



SCHOOL OF
COMPUTER SCIENCE

Information
Management Group

HCW— Web Evolution Technical Report 2, September 2008

Web Evolution: Code and Experimental Guide

Alex Q. Chen and Simon Harper

Human Centred Web Lab
School of Computer Science
University of Manchester
UK

The Web Evolution study [2] investigates the issues affecting the web user interface through understanding the evolution of the web. A Web robot [1] was used to capture data for the study. This report provides instructions for using the code, and for customising the code for future work.

HCW

Human Centred Web

Web Evolution

The Web Evolution project is investigating into the issues that cause the lag between the introduction of standards and recommendations and when they were adopted by the developers. We will focus on the relationship between these issues, especially those involving the web user interface. This project will identify trends, and provide us with graphs demonstrating how the web has been evolving over the past ten years. By understanding these evolutionary trends we can inform and predict web development in the future. <http://hcw.cs.manchester.ac.uk/research/>.

Web Evolution Reports

This report is in the series of HCW Web Evolution technical reports. Other reports in this series may be found in our data repository, at <http://hcw-eprints.cs.man.ac.uk/view/subjects/web-evolution.html>. Reports from other Human Centred Web projects are also available at <http://hcw-eprints.cs.manchester.ac.uk/>.

Contents

1	Introduction	1
1.1	Synopsis	1
2	Data Formats	1
3	Process Cycle	3
3.1	Selecting Websites	3
3.2	Capturing Data	4
3.3	Analysing the Data	4
3.4	Web Robot Configurations	5
4	How to Use	5
4.1	Extracting Alexa Top 20	6
4.2	Extracting Alexa Global Top 500	6
4.3	Extracting Random 500 Websites	6
4.4	Extracting Random 5000 Websites	7
4.5	Extracting Archives	7
4.6	Downloading the Contents and Related External Files	7
4.7	Analysis	8

Human Centred Web Lab
School of Computer Science
University of Manchester
Kilburn Building
Oxford Road
Manchester
M13 9PL
UK

Corresponding author:
Alex Qiang Chen
tel: +44 (161) 275 7821
chenqa@cs.man.ac.uk
<http://homepages.cs.manchester.ac.uk/~chenqa>

1 Introduction

The results and data provided in the standalone technical report [2] can be re-generated by running the code. This report discusses the basic requirements of the computer system required to use the code, and how the code and data should be used so that the results can be re-generated.

Knowledge of PHP Hypertext Preprocessor (PHP) and SQLite was assumed when writing this report.

This report also covers how these code can be modified for different usage and analysis. Thus further analysis to update these results can be continued with some customisation to suit future needs.

1.1 Synopsis

This report was structured as follows:

Section 2: Data Formats explains the type of file formats and the data structure of the files used in this study.

Section 3: Process Cycle describes the flow and steps for the code, and the reason for each step.

Section 4: How to Use run through the steps to be taken to re-run the study, as well as how the captured data and results were stored.

2 Data Formats

A number of data format were downloaded or extracted from the web in our study. Some of these data were results from our analysis and others were the actual data downloaded from the web for further analysis, and reproducibility. Our code was written in PHP and the downloaded data had an ASCII text file extension. Table 1 list the different data and file formats used in our study.

Data type	Type	Data format	File format
HTML file	Downloaded	Plain text	.aaf
External JavaScript files	Downloaded	Plain text	.js
External CSS files	Downloaded	Plain text	.css
Selected URLs	Extracted	Relational database	SQLite
Results from analysis	Results	Relational database	SQLite

Table 1: Orientation to the Materials

Except for the HTML files, our downloaded data were stored using the same file extension as the original downloaded file. This was because the original extension of the HTML file may come in numerous different file extensions such as ‘.htm’, ‘.html’, ‘.asp’, and ‘.php’. Therefore a standard ‘.aaf’ extension was introduced in this case. Each data file was given a unique file name as illustrated in figure 1 to

avoid over writing any existing data files with the same file name and extension. The file name of the data file consist of three parts; the time stamp, the URL, and the extension (Ext). The time stamp is a combination of the date and time in the format of YYYYMMDDhhmmss when file was stored, and the URL is the actual data captured's URL with some of the string patterns replaced as seen in table 2. Finally the extension of a data file was either assigned, reassigned or a mirror from the original file's extension as shown in table 3.

