

Web usage mining and discovery of association rules from HTTP servers logs

Monash University, Melbourne, Victoria, Australia

Gabriele Bartolini <angusgb@users.sf.net>, student ID 18316719

CSE3212, Data Mining

Lecturer: Robert Redpath

Paper on: Data mining in the Web domain

Due date: Monday 8 October 2001

Copyright (c) 2001 Gabriele Bartolini.

Permission is granted to copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License**, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and no Back-Cover Text. A copy of the license can be found at <http://www.gnu.org/licenses/fdl.txt>.

Abstract

This paper aims to give an overview about the discovery of association rules from Web logs data coming from HTTP servers. After a brief explanation of the terms Web, Web usage and Web usage mining, this document focuses the processes of data preparation and transaction identification which lead to the discovery of rules and usage patterns.

The case of how to mine Web usage data from an HTTP server is presented, which log files are taken under consideration and which information can be obtained from a collection of Web accesses.

Particular attention is given to the need of pre-processing raw data coming from a Web server, which consists of several operations of *cutting*, *merging*, *inserting* and sometimes also *inventing* pieces of information (the case of *path completion*), and which leads to the creation of the *user session file*.

At this point, in order to perform an *association rules* analysis, which is also known as *market basket* analysis, there's the need to define the basket; in the Web environment this is not as clear as in a real supermarket. A phase of *transaction identification* is needed, and in this paper I will show three methods for achieving this: *time window*, *reference length* and *maximal forward reference*.

Finally, after the application of one of these methods, the *transaction file* is created, the *basket* entities are defined and the discovery process of association rules can go on with the application of an AR algorithm such as the *apriori* one.

Part One: Web Usage mining

What's the Web?

The information space known as Web is a collection of resources (Web resources) residing on the Internet, that can be accessed using HTTP and protocols that derive from it (for instance HTTP running over SSL).

As defined in [RFC2396], a resource "can be anything that has identity. Familiar examples include an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), as well as a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources ...".

There is a very interesting work about Web characterization terminology [WCA99] which attempts to define terms often used when talking about the Internet; I want to cite some of them, in order to make the following sections more clear.

The most important concept regarding the Web is of course the resource that a server makes available to clients spread everywhere on the Internet; without any resource, the whole system won't have any sense. Some might say: also the client and the server are important as well. True. Indeed it would not be useful to make available some resources if nobody could access them, as well as clients without servers would be nonsense either.

When a resource is accessed by a client at a specific time and space, we talk of *resource manifestation*.

But ... what's a client and what's a server? Again, I grab some text from the above cited document [WCA99]; the general definition for *client* is "the role adopted by an application when it is retrieving and/or rendering resources or resource manifestations", whereas the specific one for the Web defines the *Web client* as an application "capable of accessing Web resources by issuing requests and render responses containing Web resource manifestations".

On the other hand, the *server* is "the role adopted by an application when it is supplying resources or resource manifestations" to the requesting client.

In most of the cases, spiders and crawlers apart, a request from a client is issued by an user. [WCA99] gives this definition of it: "the principal using a client to interactively retrieve and render resources or resource manifestations. Most of the time it is impossible to exactly identify a user. "

What's an URL?

First of all, a resource in order to be distinguished among others, needs to be identified; in a few words, it needs a name, an *URI*. The term *Uniform Resource Identifier* (URI) "defines a compact string of characters for identifying an abstract or physical resource" [WCA99].

When talking about Web resources, another concept becomes more important than the identification: their location, that is to say *where* to find the resource one is looking for. In this case we are talking of a subset of the URI space, called *URL*: the term *Uniform Resource Locator* "refers to the subset of URI that identify resources via a representation of their primary access

mechanism (e.g., their network 'location'), rather than identifying the resource by name or by some other attribute(s) of that resource. "

For more information about URIs and URLs, is suggested to give a look at [RFC2396]; for the scope of this paper, we only need to know that a URL is something that every Internet user manages daily. Do you ever type *http://www.yahoo.com* or *http://www.altavista.com*? Well, these two are examples of URLs which identify the main page of two of the probably most famous Internet Web sites.

