A Survey of Methods and Algorithms for Mining of Web Access Logs

Ravi Kothari, and Parul Mittal
IBM – India Research Laboratory
Block I, Indian Institute of Technology
Hauz Khas
New Delhi – 110016, INDIA

EMail: {rkothari, mparul}@in.ibm.com

Abstract

The use of the Internet as a channel for communication and commerce makes available massive access logs. The large amount of data in these logs provides significant opportunity to apply empirical methods of analysis to understand the needs of the visitors to a web site and to optimize the site and the recommendations it makes to the individual visitors. There are a number of choices that need to be made in the pre-processing of access logs (required because of proxy level and client level caching as well as other factors) as well as the representation of data and the empirical model used for mining. This manuscript reviews some of the popular methods of pre-processing, data representation and methods and algorithms for subsequent mining.

1 Introduction

All *incoming client requests* are recorded by web servers in the *access log* file. For example, the *common log file format* [1] that is supported by all HTTP servers records the name of the remote host, the remote login name of the user, the username as which the user has authenticated himself or herself, the date of request, the time of request, the actual request as it came from the client, the HTTP status code returned to the client, and the number of bytes transferred. In some cases, for example when the *extended log file format* is used or when *cookies* are used, additional information is available. In almost all cases however, the entries specified in the common log file format are recorded by web servers in the access logs. These access logs contain a *footprint* of each users visit to the site and an increasing amount of attention has recently been focused on using data analysis techniques to extract information from these access logs.

Two factors have prompted a large amount of focus on web usage mining. First, the increasing use of the web has created massive log files – this large amount of data allows empirical methods of analysis to be employed with some degree of confidence. Second, the use of the World Wide Web (WWW) as a channel for commerce (rather than merely as a sophisticated channel for catalog distribution) has created an increased demand for methods which can discover the information embedded within these access logs. The information discovered from these access logs can be used for,

- Improving the site design thereby allowing the most frequently used pages to be more

directly accessible [2]. As an extension, information gained from access logs can be used to adapt the page viewed by a user [3].

- Creating coordinated and more effective advertisement campaigns that span across multiple *page views* of frequently visited paths [4].
- Clustering the users into segments [5, 6] allowing for segment specific recommendations.
- Correlating subsequent page views (when page view attributes are known) to determine if a user is randomly browsing the site or is looking for something in a focused manner. Recommendations in either case can be customized.
- Discovering unusual access patterns and prevent security breaches.

The above mentioned use for information recovered from web access logs are representative. Perhaps because of its applicability, web usage mining research has seen continued and significant activity over the past several years. Indeed, because of the rapid change brought about by this high level of research activity, it becomes important to *continually* review the state-of-the-art and to identify potential challenges.

In that spirit, we present a review of the present state-of-the-art in mining of web access logs. Mining of access logs is a subset of the the more general web mining (see [7] for a review of general web mining). Nonetheless, because of the rather large volume of work in the area of usage mining it is difficult to fully cover all aspects of web mining in a single article. We have laid out the rest of this manuscript as follows. In Section 2 we review the heuristics and methods that have typically been used to preprocess access logs. We consider the preprocessing stage to discard entries that are irrelevant to the subsequent mining (often this is application dependent) as well as to identify those entries in the access logs that (are thought to) belong to the same user. In Section 3 we outline some of the notation used in the rest of the paper. We also briefly provide information on the common data storage models. In Section 4 we provide an overview of the different data models. Data models are fundamental in the sense that they encapsulate the underlying assumptions that are made about cause-effect relationships. In Section 5 we review the methods for the discovery of rules from the sub-sessions. These rules often serve as the basis of pre-fetching schemes to reduce latency, page recommnedation engines and other applications. In Section 6 we provide an overview of clustering of usage paths. Often the purpose of clustering of user navigation paths is to segment users for the purposes of personalization and recommendation. We conclude this paper with some open issues and conclusions in Section 7.

