EXTENDING JMETER TO ALLOW FOR WEB STRUCTURE MINING

Agustín Sabater, Carlos Guerrero, Isaac Lera, Carlos Juiz

Computer Science Department, University of the Balearic Islands, SPAIN
pinyeiro@gmail.com, carlos.guerrero@uib.es, isaac.lera@uib.es, cjuiz@uib.es

ABSTRACT

This paper describes a JMeter extension created to allow for web structure mining process. This type of mining process is very important, for example, in the field of web performance engineering and web development validations. The extension allows users to define the HTML tags that wrap the content units and it creates a graph model to represent the content units of all the web pages and the relationships among them (aggregations). The usability of the extension has been validated in a real scenario.

KEYWORDS

Web data mining, Web structure mining, JMeter, Web engineering

1. INTRODUCTION

Web mining is usually applied in research works in the field of web performance, content noise reduction, content validation, content analysis, user behaviour model and prediction... Web structure mining is the part of web mining focused on extract knowledge and data about the structure of the web pages and their content. This can be applied to several levels, for example structure of the links of the web pages, structure of the content... In this paper, we present a tool to allow JMeter for this type of process: web structure mining.

We have created a tool to obtain a model of the content units of a web page and the way they form a complete web page, i.e., the aggregations of the contents and the content units to be assembled to create a web page. The tool, using a set of steps, downloads the content of a list of web pages, analyzes the HTML of these web pages to detect the content units and, finally, creates the graph of the content units and relationships.

We have selected JMeter as starting point of the development to avoid the implementation of some steps of the web structure mining process. JMeter allows us to create extensions to cover all our requirements. Finally, we have validated the use of the tool by mining the structure of a real web site: *Reddit.*

The paper is organized as follows: Section 2 describes the details of web structure mining, the related work and the motivation to develop our tool; Section 3 contains the details of the development of the tool; Section 4 explains the case of study to validate the usability of the tool; finally, Section 5 concludes the paper and explains the future work.

2. WEB STRUCTURE MINING DEFINITION AND PROCESS

Web mining is defined as the use of data mining techniques in the field of web applications. The objective of the mining process can vary significantly and it is usually split in three different types of web mining: *Web usage mining,* related with the extraction of information about how

the users interact with the web system; *Web structure mining*, related with the analysis of the internal (parts) and external (links) structure of the web pages and the web systems; *Web content mining*, related with the extraction of knowledge from unstructured data.

This paper presents the creation of a tool to automatically extract the internal structure of the web pages on a web site. This type of information is very useful for several research works for example, research related with performance of the systems [1,2], validation of web content [3,4] or noise removal from web content [5].

Currently web applications (web 2.0) have two main differences with the first dynamic web applications (web 1.0). These differences are that users can customize their own web pages and that web pages are usually created from the aggregation of contents of other users or other web systems. Thus, the content of the web pages is not a single element, i.e., web pages are created with smaller content units. The web page is created with the assembly of these content units. Different users request different web pages because they select different content units and the same content unit can be displayed in different web pages.

There are many examples that illustrate the use of these smaller content units. For example, social networks show different content in the web pages for different users depending on their friendships and their setups. When a user creates a piece of content, e.g. a comment in his wall, all the web pages of his friends will include that comment. This comment corresponds to a content unit and it will be included in several web pages and these web pages will be created from the assembly of this content unit with other content units, e.g. comments from others users.

Therefore, it is necessary to consider content units, web pages and inclusion relationships to model the web pages. In this type of model, the web pages are created by the aggregation of content units, and the content units can be aggregated in several web pages. This conceptual description maps perfectly with a graph definition. The nodes of the graph are the content units and the edges are the aggregation/inclusion relationships. Figure 1 shows an example of a web page and its corresponding graph representation.