What do we mean by "Web usage" and "Web usage data"?

With *Web usage*, we refer to the behaviour of one or more users on one or more Internet Web sites; the main character is the user, and in order to analyse and to study this topic, some data are needed. In general, Web data are classified as follows [SCDT00]:

- **content data:** any complete or synthetic representation of the resource (the *real data*) such as HTML documents, images, sound files, etc.;
- **structure data:** data describing the structure and the organization of the content through internal tags (*intra-page*) or hyper-links (*inter-page*);
- **usage data:** collection of available data describing the usage of Web resources (e.g. access logs of HTTP servers);
- **user profile data:** demographic information derived from registration.

... and by "Web usage mining"?

The term *Web usage mining* was introduced by Cooley et al. in 1997 [CMS97] when a first attempt of taxonomy of *Web Mining* was done; in particular they define Web mining "as the discovery and analysis of useful information from the World Wide Web". Also it is defined as "the application of data mining techniques to large Web data repositories" [CMS99].

Web usage mining, together with Web content mining and Web structure mining, is a specialization of Web mining; this process of discovery and analysis of patterns focuses its attention on user accesses data (Web usage data).

By citing again the definition that Cooley et al. gave in [CMS97], Web usage mining is the "automatic discovery of user access patterns from Web servers".

Why is Web usage mining important?

A very common Italian proverb says "*Sapere è potere, l'uomo tanto puo' quanto sa*"; in a few words, "*knowledge is power*".

Getting to know information about users' behaviours and their usage patterns, can lead to interesting results that go over descriptive tasks; such examples are dynamic content Web sites which perform mass customisation and personalization by discovering clusters of users with similar access patterns and by adding navigational links and *hints* on the fly [MPT99]. Also, information mined from Web usage data allows restructuring and better management of the

site, giving more effectiveness to it; also the network system can gain benefits from this discovery process [SCDT00].

Sources of data for Web usage mining

A resource manifestation, as said before, takes place after a client performs a request to a server and this gives back a response. However not in the 100% of the cases it's the server that gives an answer back to the client; in a Web architecture, there is another character that plays an important role in this communication system: the HTTP proxy.

As defined in [WCA99], a proxy "is an intermediary which acts as both a server and a client for the purpose of retrieving resources or resource manifestations on behalf of other clients. Clients using a proxy know the proxy is present and that it is an intermediary" (whereas most of the times servers don't!).

A very important feature of proxies regards the so-called *proxy caching*, which can be used to reduce the downloading time of resources requested by users and the load of network traffic between the client and the server.

Caching can also be performed at client side; if from the user's point view, caching is certainly a good feature, from the Web miner's point of view it introduces many sources of errors in the whole analysis process.

Anyway, data that can be used for Web usage mining, can be collected at one of these three parts and thus we talk of [SCDT00]:

- **server level collection:** the server stores data regarding requests performed by the client, thus data regard generally just one source;
- **client level collection:** it is the client itself which sends to a repository information regarding the user's behaviour (this can be done either with an ad-hoc browsing application or through client-side applications running on standard browsers);
- **proxy level collection:** information is stored at the proxy side, thus Web data regards several Websites, but only users whose Web clients pass through the proxy.

In this paper, I will try to cover only the case of a Web server (HTTP server) data.

Part Two: The HTTP server case

What's in our hands?

In this part of the paper, I will try to cover the case of Web usage mining on a Web server; thus, the data that the Web miner is going to analyse regards the one automatically collected by the server, defined *HTTP server information*.

Additional knowledge is represented by every kind of available information in a structured way, like site files, site topology, user's demographic data and last but not least human resources (the so-called *experts*). This is called *auxiliary information*.

HTTP server information

The information that we have at the beginning is automatically gathered by Web servers and is usually collected in the so-called access log file. CERN and NCSA specified a *Common Log Format* (CLF) for every access stored in a log file, and this is supported by most of the HTTP servers.