2 Access Log Preprocessing

The main goals of access log preprocessing are to discard irrelevant entries, to assign the remaining entries in the log to individual visitors, and to prepare meaningful *sub-sessions* or *transactions* from the entries assigned to an individual visitor (*user session*) [8]. Attaining these goals is difficult in part because of *proxy servers* which result in access requests from multiple users to be logged as if they are coming from the same IP address. The use of *proxy level* and *client level caching* makes it difficult to extract meaningful sub-sessions and in some cases make it impossible. To add to the difficulty, the large size of access logs (on an average $10,000$ requests add approximately 1MByte to the access log file – it is not uncommon for most sites to have access logs that run into hundreds of MBytes) require the preprocessing method to be very efficient in terms of storage and speed. To some extent, the difficulties may be alleviated through the use of a *cookie* or through the use of a special purpose logging application that is installed on the client side [9]. However, it is not appropriate to count on these mechanisms as privacy concerns may prompt a user to no cooperate and disable these mechanisms. Access

log preprocessing thus has to make "inferences" in the presence of a considerable amount of uncertainty. Additional details of commonly used preprocessing techniques follow. A more detailed account also appears in [10, 11].

Discarding irrelevant entries: This is perhaps the easier part of access log preprocessing and is dependent on the subsequent mining related tasks. Table 1 shows an excerpt from a typical access log that uses the common log file format. An explanation of the various fields in the log are explained in the table caption. Entries in this log are from various IP addresses; there is also a mix of redirection (status code of 301), successful response (status code of 200), and an unsuccessful response (status code of 404).

The HTTP protocol requires a separate connection for each file that is requested. Many entries in the log file thus related to images and other scripts that accompany a specific page. In the example log shown in Table 1 for example, photo.gif is perhaps an image that appears on the main page. Entries that relate to images can be discarded for most (not all) mining related tasks (the second and fourth rows of Table 1). Redirection entries (the fifth row in Table 1) as can resource not found type of entries (the eighth row in Table 1)[1]. The remaining entries in the log file are assigned to individual visitors as discussed below.

```
xxx.xx.xxx.xx - - [03/Oct/1997:03:23:21 -0600] "GET /main.html HTTP/1.0" 200 7800
xxx.xx.xxx.xx - - [03/Oct/1997:03:23:23 -0600] "GET /photo.gif HTTP/1.0" 200 2010
yyy.yy.yyy.yy - - [03/Oct/1997:03:23:24 -0600] "GET /main.html HTTP/1.0" 200 7800
yyy.yy.yyy.yy - - [03/Oct/1997:03:23:25 -0600] "GET /photo.html HTTP/1.0" 200 2010
xxx.xx.xxx.xx - - [03/Oct/1997:03:24:00 -0600] "GET /pub.html HTTP/1.0" 301 800
zzz.zz.zzz.zz - - [03/Oct/1997:03:24:03 -0600] "GET /main.html HTTP/1.0" 200 7800
xxx.xx.xxx.xx - - [03/Oct/1997:03:24:04 -0600] "GET /paper.html HTTP/1.0" 200 9040
zzz.zz.zzz.zz - - [03/Oct/1997:03:24:05 -0600] "GET /pubs.html HTTP/1.0" 404 524
xxx.xx.xxx.xx - - [03/Oct/1997:03:24:06 -0600] "GET /personal.html HTTP/1.0" 200 981
xxx.xx.xxx.xx - - [03/Oct/1997:03:24:08 -0600] "GET /trip.html HTTP/1.0" 200 4021
zzz.zz.zzz.zz - - [03/Oct/1997:03:24:09 -0600] "GET /main.html HTTP/1.0" 200 7800
yyy.yy.yyy.yy - - [03/Oct/1997:03:24:12 -0600] "GET /links.html HTTP/1.0" 200 10204
```

Table 1: An excerpt from a typical log file that uses the common log file format. The first entry is the IP address or the hostname of the user requesting the page (e.g. xxx.xx.xxx.xx). The second entry is the response received from the user's *identd* server – this is seldom used (e.g. unused here and shown as -). The third entry is the username as which the user has authenticated himself or herself (used when access to certain pages is restricted) (e.g. unused here and shown as -). The fourth entry records the date and time of access and also the offset of the time relative to Greenwich Mean Time (e.g. [03/Oct/1997:03:23:21 -0600]). The fifth entry is the actual request received from the user (e.g. "GET /main.html HTTP/1.0"). The actual request is followed by two entries: the status code returned by the server (e.g. 200) and the actual number of bytes returned by the server not counting the header line (e.g. 7800).