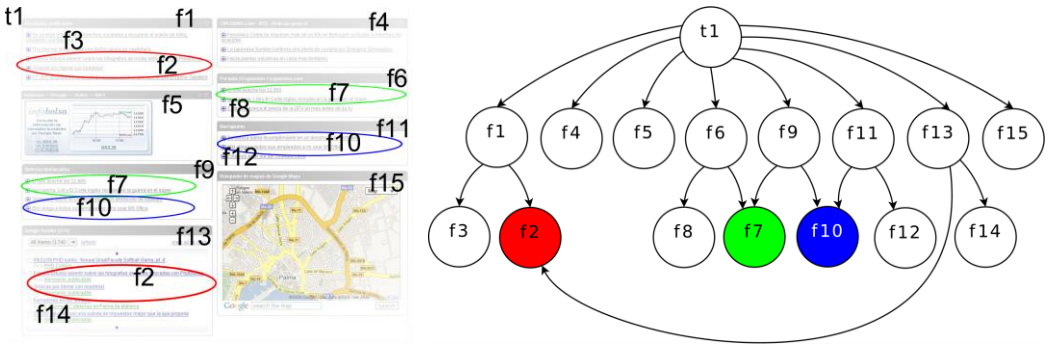

Figure 1. Example of web page modelling using a directed acyclic graph.

The graph is a directed acyclic graph (DAG). The direction of an aggregation relationship is important, so the edges are directed. Moreover, a piece of content cannot be self-included, so the graph is acyclic.

The web pages are implicitly represented in the graph. The web pages are related to the root nodes of the graph (nodes without incoming edges) and the complete web page, all its content units, is obtained from the connected subgraph starting in the root element.

Figure 2 represents the model of a web site with three web pages. The nodes *CE1*, *CE2* and *CE3* correspond to the three root elements of the web pages. If we consider the connected subgraphs

using these root elements as starting points, we obtain the content units of the three web pages: *webPage1 = { CE1, CE5, CE6, CE7, CE8, CE9 }* ; *webPage2 = { CE2, CE4, CE5, CE6, CE7, CE8, CE9 }* ; *webPage3 = { CE3, CE4 }*.

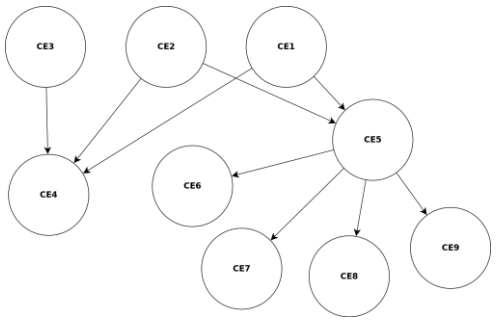

Figure 2. Example of model for a web site.

The use of this type of representation for the model of content units of web pages was suggested in [6] and they called it as Object Dependence Graph (ODG). Several research studies have used this type of models to represent the web pages or to solve their problematic [7].

The types of analysis done in a web structure mining process differ among them (web pages hyperlink graph, DOM model of the document…) [8, 9]. The number of tools to automatically analyze the structure of web pages is quite high but, from our knowledge, any of them are focused to extract these ODG or models for content units. The work presented in this paper explains the development of a tool to obtain this model from a set of web pages in a given site.

The process of obtaining the models is divided in several phases. The first one is to define the set of pages of the web site indicating the URLs of each of them. The second phase concerns to the download of the content of the web pages. In the following phase, the content units (graph nodes) are identified by the analysis of the HTML of the web pages. Finally, the edges of the graph are obtained. The definition of the intra-page edges (edges of content units of the same web page) is trivial. But the edges to content units of other web pages (inter-page edges) need also to be detected.

3. DEVELOPMENT OF THE SOLUTION

This section is dedicated to explain the design decisions about the development of the proposed tool. The tool should cover all the phases explained in the previous section, and it should be customizable for the different web sites to mine their structures.

There are several tools that cover the phase of downloading the content from a web site. It is logical to start our tool from this point. We considered to choose one of the existing tools and to create and extension or plug-in to cover the phases not implemented in it. The details of the development of this extension are explained in the following sections.

3.1. Tool Selection

We defined a list of criteria to select the tool to be used as the base of the development: (a) Free source code or API for the development of extension or plug-ins to allow us to change or add functionality; (b) Coverage of some of the phases of the structure mining process to avoid the development of some of the requirements of the tool and to reduce the work to be done; (c) Availability and valence of documentation of the tool to reduce the learning process for the development and usability of the tool; (d) Availability of an active internet community to solve doubts and get feedback of other developers.