Here is an example line of access log in common log format:

```
db01.grohe.it - [19/Sep/2001:03:23:53 +0100] "GET / HTTP/1.0" 200 4096
```

Every log entry conforming to the CLF contains these fields:

- **client IP address** or hostname (if DNS lookups are performed);
- **user id** ('-' if anonymous);
- **access time**;
- **HTTP request method** (GET, POST, HEAD, ...);
- **path** of the resource on the Web server (identifying the URL);
- the **protocol** used for the transmission (HTTP/1.0, HTTP/1.1);
- the **status code** returned by the server as response (200 for OK, 404 for not found, ...);
- the **number of bytes** transmitted.

However, this represents the minimal set of fields to be stored in every access log entry; most of the modern Web servers such as Apache and IIS allows the administrator to customise the record track of every row, by inserting further variable values. The most important ones, which if are added to the CLF make up the so-called Combined Log Format (issued by Apache Web Server), are:

- **user agent**, which identify the client application used to retrieve the resource;
- **referrer** or referring resource, which contains the URL of the document issuing the link to the current requested page.

Before the log format customisation was possible, referring URLs and user agents were stored in different log files (*referrer log* and *agent log*); from this moment on, I consider them as available entities, either as separate files (a further *merge* operation is therefore needed) or belonging to a customised format access log file.

So, if a Web server is properly set, we automatically have got information regarding every access, including referring URLs and user agent descriptions.

Auxiliary information

Other available auxiliary information is certainly represented by the site files, that is to say the real data to be made available by the server; from the analysis of those, which can be performed either manually or automatically (through *spiders* or *crawlers*), it is possible to classify documents based on the content and to build the *topology* of the site from the structure data. Cooley et al. define this pre-processing task as *site filter* construction [CMS99].

Specially the topology of the site, which describes the structure links between resources in the site, it is relevant during the pre-processing phase called *path completion*, which aims to build the path followed by a client in order to retrieve a resource, when referring URLs are missing.

In some cases, it is possible to have demographic information of users coming from previous registration processes; this data is usually stored in relational databases independent from the HTTP server. Some might argue about the reliability of this data (sometimes users prefer to give false information), but the discussion of this subject goes beyond the objectives of this paper.

Discovery of usage patterns from HTTP server logs

Before going on, I would like to introduce another term, *episode* and to cite the definition contained in [WCA99]: it is “a subset of related user clicks that occur within a user session”. It is the *related* word that has to catch the attention of the Web usage miner.

In order to discover usage patterns from the available data described above, it is necessary to perform three steps:

1. **pre-processing;**
2. **pattern discovery;**
3. **pattern analysis.**

These phases define the Web mining process and can also be used for the general case, not only the Web server one.

The first phase is the most important of all, because of the complex nature of the Web architecture, and therefore it is the most difficult one. Raw data coming from the Web server is unfortunately incomplete, and only a few fields are available for discovering patterns (IP address, time, user agent); besides, it is almost impossible to identify a user (unless an authentication check has been issued) and more often a single client (when this is behind a proxy). Thus this preparatory step is essential, and its aim is to build an as complete and robust as possible data file (server session file), by gathering information from the different available sources shown in the previous section. And this task is anything but easy.

The pattern discovery phase, consists of different techniques derived from various fields such as statistics, machine learning, data mining, pattern recognition, etc. applied to the Web domain and to the available data.

The most used techniques applied to Web usage data, as described in [CMS99], are:

- **statistical analysis:** this kind of analysis is performed by many tools, available also for free, and its aim is to give a description of the traffic on a Web site, like most visited pages, average daily hits, etc.;
- **association rules:** the main idea is to consider every URL requested by a user in a visit as basket data (item) and to discover relationships with a minimum support level between them; this is the case discussed in this paper from next section;
- **sequential patterns:** the attempt of this technique is to discover time ordered sequences of URLs followed by past users, in order to predict future ones (this is much used for Web advertisement purposes);
- **clustering:** meaningful clusters of URLs can be created by discovering similar characteristics between them according to users behaviours.