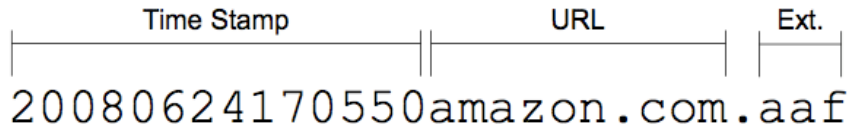


Figure 1: Structure of captured file name

String pattern	Replacement
http://	'blank'
http://www.	'blank'
http://web.archive.org/web	'blank'
'space'	'blank'
: , ; / ? %	'blank'
+ =	-
&	N

Table 2: Captured data files URL characters replacement

Actual file extension	Assigned file extension
Original webpage	.aaf
External JavaScript file	used given (e.g. .js)
External CSS file	used given (e.g. .css)

Table 3: Assigned file extension for captured files

For each website, after downloading is completed, the record in the relational database will be flagged and the related file name used to store the downloaded data is recorded. For each website's data file, the URL for the related external files will be updated with a local file path each time these external files are downloaded. These are usually CSS and JavaScript files.

The code, downloaded data and relational databases were organised in different folders designated according to their purpose. How to use the code, and how the downloaded data were stored are covered in greater depth in the next section.

3 Process Cycle

The web robot consisted of a number of components to control, process and store the captured and analysed data. All of our code and data were stored in the different sub-directories according to their designated purpose. The main directory “db” was used to house all the captured data and the database. All configuration files used by the web robots were stored in the “settings” directory. This includes database file names, the names of the database tables, type of graphical formats for analysis, as well as the detection of elements within a HTML document to identify the type of HTML standard the document was written in. Customisation of a particular process or analysis can be done by changing the variables in the configuration files.

3.1 Selecting Websites

Two sets of top websites were selected from the Alexa’s global top 500 list¹; one set for the top twenty websites from the list with archives available at Internet Archive¹ between 1999 and 2000. The second set uses that full list of URLs for capturing the snap shot of the current web. These two sets of data provided a historical and current view of the top websites for analysing trends. Another two sets of data were collected randomly to provide a historical and current view of the web for analysing trends, and to identify if the top websites do give a good representation of the web in general.

The selected URLs were stored in the “maindb” database, under its respective data set’s table. In each of these tables, besides the URLs, the status, and actual storage file names with the path and file extension were stored. For each URL, if it is a HTML file with all the required external JavaScript and CSS files, the status is indicated as ‘Extracted’. When an ‘error’ status is flashed, this means that an error has occurred during the capturing process. To identify the cause of the error, refer to the “ErrorLog” table. For the full list of status flags and their respective meanings during the capturing process, please refer to table 4.

Status	Capturing state
Error	An error occurred during the capturing process.
Extracted	After URL and related external files were successfully captured.
New	After selection of URL

Table 4: URL status and capturing state

The top twenty websites were manually selected together with the top five hundred websites taken from Alexa’s global top 500 list. When selecting the random websites, for the random five hundred websites the “selectingRandomSites.php” from the “web_archives” folder was used to ensure that the websites selected had at least one archive between 1999 and 2000. A similar script (“detectingRandomSites.php”) was used in the same folder when selecting the random five thousand websites for the current web analysis.

¹<http://www.archive.org>

Finally once the websites were selected for all four sets, the archives URL must be captured so that the websites archives data can be captured too. The script “GetArchivesOfDomain.php” in the “web_archives” folder was employed for this task.

3.2 Capturing Data

The HTML page and its external JavaScript and CSS files were successfully captured, and stored in the respective data set sub-directory folder under the “db” folder. This caused the URL status to update from ‘New’ to ‘Extracted’. The four sub-directory folders include ‘alexaTop20_webpages’, ‘alexaTop500_webpages’, ‘Random500_webpages’, and ‘Random5000_webpages’.

Besides the HTML page, all the related external files’ extensions were reused when capturing them. However, the HTML page when captured is reassigned with an “.aaf” extension during storage.

