Squidler: A Proxy Log Parser with Automatic Web Page Classifier

[1]**BARON GERARLD S. MANZANO,** [2]**Ludwig Johann B. Tirazona,** [3]**Joseph Anthony C. Hermocilla**
Institute of Computer Science, UPLB
Information Technology Center, UPLB
[1]bgsmanzano@gmail.com,
[2]ljtirazona@gmail.com,
[3]jachermocilla@gmail.com

Internet connectivity is an expensive but very important resource especially in academic institutions. Its non-academic use deprives others of the bandwidth needed to conduct research and other academic-related tasks. Thus, there is a need to determine and classify the web pages visited by constituents in order to properly manage access to this costly resource. Squidler is a web application that combines the functionality of a web proxy log parser and a web page classifier. It automatically retrieves the access log from a proxy server then outputs a web page that displays information about outgoing HTTP requests from clients. It then downloads the web pages from the URLs in the log, simulating browser requests, and then performs the classification. Our method uses Support Vector Machines (SVM) to automatically classify web pages either as academic or leisure. We tested the application using the proxy server deployed in the University of the Philippine Los Banos. Results showed that Squidler was able to classify most of the academic pages but failed on some leisure pages. Also, some performance issues were encountered in relation to the amount of data that can be processed simultaneously.

1. Introduction

The Internet has become the main source of information for our everyday lives. For businesses, communications, and academics, a lot of information is stored and retrieved in the Internet. People gain access to the Internet by connecting to private connections or public servers using a Web browser. These servers are hosted by organizations that provide services and manage resources for the benefit of all. These servers, or proxy servers, act as gateways between a private network and the Internet. When a computer, or client, connected to the server tries to access a website, the request is sent to the proxy server, and the proxy server sends the request to the actual server hosting the website. After receiving the request, the server sends the information needed back to the proxy server, and then back to the client who requested it. The proxy server also saves a cache, or a copy, of the data requested, for faster access later [1]. Proxy servers can also keep track of cache refreshing, garbage collection for old cache files, as well as selecting specific documents to be cached [2]. The Information Technology Center (ITC) in UPLB uses a proxy server as well. It uses Squid, a widely used proxy server for Linux and UNIX platforms[1]. They also use Squint, a Squid proxy log parser that checks their network traffic, and it is viewed in HTML. They can view who are the ones connected to the server, and for how long they are connected. It also displays the amount of bandwidth and resources they have consumed. It can also display the different websites visited.

The resources provided by proxy servers are shared to everybody accessing the server, so when a client connected to the server requests for websites with high bandwidth and the resources cannot keep up, the connection slows down and everyone in the server is affected as well. Extreme cases of resource hogging could make the server reject all service requests, which would make matters worse for the clients and the administrators. Prohibiting all high-bandwidth sites is a solution, but it may not be much helpful when the sites in question are needed for academic purposes. Not knowing what kinds of sites to allow or forbid can lead to problems later on.

A useful extension to log parsers is a web page classifier, an application that sorts web pages according to different categories. The administrator can sort the different websites that the clients visit, as well as pinpoint the categories that consume the most bandwidth. He can potentially control what kinds of websites should be allowed or forbidden. This paper focuses on integrating and demonstrating these two applications together: a squid log parser and a web page classifier into a web-based application. The output of the application would display information about the clients connected to the server, as well as the websites visited and the categories they fall under.

The main objective of this work is to combine the functionalities of a Squid log parser and a web page classifier in one web-based application. The specific objectives are:

1. to create a web-based squid log parser that displays information of the IP addresses and their connections;
2. to gather data from the websites visited by the IP addresses and classify them according to subjects (business, leisure, etc.); and
3. to test the final application in a working environment.

2. Review of Related Literature

Research and development of proxy servers is still ongoing. These proxy servers usually have logs stored in them that contain information. It is up to a separate log parser application to make use of this information.