In the next sections, I consider only the case of the discovery of *association rules* from an HTTP server data.

Part Three: Association rules from a Web server data

Association rules basic concepts on the Web

Item = Web resource

When considering the process of discovering *association rules* from data, everyone comes up with the classical example of the supermarket and the basket. A basket that is filled of items. It is not a coincidence if the association rules analysis is also known as *market basket analysis*.

On a Web domain, an item can be easily associated to a Web resource, that is to say an URL. However, the basket (or *transaction*) identification is not as easy as this one.

Basket = ... ?

As Mobasher et al. say in [MJHS96], "unlike market basket analysis, where a single transaction is defined naturally, we don't have a natural definition of transaction for the task of mining association rules".

That's the main problem; when purchasing products in a supermarket, every transaction is defined, at the moment of paying the cashier. We don't know if a product has been picked up before another one (in the Web domain we have this information, giving the possibility of mining sequences of patterns), but we have the exact definition of the basket.

In Web terms, we never know when a user leaves our site; he never warns a Web server that his *visit* has ended. A *visit* or *server session* is "a collection of user clicks to a single Web server during a user session" [WCA99].

The only criteria for identifying a server session is to take notice that the client has not surfed through the site for a *reasonably long* time interval (e.g. 30 minutes). Thus, by examining next entries in the access log.

User = ... ?

Another problem regards the user identification. As said before, it is probably more appropriate to refer to *client* instead of *user*. Indeed, if they are not using authentication techniques, it is impossible to know exactly who really requests for resource manifestations on the Web.

The identification of the user is important because the user is the *character* which creates a transaction, which choose what items or resources go into the basket; therefore, in the pre-processing phase, the user identification is applied before the transaction one.

The pre-processing phase

In Web Usage Mining, with the term *pre-processing phase* we intend a set of operations that process the available sources of information (HTTP server and auxiliary ones) and lead to the creation of an *ad-hoc* formatted dataset to be used for knowledge discovery through the application of data mining techniques such as association rules, sequential patterns, clustering.

In this paper I only want to discuss the processing of HTTP server log files, displaying the steps to be taken in order to build these two output files:

- **user session file;**
- **transaction file** (derived from the previous one).

By examining auxiliary information such as site topology, page classification and demographic information of users it is possible to create more accurate output files.

Anyway, the pre-processing phase of HTTP server information is made up of the following five steps:

- **data cleaning;**
- **user identification;**
- **user session identification;**
- **path completion** (at this point we have the *user session file*);
- **transaction identification** (*transaction file* creation).

Data cleaning

Well, it is probably surprising for the end user to know that most of the information stored in an HTTP server log file is useless for the majority of KDD cases. On an average size Web server, access log files easily reach tens of megabytes per day, causing the analysis process to be really slow and inefficient without an initial cleaning task.

Just think that every time a Web browser downloads an HTML document on the Internet, the images included are requested as well, causing each of those accesses to be stored in the log file of the server (unless images automatic load option of the client is switched off).

If our discovery objectives take no notice of images, it is understood that all of these entries have to be excluded from our analysis output files; similar cases of data cleaning regard entries with *HTTP status code* equals to 404 (resource not found on the server), particular resources with some defined characteristics (CGI applications for counters, URLs with some string patterns in their path that is not desired) and also accesses performed by the so-called *spiders* or *crawlers* or *robots*. These are automatic agents that surf the Web in order to gather and store information; the most common case regards search engine spiders, which crawl the Web and index words and documents in databases to be queried by the end user, and link checkers and site managements tools.

Unless our analysis aims to discover knowledge of Web robots agents, log entries caused by these clients have to be pruned; it is not an easy task them. Tan and Kumar [TK01] recently wrote a very interesting paper about this topic.

Finally, a cleaning phase is certainly necessary; however its definition varies from case to case, depending on the aims and objectives of the whole analysis process.