The PHP files that were used to carry out these tasks are found in the “tasks” folder. The order of the files and their purposes are presented in table 5. To extract the HTML code and its related external files (JavaScript and CSS), the order in which these two files were run is important. A feature to automatically call the required script was programmed. Thus, once the first script (webpage_extraction.php), has completed its task, it will automatically call the “external_file_extraction.php” script. Upon successfully executing these two scripts, a loop back to the first script is automatically called. This process continues until no additional files are required.

Order	Code File	Purpose of this code
1	webpage_extraction.php	To extract and store the HTML code into a “.aaf” file. The URLs to be captured is based on the list selected in the previous stage (Sub section 3.1, Selecting Websites)
2	external_file_extraction.php	Code to extract the related external files. For JavaScript and CSS external files.

Table 5: Capturing the HTML code and its related external files.

After capturing the required data based on the list produced in the website selection process (see sub section 3.1), manual verification of these data is required to ensure the integrity of the captured data. Once this is done, analysis of this information can be conducted to identify trends.

3.3 Analysing the Data

The main analysis file named “SourceAnalysis_1.2.php” can be found in the “analysis” folder. This file analyses the HTML standards used by the webpage, the type of graphical formats used by the webpage, the type of client-side scripting languages, and platforms AJAX detection and accessibility detection. The output of this analysis is printed to the screen and stored in the “source_analysis” table

from the “maindb” database after each time the code is run. The table is formatted into four columns: ‘PERIOD’, ‘URLType’, ‘AnalysisType’, and ‘Results’. The ‘PERIOD’ column is the time period of the websites that are analysed. The ‘URLType’ indicates the data set of the website (Top20, Random500, Random5000, Top500). The type of analysis for the results is indicated in the ‘AnalysisType’, and the ‘Results’ column stores the number of websites that conforms to the standards or recommendation of the analysis.

3.4 Web Robot Configurations

Our web robot can be configured to analyse different graphical formats and different HTML standards by changing some of the parameters in the settings files. Three files in the ‘settings’ folder were used to configure the web robot so that it can be customised, and updated to the latest standards and recommendations.

The analysis indicates that HTML standards, and graphical formats evolve the quickest. Thus two configuration files were used to allow the user of the web robot to configure the different types of HTML standards and graphical formats required by the analysis. These lists can be expanded, changed or reduced. The file ‘html_versions.dat’ contains the configurations for the analysis of different HTML standards. This file is divided into two main parts where each part consists of two columns. The first part initialises the different HTML standards by using the first column to store the different type of HTML standards and the second column to store the respective date of when the HTML standards became a W3C recommendation. The first column of the second part lists the unique elements corresponding to the respective dates of the HTML standards when they became a W3C recommendations. The other file, ‘graphic_types.dat’ lists the different graphical formats and their possible file extensions. The first column lists the file’s extension that corresponds to the respective graphical format in the second column. With these configuration files analysis can be customised to suit specific needs.

The basic settings of the web robot(s) can be configured using the ‘settings.php’ file. This is a PHP class file that allows the web robot to do its work normally or in debug mode for close examination. It also allows the user to specify the database and tables within the database where the results will be stored.

4 How to Use

The code used in this study was all written in PHP, thus no compiling and installation to the code is required on the client-side. However a user-agent is required to request for this service from the server. On the server-side besides the web server, the following plug-ins are essential to run the code.

- PHP 5
- SQLite 3

Once the web server is properly set up, the next thing to do is to copy our code to the root folder of the web server public directory. To run or test the code, a web

browser is required on the client-side and on the server-side the web server has to be running. In the web browser, to run the code, call the root directory of the web server followed by the code path from the “WebRobot_1.1” directory.

The scripts should be ran in a certain order. This is because some of the code cannot be executed without the results from a previous script. Hence this section discusses the order of steps required to run the code successfully. The first step in the overall process flow cycle [2] selects the websites for the four data sets.

4.1 Extracting Alexa Top 20

The top twenty websites were selected manually from the Alexa global top 500 list². These were the top twenty websites from the list with at least one archive available from Internet Archives³ between the year 1999 and 2000. Thus these websites’ URLs were manually input into the database for downloading the content and the archives in the latter stage of the overall process cycle.