From our knowledge, there are three candidate tools to be used as starting point for our development: JCrawler, JMeter and Solex. JCrawler is an open-source multiplatform tool created in Java. The execution of the tool is based on the command line and its setup is done using XML files. The tool requests a list of URLs to a server and it supports HTTP forwarding and cookies.

Solex is an open-source plug-in for Eclipse that allows to record and repeat user sessions, stress tests and performance test of web sites. Solex acts as an HTTP proxy and records all HTTP requests and responses. The task of replaying a scenario consists in sending the previously recorded and eventually customized HTTP requests to the server and asserting each response.

Finally, JMeter is an open-source tool created by the Apache foundation. This tool is very flexible and covers a wide range of tasks of web test and stress tests. Among the three tools, this is the one with more features. For example, it covers user login, HTTPS, AJAX requests… The user community is also important. There are a high number of programmers and users and there is more documentation than for the other two tools. Moreover, there are some plug-ins that cover some of the phases of the web structure mining, e.g. the download of the content of a web page. It also provides a better way to implement the concurrency of the users in a system by the use of multithreading. We considered JMeter as the best tool to start with. Table 1 shows the data about the three tools in a summarized way.

Table 1. Tools comparison.

Requirement	**JCrawler**	**Solex**	**JMeter**
Free source / extendability	Yes	Yes	Yes
Already implemented our requirements	None	None	Partially
Documentation	Partially	Very poor	Good
User community	Very poor	Very poor	Good

JMeter allows us to use its already implemented plug-ins and to extend them to cover all our requirements. Thus, the programming tasks were reduced. We also had a wide user community to ask about doubts and problems with the development. The programming language for the extensions of JMeter is Java.

3.2. Requirements Implementation

The development of the tool to cover all the tasks of the web structure mining process was done in two different extensions or plug-ins. We called *HTML downloader* and *Query Model Analyzer* to both extensions. The tasks of the process are divided between both extensions.

The main reason to do this is to provide more general plug-ins. Thus, if any user needs only to download the content of a list of web pages can use the first of the plug-ins. The inclusion of modules in a test plan of JMeter is very easy. The guidelines of development of JMeter extensions also indicate that it is better to deploy smaller and simpler modules.

The first one covers the definition of the site and the web pages to be analyzed and the download process of the web pages. The standard components of JMeter allow us to repeat this process periodically as many times as required.

To use the first extension it has to be included in a test plan. The web pages to be downloaded and the local folder to store the downloaded content have to be provided in the setup process. Figure 3 shows the JMeter setup window of this extension.

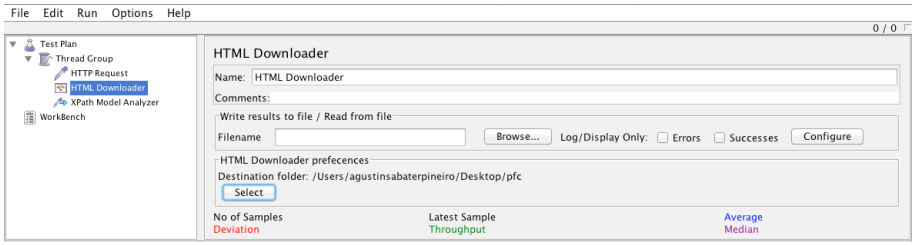

Figure 3. JMeter window for the *HTML downloader* extension.

The creation of this extension was done by the use of the sampler elements of JMeter. A sample element allows to the user to request a web file. By the inclusion of post-processors, some task can be done after the sampler is executed. We defined a task to store locally the content requested from the server.

The second module was designed to analyze a previously stored HTML content. The result of the analysis is the delimitation of the content units and the creation of a graph to represent them and their inclusion relationships. This plug-in should be used in conjunction with the previous one. Figure 4 shows the setup window of JMeter for this extension.