User identification

Once HTTP log files have been cleaned, next step in the data preparation is the identification of the *user*, through heuristics. Unless we are treating a site that provides HTTP authentication and store the *uid* field of the access log files with a meaningful value, the user identification becomes a quite complex task. Unfortunately, this represents the majority of the cases, where accesses and requests are performed anonymously.

As said before, the fields available for the identification of the user, apart the *uid* field (which most of the times is empty), are:

- **IP address** (or hostname if DNS lookups are performed);
- **user agent**;
- **referring URL**.

IP address

Unfortunately, the IP address of the client is not enough to successfully identify a user because many hosts can be presented to the HTTP server under the same IP address or hostname (proxies).

Also the use of DHCP technology by ISPs makes difficult to identify the same user through different TCP/IP connections, because the IP address (and consequently the resolved hostname) changes dynamically (see case 1 in the summary at the end of this paragraph). This can cause to wrongly consider different users as the same one (having the same IP address from the same ISP on a different time), and the same user as two or more different users in the considered period of time (having different IP addresses, see case 3 in the summary).

Moreover, some ISPs give the user more privacy controls, by randomly assigning a different IP address from those available for every single request performed by the client: discovering users in this situation becomes really hard (see case 2 in the summary).

Here follows an example of lines grabbed from the access log of the web server of the Rete Civica of Prato, Italy (the format used for the log file is a customised one, but easily understandable).

```
cache-dl05.proxy.aol.com - [19/Sep/2001:03:18:42 +0200] "GET
/biblio/riviste/r-t/review2.htm HTTP/1.0" 200
"http://google.yahoo.com/bin/query?p=3+waves+of+European+industrialization&h
=0&hs=0" "Mozilla/4.0 (compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x
4.90)"

cache-dl09.proxy.aol.com - [19/Sep/2001:03:18:45 +0200] "GET
/biblio/riviste/img/r-t/rew8.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"

cache-dl09.proxy.aol.com - [19/Sep/2001:03:18:45 +0200] "GET
/biblio/riviste/img/r-t/rew7.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"

cache-dk06.proxy.aol.com - [19/Sep/2001:03:18:45 +0200] "GET
/biblio/riviste/img/r-t/rew9.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"

cache-dl09.proxy.aol.com - [19/Sep/2001:03:18:45 +0200] "GET
/biblio/riviste/img/r-t/rew6.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"

cache-dl09.proxy.aol.com - [19/Sep/2001:03:18:49 +0200] "GET
/biblio/riviste/img/r-t/rew5.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"

cache-dl09.proxy.aol.com - [19/Sep/2001:03:18:49 +0200] "GET
/biblio/riviste/img/r-t/rew4.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"
```

```
cache-dm01.proxy.aol.com - [19/Sep/2001:03:18:50 +0200] "GET
/biblio/riviste/img/r-t/rew4b.jpg HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"
cache-dm01.proxy.aol.com - [19/Sep/2001:03:18:50 +0200] "GET
/biblio/riviste/img/ind.gif HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"
cache-dl04.proxy.aol.com - [19/Sep/2001:03:18:52 +0200] "GET
/biblio/riviste/img/av.gif HTTP/1.0" 200
"http://www.istitutodatini.it/biblio/riviste/r-t/review2.htm" "Mozilla/4.0
(compatible; MSIE 5.5; AOL 5.0; Windows 98; Win 9x 4.90)"
```

User agent

So, by itself the client host field is not reliable; more accuracy to the process of user identification is given by the value of another field: the user agent, which gives a name to the browser used by the client.

Different values of the user agent field on the same client represent different users; this heuristic can lead however to consider two or more users surfing from the same browser in a short time interval as the same user; conversely by surfing from the same client with different browsers can lead to the identification of more instances of users (see case 4 in the summary).

Referring URL

It is also possible to use the referring URL information to discover different users who have the same client host and the same browser, by watching the path they followed in order to get to a resource.

Summary for the User identification process

Anyway, at the end of this phase, additional information is added to every entry, representing a user (user id); moreover the log file is ordered by its value and then by request time.

