (2012) who had given the web server data log to be tested and analysed.

Daftar Pustaka

[1] D. Tomar and S. Agarwal, "A survey on pre-processing and post-processing techniques in data mining," *IJDTA :International Journal of Database Theory and Application*, 2014.
[2] K. K. Pandey and N. Pradhan, "An analytical and comparative study of various data preprocessing method in data mining," *IJETAE : The International Journal of Emerging Technology and Advanced Engineering*, 2014.
[3] B. Maheswari and P.Sumathi, "An effective method to preprocess the data in web usage mining," *ARPN Journal of Science and Technology*, 2013.
[4] S. Fong, R. P. Biuk-Aghai, Y. whar Si, and B. W. Yap, "A lightweight data preprocessing strategy with fast contradiction analysis for incremental classifier learning," *Hindawi Publishing Corporation Mathematical Problems in Engineering*, 2015.
[5] M. Agung and A. I. Kistijantoro, "High performance cdr processing with mapreduce," *Journal ICT Research and Application*, 2016.
[6] D. Peralta, S. del Ro, S. Ramrez-Gallego, I. Triguero, J. M. Benitez, and F. Herrera, "Evolutionary feature selection for big data classification: A mapreduce approach," *Hindawi Publishing Corporation Mathematical Problems in Engineering*, 2015.
[7] B. J. G. S. H. F. Triguero I., Peralta D., "Mrpr: A mapreduce solution for prototype reduction in big data classification," *neurocomputing*, 2015.
[8] D. C.E, "An application for clickstream analysis," *IJCC: International Journal of Computers and Communications*, 2012.
[9] A. R. Anand S., "An efficient algorithm for data cleaning of log file using file extensions," *IJCA: International Journal of Computer Applications*, 2012.
[10] N. D. Grace L.K.J., Maheswari V., "Analysis of web logs and web user in web mining," *IJNSA:*

International Journal of Network Security and Its Applications, 2011.

[11] R. K. Makwana C.H., "An efficient technique for web log preprocessing using microsoft excel," *IJCA: International Journal of Computer Applications*, 2014.