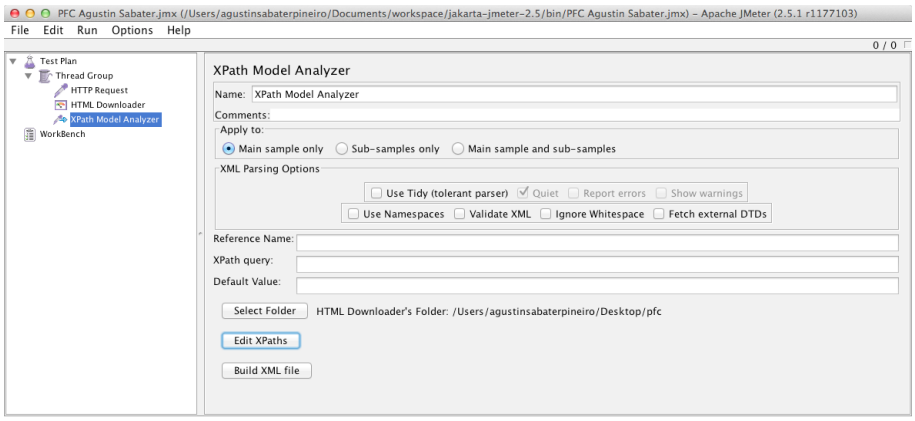

Figure 4. JMeter window for the *Query Model Analyzer* extension.

The main part of the module is the algorithm that detects the content units of the HTML document. We implemented a first version in which users have to assist the detection algorithm. This detection is done analyzing the HTML tags of the document. Users define XPath patterns to detect the HTML tags that indicate the starting and ending point of a content unit.

In a first step, the extension parses the HTML code to check the mapping of the XPath patterns with the document tags. When a matching is triggered, the content unit is added to a list and a new node in the graph model is included and connected with an edge to the parent content unit to indicate the aggregation relationship. Previously to this inclusion, the plug-in walks all the content units in the list in order to check if this content unit is also included in other web page and it has been detected previously. In this case, just the aggregation relationship is added to the graph model. The extension allows detecting content units inside the already detected content units. Figure 5 shows an example of three XPath patterns and their matching with an example of HTML code.

The created model of content units and aggregation relationships is represented using an ODG and it is stored in a XML format. The XML uses GraphML that is a standardized format to represent graphs.

Therefore, the setup of the second extension needs to indicate the source folder with the downloaded HTML web pages and the XPath patterns. The development of the plug-in keeps open the use of other alternative algorithms for the detection of the content units, e.g. algorithms as the one presented in [3] or [4]. Other developers can include new Java classes to implement new content unit detection algorithms in order to allow to the users to select ones.

The plug-ins and their source-code are free and available in [10]. This development has been presented as a bachelor dissertation in the Balearic Islands University [11].

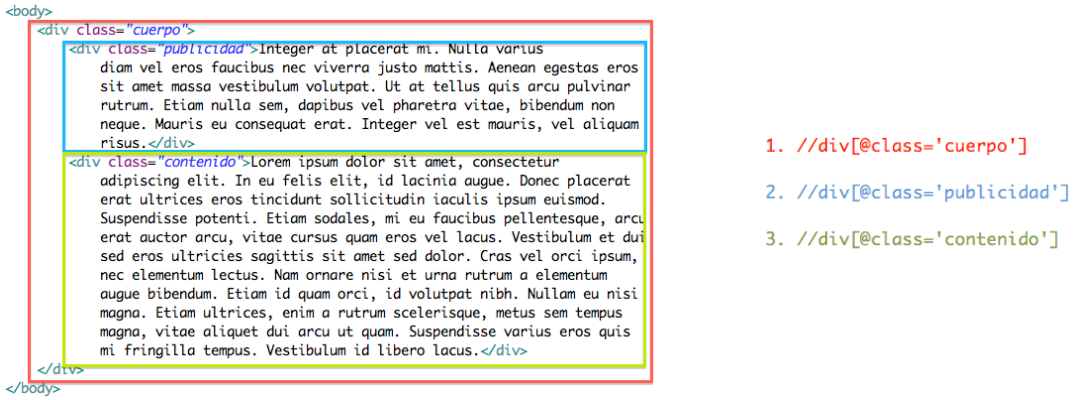

Figure 5. Example of XPath patterns matched with content units of a HTML web page.