The operation of user identification on an access log file is a *merge* operation, because it merges single entries into a collection of entries with the same *user id*.

Finally, as Srivastava et al. display in [SCDT00], the typical problems encountered in the identification of the user are:

1. Single IP address / Multiple server sessions: the typical case of ISPs with proxies;
2. Multiple IP address / Single server session: some ISPs assign randomly a different IP address to every single request performed by the client (a famous case is AOL);
3. Multiple IP address / Single user: when the same user accesses the Web, it is very likely that his IP address changes from connection to connection;
4. Multiple agent / single user: the same user can access resources from the Web by using different browsers from the same host.

User session identification

The previous output file ordered by identified users and time of request, could contain requests done in long periods of time and also performed by the same users. It becomes important to *divide* the log entries of the same users in sessions, or visits.

Usually a 30 minutes timeout between sequential requests from the same user is taken in order to close a session.

Path completion

This is a pretty complex task, because it involves the use of referring URLs and auxiliary information (site topology in particular).

There are several reasons why some references are missing in the path that leads to the request of a particular resource; on of the major reasons is that cache mechanisms are applied both by clients and proxies.

However, by examining the site topology and the referrer field it is possible to partially reconstruct the path followed by the user, even if it is still impossible to assign a valid timestamp value for the “fictitious” entry.

At the end of this phase, the *user session file* is ready.

Transaction identification

The identification of transactions varies from case to case, depending on the Web Usage Mining technique that we want to use. In the next section, I want to show the cases involving association rules mining.

Association transactions and their identification

After the path completion phase, the User session file is prepared and formatted; the initial log file is now prepared for data mining and it is a collection of page references grouped by user sessions. As Cooley et al. say in [CMS99], “the goal of transaction identification is to create meaningful clusters of references for each user”.

This process can be achieved by using 3 different methods, which leads to different results:

- **time window;**
- **reference length;**
- **maximal forward reference.**

Once one of the above methods has been applied, the transaction file is ready.

Before analysing in depth these methods, a few definitions must be given: a set of entries of an HTTP server access log is defined as L . A single log entry is l , with $l \in L$.

Every entry l contains these fields:

- $l.ip$: the client IP address (or hostname);

- *l.uid*: the user id of the client;
- *l.url*: the accessed URL;
- *l.time*: the access time of the request.

Time window

As stated in [CMS97] the time window transaction identification module “divides the log for a user up into time intervals no larger than a specified parameter”. A *time window transaction* t is a triple:

Time window transaction:
 $t = \langle ip_t, uid_t, \{(l^t_1.url, l^t_1.time), \dots, (l^t_m.url, l^t_m.time)\} \rangle$
 where, for $1 \leq k \leq m$, $l^t_k \in L$, $l^t_k.ip = ip_t$, $l^t_k.uid = uid_t$
 and $(l^t_m.time - l^t_1.time) \leq W$, with W representing the length of time window

The time window length parameter is the only one that should be provided for the algorithm, making it a *supervised* one.

Content resource or navigation resource

The *reference length* and *maximal forward reference* methods allow the Web miner to identify transactions in a more complex way, by assigning them further information regarding their navigational purposes from an user’s point of view. As Cooley et al. say in [CMS97] and [CMS99], for an Internet user a Web resource can be of two types: an *auxiliary* (or *navigation*) resource or a *content* resource. That’s what they called the *user’s browsing behaviour model*.

An *auxiliary resource* or reference represents an intermediate step for the user in order to get to the *content* resource, which is the one that contains the information. Basically, the auxiliary resource has navigational purposes, so that’s why it is often referred as *navigation* resource.

But, how can the Web miner know whether a resource is a content or navigation one? By knowing the topology of the site, this can be achieved. Otherwise, this is possible by using one of the following methods. The reference length is able to discover content references by considering the time a user spends on a resource, whereas the maximal forward reference considers the path followed by the client (no operation is made on the time field).