Assigning remaining entries to individual visitors: The presence of proxy servers makes the assignment of individual entries to individual visitors difficult. To the web server, request made by multiple users appear to come from the same IP address and are recorded as such. In some cases, Internet service providers randomly assign a unique IP address to every single request. In the absence of a client side logging mechanism or information from cookies, identifying log entries with individual users is an ill posed one and can only be partially solved based. Often inferences are made based on three parameters. Of these,

[1] A large number of such errors indicates a problem with the referring page (which may be within or outside the site). For the type of mining tasks considered here, they can be ignored.

one parameter appears in Table 1; the other two are not shown in Table 1 though are commonly recorded in the *Combined Log Format* issued by the Apache Web Server.

- *Browser software and operating system:* This entry refers to the browser being used by the client as well the operating system in use at the client side. Thus accesses with the same browser and operating system can be grouped. While this allows disambiguation to a certain extent, it is by no means definitive.
- *Referring page:* The referring page can also be used to assign log entries to individual users. For example, when the site topology is known then it can be determined if a page is directly accessible from a previous page.
- *Username:* Finally, if the username entry in the log table is filled (usually filled when access is to restricted pages), then it is possible to assign different log entries to the different users.

The above three heuristics allow assignment of different log entries to different users. Even when used in combination it is not possible to do this assignment without the possibility of error. However, in the absence of a client side logging mechanism or information from cookies the above mentioned approaches represent the best that is possible.

Sub-session identification: When the log files span a long interval, it is possible that the same users visit the site more than once. Thus it is possible that the pages assigned to the users are from two independent visits. *Session identification* attempts to segment all the log entries assigned to a particular user based on temporal coherency, i.e., by creating sessions such that subsequent accesses within a group are separated by access times that are within a certain pre-determined threshold. The threshold can be empirically determined – in [10] it is taken to be 25.5 minutes which corresponds to 1.5 times the standard deviation of the time between user interface requests. Even within a session however, it is possible that a user follows two unrelated browsing paths. For example, initially a user might be traversing some garment related links and then uses the menu bar to jump to the sporting section of the web site. The goal of sub-session (or *transaction*) identification is to segment a session into smaller (semantically connected) sessions. The concept of a *maximum forward reference* [12] can be used to identify sub-sessions. A maximum forward reference being defined as the longest path before backtracking occurs. This method would result in a session such as $\{a, b, c, a, e\}$ to be represented as two sub-sessions $\{a, b, c\}$ and $\{a, e\}$. The *reference length* approach identifies sub-sessions based on the assumption that the time spent on a page is proportional to the degree to which a page is meaningful to a user. Details on the method to estimate the cutoff time for classifying an access as *auxiliary* (intermediate *en route* page) or *content* appear in [8]. Overall, the reference length method results in very small sub-sessions. The *time window* method divides a session into sub-sessions which span a time interval no larger than a specified interval [8]. All of these three methods (maximum forward reference, reference length and time window) rely on time to divide a session into sub-sessions. In reality, sub-session identification requires some information about the content of pages that appear in a users browsing path. However, content analysis of different pages and finding the distance between them is time consuming and itself not very accurate given the present state of research. Approximations, such as that presented in [13] can be used to infer the similarity between two pages. A point in a session which has an abrupt change in the similarity between the preceding and the following page access can be used to demarcate sub-sessions. Preprocessing of the access log shown in Table 1 based on discarding of entries as discussed above and sub-session identification based on content would result in the sub-sessions shown in Table 2. User # 1 initially followed the professional trail main.html $\rightarrow$ paper.html and then followed a personal trail personal.html $\rightarrow$ trip.html. Content based analysis (or its approximation) would correctly identify these two sub-sessions whereas the other methods of sub-session identification which are essentially based on time would not.