Squint is a Squid log analyzer that is still under development as of 2011. It parses the log files in the Squid server and produces static output presented in an HTML form. From there, the user can view different information regarding the proxy server connections. There has been a lot of research for web page classification over the past few years, because of its importance in information retrieval and increasing complexity. It helps optimize search results and manage Web directories. Though recently, sorting content has been quite a hard task because of the different forms of information present in the Internet [3].

J. Ulrich from the University of British Columbia has developed an automatic classification system and compared different web page classifiers. He tested two different classification algorithms: LogitBoost and Positive Example Based Learning (PEBL). LogitBoost is a multi-class classifier. It is a boosting algorithm that involves having a set of base classifiers, and based on these classifiers, weigh the correctly classified points and the misclassified ones. PEBL is a single-class classifier and is generally easier to implement than multi-class ones because of the existence of negative examples. It can sort websites into “academic” and “non-academic”, among other things. He also developed a web-based page classifier that implements these two algorithms for comparison [4].

A recent journal article published in 2011 by W. Ali, S. Shamsuddin and A. Ismail entitled “Web Proxy Cache Content Classification based on Support Vector Machine”. The study involves classifying the web objects stored in the web proxy for optimization of web caching. Support Vector Machine (SVM) is an algorithm that is much more preferred over other algorithms when it comes to text and web page classification [5].

Similar problems involving web classification has also been tackled at the Institute of Computer Science in UPLB. In 2006, T. Monserrat conducted a study that describes the structure of a website using Information Visualization. He used a crawler in the study to gather information and links in the website, and used Quasi-Hierarchical graphs to give structure to the website [6]. In 2009, R. Dilla developed an online job information retrieval system entitled “JobSpider”. Given a job title and location, it will search for websites relevant to the users inputs, and save important information in a database. The web crawler was written in Java and implements a breadth-first search algorithm. Breadth-first search was more appropriate because “it accumulates a list of all the links that are on the current page before it follows any links to a new page” [7].

3. Theoretical Framework

3.1 *Proxy Log Parsing*

Proxy log parsing, or analyzing, creates reports of Internet connection usage based on the log files created by proxy servers. These logs contain information about the IP addresses that connected to that server, including information about their Internet connection. Proxy servers are useful for organizations who access the Internet, as well as monitoring the Internet usage through the server logs [8].

3.2 *Web page Classification*

Web page classification, according to Qi and Davison, is “the process of assigning a web page to one or more predefined category labels”. The problem with web page classification is the training of the classifier to recognize these data. There are different types of classification, like subject classification, functional classification, and many others. Subject classification deals with the topic of the web page. It sorts the web page by “business”, “games”, or other categories. Functional classification sorts the web page by its purpose on the Web. It determines if a page is a “home page” or “login page”. There are a lot of options with classification, including categorizing a web page in two or more categories, among a large set of categories [3].

3.3 *Support Vector Machines*

Support Vector Machines (SVM) is a learning algorithm that is widely used for classification problems. It takes in a set of inputs and plots it in a two-dimensional space. From there, each input is weighted based on its importance and determines the likelihood of these inputs belonging to a class. SVM can be implemented as a binary classifier that determines if the inputs belong to ”this class” or ”not this class”[9].

The idea of SVM, as used in log file processing, is to divide the positive and negative samples while achieving a minimum error rate. Typical use of SVM involves three steps; data collection, data preprocessing, and the training phase. In the data collection step, information is retrieved from the web proxy log files. Data preprocessing involves arranging the data in preparation for the next step. It involves parsing the log files, filtering unwanted information, and formatting. In the training phase, SVM that utilizes the radial basis function kernel arranges the formatted data [5].

4. Methodology

The overall architecture of Squidler is shown in Figure 1. PHP [10] was used as the scripting language and MySQL [11] as the database backend.