Reference length

The most important field on which the *reference length* method is based, is the time field: by considering the time difference occurring between two transactions made by the same user on the same server (two adjacent rows in a user session file), which represents the estimated amount of time the user spends on a resource (*reference length*), it is possible to determine whether the resource is a navigation or content one.

If the time spent on a resource is *long enough*, the resource is considered as a content one, otherwise it is just an auxiliary reference in order to get to a desired goal. How can we estimate if a reference length is “long enough”?

Cooley et al. defined this in [CMS99], through a formula that calculates the *cut off time* by giving just one parameter: the percentage of auxiliary pages in the whole site. This parameter obviously makes the algorithm and the method to be *supervised*. However I don’t want to go over it here, just to state that it is possible to calculate a cut off time, which is discriminatory

and allows us to assign the right label (navigation or content) to a resource, in short, to *classify* a resource. From now on, I assume that we have the cut off time (defined by C).

As long as the reference length needs a *next* entry to be calculated, it goes without saying that the last entry in a user session is considered as content reference.

By the way, it is time to define a *reference length transaction* t as a quadruple:

Reference length transaction:
 $t = \langle ip_t, uid_t, \{(l^{t_1}.url, l^{t_1}.time, l^{t_1}.length), \dots, (l^{t_m}.url, l^{t_m}.time, l^{t_m}.length)\} \rangle$
 where, for $1 \leq k \leq m$, $l^{t_k} \in L$, $l^{t_k}.ip = ip_t$, $l^{t_k}.uid = uid_t$

Let C be the cut off time value,

we define a l^{t_k} reference to be a **navigation** reference if
 for $1 \leq k \leq (m - 1)$: $l^{t_k}.length \leq C$

On the other hand we define a l^{t_k} reference to be a **content** reference if
 $k = m$ or for $1 \leq k \leq (m - 1)$: $l^{t_k}.length > C$

Of course some might say that with this method, we may consider an auxiliary reference as a content one just because maybe when a user was surfing he was called on the phone and left the computer for a time greater than the cut off time. Unfortunately we cannot get to know that, but ... who really knows?

Maximal forward reference

The maximal forward reference is a completely unsupervised method for discovering transactions, which means that no input parameters are required.

Accordingly to Chen et al. [CPY96], "users are apt to travel objects back and forth in accordance with the links and icons provided". Two types of references are so possible: backward and forward references. A *backward* reference, in the same user session, is the revisit of previously visited resource; on the other end, a *forward* reference is the visit of a new resource in the user session path. Anyway, here is the definition of maximal forward reference given by Chen et al. in [CPY96]: "when backward references occur, a forward reference path terminates. This resulting forward reference path is termed a *maximal forward reference*."

By using the examples given by the above cited authors, let's suppose we have this path for a user: {A, B, C, D, C, B, E, G, H, G, W, A, O, U, O, V}. From the application of the MF (Maximal Forward) algorithm, the resulting set of maximal forward reference transactions is made of: {ABCD, ABEGH, ABEGW, AOU, AOV}.

Moreover, a *maximal forward reference* could be considered as the final destination of our temporary navigation path, that is to say the *content* resource, whereas the previous ones in the path could be seen as *navigation* or *auxiliary* resources.

The set of transactions found by the MF algorithm is a mix of content and navigation resources; we could only consider the content ones, and in this case we would have this set of resources: {DHWUV}, i.e. the content-only references.

Unfortunately this method is not suitable in Web sites with a high level of connectivity between resources (especially *content* ones); indeed users tend to use the provided direct links

instead of going backwards and issuing a maximal forward reference. There could be an underestimation of content resources.

Finally, this method does not use any *time* information.

Application of algorithms for association rules discovery

Once transactions have been identified, each of them could represent a *basket*, and each resource an *item*. After the pre-processing step, the Web miner has the *transaction file*, containing the transactions that have been discovered by applying one of the previous shown methods of discovery.