4.2 Extracting Alexa Global Top 500

Our top five hundred websites were extracted directly from the Alexa global top 500 list. The script that does this can be called via a web browser via the following path.

```
WebRobot_1.1/tasks/alexa_top500_extraction.php
```

This code extracts the top five hundred websites from the most recently available Alexa global top 500 list each time it is run.

4.3 Extracting Random 500 Websites

To extract the random 500 websites from the Google Directory, the following script was used. Notice that the URL with the ‘no’ parameter set to 500 was to cause the program to stop at five hundred websites. This parameter was hard coded, and the program will recursively extract up to eight websites from each subdirectory (or) until the parameter ‘no’ value is met.

```
WebRobot_1.1/web_archives/detectingRandomSites.php?no=500
```

This code randomly selects the websites from the Google Directory⁴ that have at least one archive available from the Internet Archive⁵<http://www.archive.org> between the year 1999 and 2000.

²http://www.alexa.com/site/ds/top_sites?ts_mode=global&lang=none

³<http://www.archive.org>

⁴<http://www.google.com/dirhp>

4.4 Extracting Random 5000 Websites

To extract the random 5000 websites from Google Directory, the following URL was used.

```
WebRobot_1.1/standards/random_sites_helper.php
```

This code randomly selects the websites from the Google Directory³. It was hard coded to extract five thousand websites in total where in each subdirectory sixty websites are selected.

4.5 Extracting Archives

Two of the four data sets required the downloading of the archives of the websites for the past ten years. The following code was created to extract the URLs of the websites from Internet Archive for content download later. To call this code the following URL is called

```
WebRobot_1.1/web\_archives/Auto\_GetArchivesOfDomain.php
```

However when using this code, some of the variable names have to be manually changed when used for different sets of data. Refer to table 6 for the different variable names for each respective data set.

4.6 Downloading the Contents and Related External Files

After the websites were selected and entered into the database, the contents were downloaded and the related external files were processed. At this stage it includes downloading the archives material when necessary. To download the content the following URL is called.

```
WebRobot_1.1/tasks/webpage_extraction.php
```

When using this code, some of the variable names have to be manually changed when used for different sets of data. Below is the list of variable names for the different sets of data (see table 6). In the code, replace any of the following with the data set required for content extraction.

Variable name	Data sets
\$ms->maindb_top20_table	Alexa top 20 websites
\$ms->maindb_alexatop500_table	Alexa top 500 websites
\$ms->maindb_random500_table	Random 500 websites
\$ms->maindb_random5000_table	Random 5000 websites

Table 6: Variable names and data sets

Next the “\$process” variable also needs to be changed for the automatic process to be completed correctly. The value of this variable has to be changed to the respective data set as shown in table 7.

Variable name	Data sets
Top20	Alexa top 20 websites
alexatop500	Alexa top 500 websites
Random500	Random 500 websites
Random5000	Random 5000 websites

Table 7: Variable values (case sensitive) and data sets

Once this code has completed its task for each website, it should automatically call the following URL to download the related external JavaScript and CSS files.

`WebRobot_1.1/tasks/external_file_extraction.php?t=alexatop500`

For the above example, the parameter ‘t’ is set to ‘alexatop500’ indicating that it is presently extracting the Alexa top 500 websites list. After this code has completed its task, it should automatically loop back to the “webpage_extraction.php” code until all the websites in the database for that data set have been successfully download.

4.7 Analysis

A number of analysis processes were required for this study. This code was compiled in a main analysis file, which can be called using the following URL when using a web browser:

`WebRobot_1.1/analysis/SourceAnalysis_1.2.php?t=Top20&d=200507`

For this code two parameters were used: ‘t’ and ‘d’. In this case, parameter ‘t’ indicates to the code the type of data set it needs to analysis such as Top20 and Random500. The parameter ‘d’ indicates the start time period of the data set it needs to analyse, e.g. 200507 means July 2005. Change these parameters as required for different analysis.

After computing the results, the code will print the final results of all the analysis to the screen, as well as storing them in the table called “Source_Analysis” in the relational database. This will allow further analysis of the results in the future.

References

- [1] A. Q. Chen. Codes and downloaded data. http://hcw-eprints.cs.man.ac.uk/75/2/WebRobot_1.1.zip, September 2008.
- [2] A. Q. Chen. Web evolution: Method and materials. Technical report, The University of Manchester, September 2008.