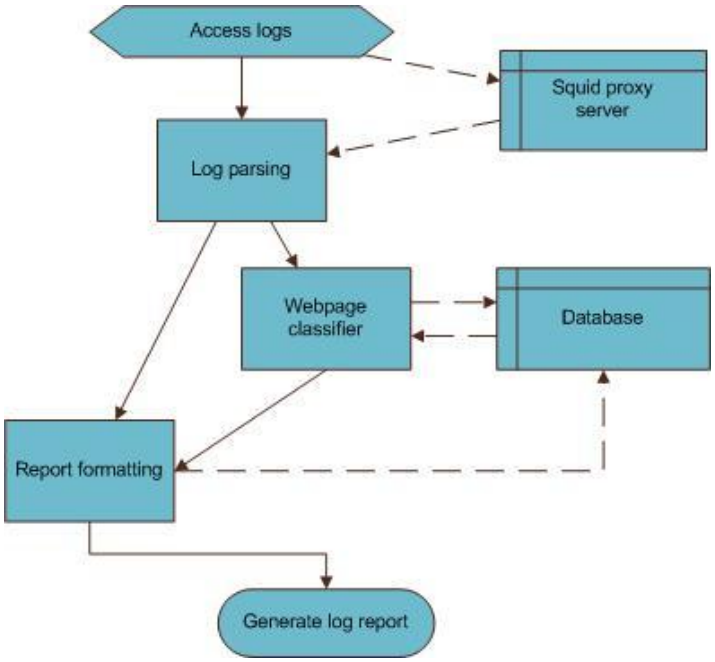

Figure 1. Overview of Squidler's processes.

4.1 *Squid log parser*

The log parser was developed in PHP. It is only a single class named SQParser composed of two main methods or functions:

- readLog(): This is the function called at the start of the application. It gets the access.log from the Squid proxy server in through SSH (Secure Shell) command. If it fails, it retrieves a backup copy of the access.log in the local data instead. Once it has gathered the data, it tokenizes the data line by line and stores it in an array. The information gathered from readLog() is passed to parseLine() afterwards.
- parseLine(): This function receives data from the array created by readLog() and parses it to a human-readable language. It tokenizes the given string and saves it in an array. Using the tokenized data and the given integer, it checks the database if that information has already been stored. If there is no record, it saves that data. Afterwards, it returns the data array.

4.2 *Web page classifier*

The web page classifier was developed in PHP to integrate the log parser later on. It is composed of the SQCrawler and SQClassifier classes that handle the web page and calls the different functions needed. The web page classifier used the Support Vector Machine (SVM) algorithm using a separate PHP library(PHP SVMLib) by Ian Barber [9]. A separate PHP file needs to be run first to initialize the model the SVM works upon.

SQCrawler: This is the main class. A single URL is passed to this class, and it checks if the URL passed is valid or not by browsing the content of the webpage. After confirming its validity, it then checks the database if that same URL has already been categorized. If the database contains the required data, it returns the category of that URL and terminates the class. Otherwise it calls on the SQClassifier class, that returns an array of words that are greater than a specific length (5 characters). These words are compared to a list of words retrieved from a file, and values are assigned to these words. Afterwards, these values are passed to the SQClassifier class that returns the chosen category. The URL and the category will be stored in the database, and the category will be returned, terminating the class.

SQClassifier: This class is responsible for classifying the web page given the array of strings. It opens a file containing keywords and values, and compares these words to the array. If there is a match, the value corresponding to the keyword is given to the string, otherwise it randomizes a value. Afterwards, it puts these values in a file and calls on the SVM function.

4.3 *Database*

The database contains one table, logdata. This database is used for the storing some parts of the log file, as well as the web page classification.

Table 1. Structure of the logdata table.