User #	Sessions	Sub-sessions
1	A → B → C → D	A → B C → D
2	A → A	A A
3	A → E	A → E

Table 2: The sessions and sub-sessions identified with each user. In the notation used, User # 1 is from the IP address xxx.xx.xxx.xx, User # 2 is from the IP address yyy.yy.yyy.yy and User # 3 is from the IP address zzz.zz.zzz.zz. A refers to main.html, B refers to paper.html, C refers to personal.html, D refers to trip.html, and E refers to links.html.

3 Notation

Discarding of entries amounts to segments of the overall activity of a user being discarded. In many cases, the difference between the access time of the following page from the access time of the present page is used to indicate the time spent on the present page. However, if an intermediate entry is discarded, the difference between the access time of two subsequent entries in the preprocessed log would incorrectly indicate the amount of time spent on a page. To clarify this and for ease of reference through the remainder of this manuscript we introduce some notation.

We propose to mathematically represent a sub-session $\mathcal{Q}_i$ resulting from the pre-processing as, $\mathcal{Q}_i = \{(q_{i1}, t_{i1}, \Delta t_{i1}), (q_{i2}, t_{i2}, \Delta t_{i2}), \ldots, (q_{in}, t_{in}, \Delta t_{in})\}$. Note that the first subscript represents sub-session number, while the second subscript represents the position in the sub-session. Thus, q_{i1} represents the first page in the sub-session i, t_{i1} represents the physical time at which the page was accessed, Δt_{i1} represents the time that the user spent on the page q_{i1}, and n is the total number of pages in sub-session $\mathcal{Q}_i$. We often use $|\mathcal{Q}_i|$ to represent the length of the sub-session. In this case, $|\mathcal{Q}_i| = n$. Δt_{i1} thus has to be computed as the difference in the access time from the following page even if that page is discarded. Depending on the subsequent mining task, it may be necessary to also store the sub-sessions corresponding to each user. In that case, we use the notation $\mathcal{U}_i$ to represent the set of sub-sessions that belong to the i^{th} user. We use $|\mathcal{U}_i|$ to represent the number of sub-sessions that are identified with the i^{th} user.

Physically, the sessions may be *stored* in a cube model as proposed in [14]. The cube model organizes a session along three dimensions – the *component dimension*, the *attribute dimension*, and the *session dimension*. Along the component dimension, a session is represented as a set of ordered pages. Attributes associated with each page are stored along the attribute dimension. The attributes can be for example, the time spent in viewing the page, the category of the page and son on. The session dimension indexes the individual session. The cube model is flexible in that different slices of the data provide the data in a form suitable for various operations. A more flexible model is based on a general data cube which may have more than 3-dimensions [15]. The use of standard data warehousing model allows the application of OLAP (On Line Analytical Processing) operations and provides considerable flexibility in viewing the data from different perspectives [16, 17, 18]. When the application is very specific, then standard data structures (e.g. linked lists) can be used for storage. For some applications such as path prediction more compact representation can be achieved using tries [19].

4 Data Models

In the previous section, we provided a mathematical representation of the (sub-)sessions and indicated some ways of storing the sessions. However, for the purposes of subsequent mining tasks, a data model has to be devised within which the user sub-sessions can be represented. Either consciously or unconsciously, each model represents a trade-off between the complexity, the predictive accuracy required, the ease with which subsequent mining can be performed, the information that is lost (for example, sequence information is lost in some representations) and so on.

It is perhaps necessary to distinguish between storage and data models. *While storage refers to the physical arrangement and layout of the data, a data model embodies the underlying assumptions that are made about cause-and-effect relationships.* In the following we provide an overview of some of the data models that have been proposed.