4. CASE OF STUDY

In this section, we are explaining the use of the proposed tool in a practical case. Thus, we want to validate the usability and the correctness of the tool. We have selected a well-known web site and we have used our tool to extract the structure model (ODG) of their web pages.

We selected *Reddit* as the web site to be analyzed. This is a web site where users include news and they vote the most interesting one. The news articles are organized in a home page and in sections. Our objective is to detect each of the content units corresponding to a news article (headline and the lead paragraph). Figure 6 includes a screen shot of the main page of *Reddit* with some of the news links highlighted.

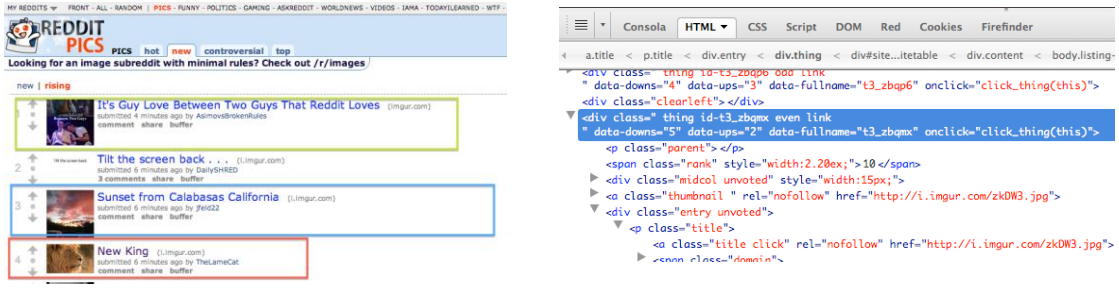

Figure 6. Content units highlited in *Reddit* and their HTML code.

The setup of our extension of JMeter involves creating the list of URLs to be downloaded and the XPath patterns that we have defined to detect the HTML tags that wrap the content units. We have included 10 web pages: the home page and 9 category sections as pics, funny, politics, gaming… The URL of this websites looks like *http://reddit.com/r/[sectionName]*. The second point of the configuration of the extensions is to determine the XPaths. After the analysis of the HTML we determined that the content units (news) were wrapped in a *div* tag with class *thing*.

Figure 6 includes an example of the HTML code for a news link. Therefore the XPath to detect this content units was defined as *//div[contains(concat(‘’,’normalize-space(@class,’’),’thing’)]*.

The configuration of the test plan of JMeter was done including a *Thread Group* with just 1 thread and a *Loop Controller* with 2 iterations to download the contents of the web pages in two different time shots. Finally, the *HTTP Request* controller for each of the web pages is included and the *HTML downloader* extension to store the content of the web pages and the *Query Model Analyzer* to analyze the code, detect the content units and create the ODG model. Figure 7 includes the JMeter test plan configuration for an example of two web pages instead of 10.

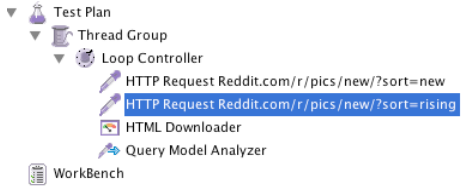

Figure 7. Example of the correct JMeter test plan configuration using our extensions.

After the execution of the test plan, we obtained an XML file with the ODG model. The content units detected were around 500. The execution of the model creation (*Query Model Analyzer*) took 24.8 seconds in a Macbook Air with an Intel I5 1.7 GHz processor and 4GB RAM. The download time of the pages has a strong correlation with the network connection. In our case, the *HTML downloader* took 14.4 seconds using the wired network of the university campus.

We manually analyzed the results in order to detect if all the content units were extracted and if the ones repeated in different sections of the web site were detected and not included twice. This analysis showed us that the functionality of the tool was correct and the repetition of content units in the model was avoided and all of them were correctly detected. Therefore we validated the use of the tool with this case of study.

5. CONCLUSIONS

We have presented a tool to assist in the web structure mining process. By the creation of extensions, we have allowed JMeter for the possibility of request and store a set of web pages and analyze the structure of the HTML documents in order to find content units. These content units are detected by the analysis of the tags that wrap the content. The user sets up the detection process by indicating the XPath patterns that match those tags.