Fields	Description	MySQL Data Type
timestamp	timestamp of the log entry in UNIX format	double
ipadd	IP address of client	varchar
webpageid	id of the webpage visited	int
ctype	type of content of the webpage visited	varchar
webpageclass	classification of the webpage visited	varchar

5. Results and Discussion

The Squid log parser has been able to retrieve the Squid log file and parse it on the web page. Given the a log file with 659 records, it took about an average of 15.2 seconds for it parse without implementing the webpage crawler. The webpage crawler did not perform as expected because of internet connection problems as well as inaccessible content. It disregards other forms of media based on its content type. It only crawls on web pages that have a content type of ”text”. From the 659 records, 264 of those have a content type of ”text”. Together with the classifier and parser, it performed on an average of 550.33 seconds, or roughly 9 minutes. For every webpage it crawls, it takes about 2.9 seconds for it to gather the text from the web page. Considering the large-sized nature of the log file, this is very slow.

Another log contained 4546 records. For testing purposes only the 1300 records were taken, which were further split into half. It performed on an average of 45 seconds on the first half, since there were not much crawlable web pages available. The latter half did not perform as good as the other one, since there were web pages with inaccessible or blocked content. The web page classifier only works on two classes: Academics and Leisure. If the classifier is unable to classify it within these two, it coins it unclassified. Training for the SVM model given the set of 200 academic words and 200 leisure words took about 20 seconds to process. Classifying inputs, however, takes about less than 10 seconds, so this should not pose much of an issue on multiple classifications. The inputs were weighted based on a dictionary file. If a specific word does not exist in the dictionary, it is weighted as 0. The academic-based web pages have been classified correctly. It had problems classifying the leisure-based web pages. There are also some webpages that were wrongly classified, or haven’t been classified at all. Corrections in the dictionary weights is necessary in order to improve the classification model.

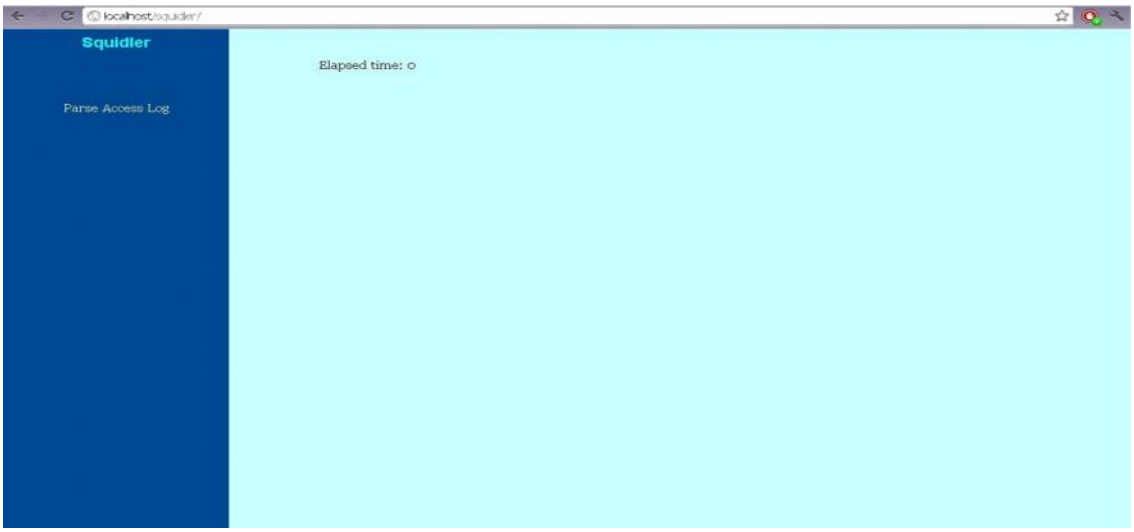

Figure 2. Squidler’s main page.