In [20] a *dependency graph* is used to store all accesses to the pages at a web site. Each node represents a page. The weight on the edge between two nodes (say A and B) is proportional to the frequency with which B is accessed within some number of accesses after A was accessed. Instead of the number of accesses, time is used in the model used in [21]. The notion of conditional probability for transitioning from a page to another page is computed using the concept of a *cooccurrence matrix* in [13]. The cooccurrence matrix is a square matrix of dimension equal to the total number of pages. A *constraint vector* is defined as a 3-tuple comprising of $\theta_g \in \mathbb{Z}^+$, $\theta_v \in \mathbb{R}^+$, and $\theta_e \in \mathbb{R}^+$. Entries in the cooccurence matrix are incremented each time a page i cooccurs with page j while satisfying the constraint vector. θ_g is a positive integer (> 0) that is related to the maximum allowable "gap" between page views. θ_v is a positive (> 0) real number that is related to the "minimum" time that a page must be viewed for. θ_v thus allows for the possibility that rapid clicks are not used for subsequent inference. θ_e is a positive (> 0) real number that is related to the maximum acceptable "elapsed" time between two page views. θ_e thus allows for the possibility that a user in interrupted or is multi-tasking in which case the connection between the page views might be suspect. The model of [13] also allows for the flexibility in enlarging or reducing the constraint vector depending on the demands of the subsequent mining task. One thing worth noting that in the above models, pages within a certain "gap" of each other are said to cooccur. They do not consider the sequence within the gap. The *n-grams* model of [22] is based on considering the sequence information within the "gap". An n-gram is simply an ordered sequence of pages visited in a sub-session as in $< q_1, q_2, \ldots, q_n >$. The probability of transition is then conditioned over the sequence desired.

In [23], the collection of user navigation sessions is modeled as a *hypertext probabilistic language* generated by a *hypertext probabilistic grammar* (HPG). The probability of a grammar string is given by the product of the productions used in arriving at the string. They introduce the concept of a *start* state S and a *finish* state F of navigation sessions. A production with S on its left side are *start productions* and the productions corresponding to the links between the pages are *transitive productions.* To ue the example given in [23], assuming that a certain page, say there are a total of 24 pages and a page A was the first page in a sub-session and the total number of visits to A is 4. Then the probability of its start production is given as, $(\alpha \times 4]/24 + ([1-\alpha] \times 2)/(4+2)$. α is a user chosen parameter between 0 and 1 and introduces a relative weighting between the fraction of times a page is visited and the fraction of times it is visited as the starting state. Transitive productions can be similarly defined based on the asssumption that the choice of the next page depends only on the current page of the user; strings generated by the grammar thus correspond to sessions.

As mentioned at the beginning of this Section, a data model represents a trade-off between several parameters. The complexity or accuracy of n-grams or production rules may not be required in applications where the end goal is simply to find *association rules* or *frequent itemsets.*

5 Discovering Rules

Discovering rules from sub-sessions is one of the most common of mining tasks. A rule is something of the form: if *antecedent* then *consequent* or more succinctly as $A \implies B$. Such rules form the core of many pre-fetching schemes [24], recommender systems [25], site reorganization efforts [26, 27, 28] and so on. In part, the popularity of rules arises from the relative ease with which readily implementable knowledge can be discovered. This ease of "closing-the-loop" (from discovering to implementing so as to maximize an objective such as increased revenue through *cross-selling*) makes the discovery of rules a cornerstone of contemporary access log mining.

There are multiple types of rules. For example, the goal in sub-sequence analysis is to identify the longest sub-sequences [22]. For example, the sub-sequence (A, B, C) appear in many sub-sessions in that order. The burden of deriving a consequent is not necessarily that of the sub-sequence analysis algorithm. However, the order of terms appearing in the antecedent is relevant. In association rules, the order is irrelavant though the algorithm has the burden of identifying the consequent [29]. For example, a typical association rule can be of the type $(A \text{ AND } B) \implies C$ as in *People who bought a portable CD player and a pair of shorts also bought a pair of running shoes* 30% *of the time*. Here, the order is irrelavant and the rule holds for a customer who buys a pair of shorts prior to buying a CD player. The above two types of rules can be obtained directly from sub-session information obtained by preprocessing access logs. In yet other types of rules, such as those obtained through decision tree induction [30, 31, 32], it is necessary to first prepare a *feature space* from the sub-sessions. Some examples of features include the length of a sub-session, the number of categories traversed, domain related information (whether from a .edu or .com or some other domain) and so on. In decision tree induction, the goal is to find rules in terms of these features and not directly in terms of pages viewed. Such rules are broadly applicable though they are derived undre *supervision*, i.e. a *training dataset* is required for their induction. Figure 1 provides a taxonomy of the different types of rules. However, due to the strong links that decision tree based rule induction has with other classification methods, we discuss rules derived from decision trees later.