The result of the web structure mining process is an ODG model that represents the content units as nodes and the aggregation/inclusion relationships as edges. The tool also detects the presence of the same content elements in different web pages to avoid the repetition of the node in the ODG model. The resulting models can be used in other research works in order to improve the performance of the system, to validate the content structure of a web, etc. We have validated the usability of the tool by mining the structure of a real web site.

The tool keeps open the possibility to other programmers to implement a different content unit detection algorithm. The tool is public available and the license allows to other users to modify the source code.

As a future, the analysis of the evolution of the ODG models should be included. Performance studies not only need to have a model of the content structure in a given point in time, they also need to know the evolution of the contents [12]. By the analysis of the ODG models of different points in time, a higher model with the evolution of the content could be represented by the

matching of the content units among the different ODG graphs. This matching process should consider not only the identical content units but also similar content units because the content is dynamic and it is changing continuously. By this, a new type of edges would be included in the model, the similarity edges.

Another important future work is to make a deeper analysis of the performance execution of our JMeter extension. In this study we have measure the execution times, but further studies could be done. The scalability of the system for very large content sites should be also studied.

ACKNOWLEDGEMENTS

This work is partly financed by the Spanish Ministry of Education and Science through the TIN11-23889 project.

REFERENCES

[1] Carlos Guerrero, Isaac Lera and Carlos Juiz. (2013). "Performance Improvement of Web Caching in Web 2.0 via Knowledge Discorvery". *Journal of Systems and Software*. DOI: 10.1016/j.jss.2013.04.060. ISSN: 0164-1212.

[2] Carlos Guerrero, Carlos Juiz and Ramon Puigjaner. (2011). "Evaluation of a Fragment-Optimized Content Aggregation Web system". *Fourth International Conference on Internet Technologies and Applications (ITA 2011).* Wrexham, Wales. UK.

[3] Lakshmish Ramaswamy, Arun Iyengar, Ling Liu, and Fred Douglis. (2004). "Automatic detection of fragments in dynamically generated web pages". *In Proceedings of the 13th international conference on World Wide Web (WWW '04)*. ACM, New York, NY, USA, 443-454.

[4] Arvind Arasu and Hector Garcia-Molina. (2003). "Extracting structured data from Web pages". *In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03)*. ACM, New York, NY, USA, 337-348.

[5] Sivakumar, P.; Parvathi, R. M. S (2011). "An Efficient Approach of Noise Removal from Web Page for Effectual Web Content Mining". *European Journal of Scientific Research;* 3/ 1/2011, Vol. 50 Issue 3, p340.

[6] Jim Challenger, Paul Dantzig, Arun Iyengar, and Karen Witting. (2005). "A fragment-based approach for efficiently creating dynamic web content". *ACM Trans. Internet Technol.* 5, 2 (May 2005), 359-389.

[7] Carlos Guerrero, Carlos Juiz, and Ramon Puigjaner. (2011). "Improving Web Cache Performance via Adaptive Content Fragmentation Design". *In Proceedings of the 2011 IEEE 10th International Symposium on Network Computing and Applications (NCA '11)*.

[8] Shi, W.; Collins, E.; Karamcheti, V. (2003). "Modeling object characteristics of dynamic Web content". *Journal of Parallel and Distributed Computing;* Vol. 63 Issue 10, pp. 963-980.

[9] Ricardo Baeza-Yates and Paolo Boldi. (2010). "Web Structure Mining". *Advanced Techniques in Web Intelligence, Studies in Computational Intelligence.* Vol. 311 pp. 113-142.

[10] Agustin Sabater, Carlos Guerrero. (2012). "Plugins for data mining with JMeter.". *GitHub repository*. https://github.com/asabater/Data-mining-with-JMeter

[11] Agustin Sabater. (2012). " JMeter Adaptation for web structure". *Bachellor dissertation*. Balearic Island University. Spain.

[12] Dennis Fetterly, Mark Manasse, Marc Najork and Janet Wiener. (2003). "A large-scale study of the evolution of web pages". *In Proceedings of the 12th international conference on World Wide Web (WWW '03)*. ACM, New York, NY, USA, 669-678.