Many algorithms can be used to mine association rules from the data available; one of the most used and famous is the *apriori* algorithm proposed and detailed by Agrawal and Srikant in 1994 [AS94]. This algorithm, given the minimum support and confidence levels, is able to quickly give back rules from a set of data through the discovery of the so-called *large itemset*.

Anyway, this goes beyond the objectives of this paper; the only important thing to know at this point is that with the available data resulting from the pre-processing phase (the transaction file) it is now possible to discover association rules by applying either an existing algorithm or a new one.

Conclusions

This paper has attempted to give an overview to the process of association rules mining from a Web server data. It is not meant to be more than a survey paper, as it is my first one on Web mining.

Due to the heterogenous nature of the Web, it is almost impossible to get accurate knowledge from the common server data (the user identification problem remains still hard to accomplish).

Better results can be obtained by the use of dynamic Web sites, developed using server-side programming languages with information stored in *ad-hoc* databases and with HTTP communications that take advantage of both temporary (session) and permanent (with a long lifetime) *cookies*. Even if the user has the option to disable them, it must be said that using them can only lead to an improvement. I recently started a project which aims to build dynamic Web sites and track down users information, but it needs a lot work: it is called *Domus*, it is written with PHP and it is available at <http://domus.prato.linux.it> .

Also some mechanisms of *cache busting*, when needed, can lead to a more accurate knowledge availability.

Anyway, users must be informed about the uses of the data collected by a Web server, for privacy reasons; and when this will finally be accomplished, the whole process will have benefits. In this direction, the *W3C, World Wide Web Consortium* has taken a very firm decision with the project called *Platform for Privacy Preferences, (P3P)*: it is an “automated way for users to gain more control over the use of personal information on Web sites they visit” (P3P).

Unfortunately I have not been able to write any code for Web usage mining on HTTP servers log, yet; I have also noticed that the Free Software Community lacks of Web usage mining tools able to identify transactions, and it could be a very interesting task to promote a project in this sense and share the source code.

References

- [AS94] Rakesh Agrawal, Ramakrishnan Srikant (1994). *Fast algorithms for mining association rules*.
- [CPY96] Ming-Syan Chen, Jong Soo Park, Philip S. Yu (1996). *Data mining for path traversal patterns in a Web environment*.
- [MJHS96] Bamshad Mobasher, Namit Jain, Eui-Hong Han, Jaideep Srivastava (1996). *Web mining: pattern discovery from World Wide Web Transactions*.
- [CMS97] Robert Cooley, Bamshad Mobasher, Jaideep Srivastava (1997). *Grouping Web page references into transactions for mining World Wide Web browsing patterns*.
- [CMS97b] Robert Cooley, Bamshad Mobasher, Jaideep Srivastava (1997). *Web Mining: information and pattern discovery on the World Wide Web*.
- [MPT99] F. Massegia, P. Poncelet, M. Teisseire (1999). *Using data mining techniques on Web access logs to dynamically improve Hypertext structure*.
- [CMS99] Robert Cooley, Bamshad Mobasher, Jaideep Srivastava (1999). *Data preparation for mining World Wide Web browsing patterns*.
- [SCDT00] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan (2000). *Web Usage Mining: Discovery and applications of usage patterns from Web data*.
- [MDLNSW00] Bamshad Mobasher, Honghua Dai, Tao Luo, Miki Nakagawa, Yuqing Sun, Jim Wiltshire (2000). *Discovery of aggregate usage profiles for Web personalization*.
- [TK01] Pang-Ning Tan, Vipin Kumar (2001). *Discovery of Web robot sessions based on their navigational patterns*.
- [WCA99] Web Characterization Terminology & Definitions Sheet. <http://www.w3.org/1999/05/WCA-terms/> . W3C Working Draft 24-May-1999. 1999.
- [RFC2396] Uniform Resource Identifiers (URI): Generic Syntax. <http://www.rfc-editor.org/rfc/rfc2396.txt> . 1998.
- [P3P] Platform for Privacy Preferences (P3P) Project. <http://www.w3.org/P3P>. 2001.