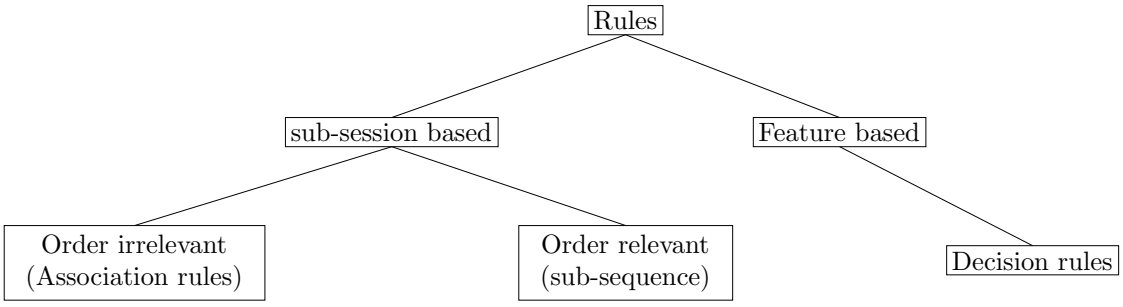

Figure 1: Taxonomy of the different type of rules.

5.1 Frequent Itemsets and Association Rules

Frequent itemset discovery leads to the discovery of a set of pages that cooccur in sub-sessions[2]. It is easy to see that if it is discovered that people who visited (or bought from) page A also

[2]The term itemset is derived from *basket analysis* in which a transaction may record the purchase of several items. In the present context of mining web logs, items refer to page views and transactions refer to sub-sessions.

visited (or bought from) page B, then it becomes possible to build recommender systems in which the content of page $B(A)$ is promoted when a visitor is on page $A(B)$. Such *cross-selling* can be particularly effective when the frequent itemsets are discovered by segments, and it possible to determine the segment that a visitor to a site belongs to. Frequent itemsets also provide clues to site reorganization. For example, if pages A and B frequently cooccur and are related, then it suggests that perhaps their contents can be merged. This may not be always possible and depends on other factors including download time and so on.

Association rules impose a directionality constraint on frequent itemsets. A typical association rule for example is: *People who bought a portable CD player also bought a pair of running shoes* 30% *of the time*. Succinctly, association rules are often written as $A \implies B$ which may be interpreted as: if A then B. The expression to the left of $\implies$ is the *antecedent* while the one to the right is the *consequent*. Two measures are often associated with association rules,

- *Support:* The support for an association rule is the probability of the itemsets appearing together. In practise, the empirical probability as determined from the access logs can be taken as the support. Succinctly the support for an association rule $A \implies B$ can be written as,
$$S(A \implies B) = P(A \cap B) \tag{1}$$
- *Confidence:* The confidence of an association rule is the probability of the consequent being true when the antecedent is true. Succinctly the confidence for an association rule $A \implies B$ can be
$$C(A \implies B) = P(B|A) = \frac{P(A \cap B)}{P(A)} \tag{2}$$

The minimum support and confidence values are user supplied; having a low value for the support leads to many rules being discovered while having a low confidence value leads to poor predictability. A fast and frequently used algorithm for finding association rules is the *Apriori* algorithm [29] (see also [33]). The algorithm is iterative (multi-pass) and is based on the assumption that any subset of a frequently occurring itemset is also frequent. Thus the algorithm starts by finding items (in the present context, pages) that are frequent. Those that have a support larger than the minimum specified support are used as *seed sets* and combined with other qualifying sets to form yet larger item sets. During the process, intermediate rules that do not meet the minimum specified support and confidence are discarded. It has been shown that the APriori algorithms scales linearly with the number of transactions (sub-sessions) [29]. The algorithm given in [34] is similar in spirit though uses a hash table to reduce the qualifying sets that are considered at each iteration. Finally, to deal with large or changing databases, an incremental updating of association rules can be utilized [35].