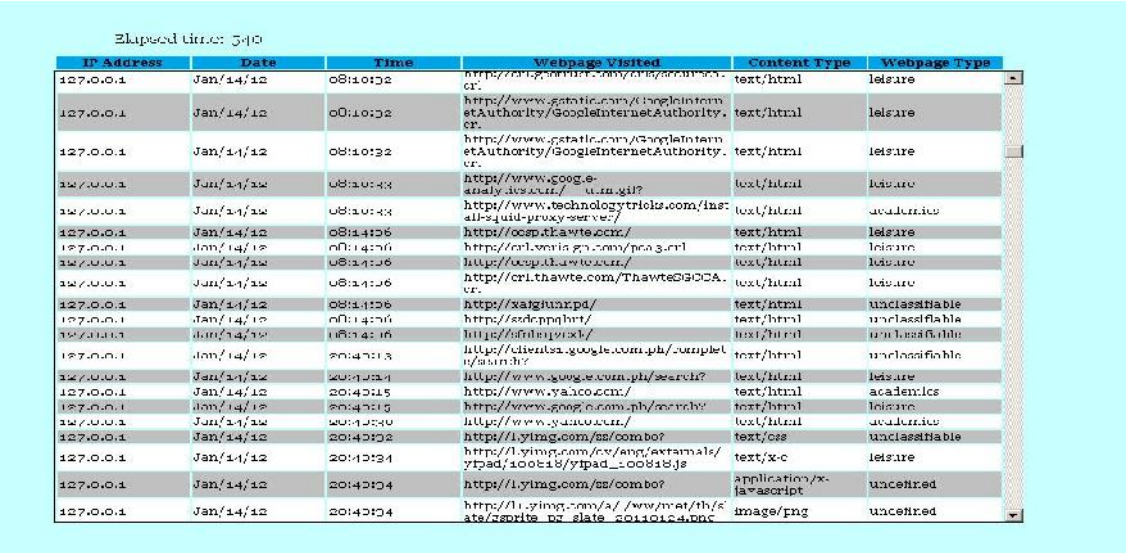

Figure 3. Squidler showing classified pages.

6. Conclusion

The integration of a Squid log parser and a web page classifier using SVM implemented in PHP was completed. The web-based Squid log parser has been able to parse the necessary information, though it would slow down on data logs of enormous file size. A much more systematic method for initializing weights on the training sample is needed to increase the SVM's classification success rate. Adding more categories would probably raise the accuracy as well, though it would slow down the processing time for each record even more. Speeding up the data gathering of the web crawler is also needed, though given the current PHP functions, this is the best that it could have.

References

[1] **Bodammer, F., et al. (n.d.)** SUSE LINUX - Administration Guide. [Online].Available: http://www.novell.com/documentation/ suse91/suselinuxadminguide/html/index.ht ml.

[2] **Overview of proxy server caching.** [Online]. Available: http://publib.boulder.ibm.com/infocenter/wasinfo/ v7r0/index.jsp.

[3] **Qi, X., & Davison, B.D. (2009).** Web page classification - Features and algorithms. [Online]. Available: http://dl.acm.org/ citation.cfm? doid=1459352.1459357.

[4] **Ulrich, J. (n.d.)** Dividing the Haystack: Web Page Classification. [Online]. Available: http://janulrich.org/web/.

[5] **Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2011).** "Web Proxy Cache Content Classification based on Support Vector Machine," *Journal of Artificial Intelligence*, vol. 4, pp. 100–109.

[6] **Monserrat, T.K.P. (2006).** "NetTrace: Information Visualization of a Website using Quasi-Hierarchical Graphs in 3-D Hyperbolic Space," Undergraduate Special Problem, Institute of Computer Science, University of the Philippines, Los Ba˜nos.

[7] **Dilla, R. L. (2009).** "JobSpider: An Online Job Information Retrieval System," Undergraduate Special Problem, Institute of Computer Science, University of the Philippines, Los Ba˜nos.

[8] **Avelar, J. (2011).** Proxy Log Analysis. [Online]. Available: http://www.ehow.com/facts 7826041 proxy-log-analysis.html.

[9] **Barber, I. (2009).** Support Vector Machines In PHP. [Online]. Available: http://phpir.com/support-vector-machines-in-php.

[10] **M. Achour et al. (2011).** PHP Manual. [Online]. Available: http://php.net/manual/en/index.php.

[11] **Oracle. (2011)**. MySQL 5.1 Reference Manual.