5.2 Finding Longest Common and Longest Repeating Sub-sequences

A sub-sequence may be described as consecutive series of pages that appear in sub-session with a frequency greater than a pre-specified threshold. To be most useful, such sub-sequences should be of the largest possible length. More specifically, the goal is to find the longest repeating sub-sequences where a longest sub-sequence is one which is not always contained in another longest repeating sub-sequence [22, 19]. Once found, information from sub-sequences can be used, for example to pre-fetch pages and reduce latency, to recommend related pages and to improve site design.

One of the most well founded and popular way of finding the longest common sub-sequence is based on the so-called *Levenshtein distance* or, as it is more commonly called, the *edit distance* [36]. The edit distance reflects the minimum amount of effort required in transforming two sequences such that they are identical. The transformation typically consists of three

operations namely, insertion, deletion, and replacement with associated costs C_I, C_D, and C_R respectively. The edit distance $d(\mathcal{Q}_1, \mathcal{Q}_2)$ between two sequences, say $\mathcal{Q}_1$ and $\mathcal{Q}_2$ of lengths n and m respectively, can then be obtained from the following recurrence,

$$\begin{aligned} d_{0,0} &= 0 \\ d_{i,0} &= d_{i-1,0} + C_D \\ d_{0,j} &= d_{0,j-1} + C_I \\ d_{i,j} &= \min \left\{ \begin{array}{l} d_{i-1,j} + C_D \\ d_{i,j-1} + C_I \\ \left\{ \begin{array}{ll} 0 & \text{if } q_{1i} = q_{2j} \\ d_{i-1,j-1} + C_R & \text{if } q_{1i} \neq q_{2j} \end{array} \right. \end{array} \right. \end{aligned} \tag{3}$$

The recurrence of Equation (3) can be solved using *Dynamic Programming* [37]. Several variations to the above method for finding the similarity between two strings have been proposed [38, 39, 40, 41]. When, the recurrence of Equation (3) continues until $i = n$ and $j = m$ one recovers $d(\mathcal{Q}_1, \mathcal{Q}_2) = d_{nm}$ or the distance between the entire sub-sequences $\mathcal{Q}_1$ and $\mathcal{Q}_2$. For some $i < n$ and $j < m$ one recovers the distance between the first i^{th} page views of $\mathcal{Q}_1$ and the first j^{th} page views of $\mathcal{Q}_2$. One can then find the longest sub-sequence from amongst those sub-sequences that have a distance less than a predetermined threshold. Modifications can be made when if the longest repeating sub-sequence is desired so that all sub-sequences repeating less than a predetermined number of times are discarded.

A somewhat different approach to finding the longest repeating sub-sequences is to use the n-grams introduced earlier. Recall that an n-gram is simply an ordered sequence of pages as in $< q_1, q_2, \ldots, q_n >$. The empirical probability of a visit to page q_n, given visits to pages $q_1, q_2, \ldots, q_{n-1}$ can be expressed as,

$$P(q_n | q_1, q_2, \ldots, q_{n-1}) \tag{4}$$

which can be computed from the access logs. Storing all potential sub-paths of a sub-session results in an explosive growth in storage requirements. For a sub-session of length $|\mathcal{Q}_i|$, all the possible sub-paths require storage proportional to $O(|\mathcal{Q}_i|)^3$ in the worst case. Various heuristics such as discarding *l*ow probability transitions or the *s*pecificity heuristic [22] can be used to minimize the computational requirements. Thus transitions between successive pages or sub-sequences with a low probability can be removed from further consideration and higher order path are given preference over lower order ones [22]. Data structures such as such as tries [19] can be used to contain the storage requirements.

6 Discovering Clusters

Clustering is a popular unsupervised pattern classication method which partitions the input space into k partitions. Though there are varying notions that one may attach to clustering (for example, cluster centers are points of local maxima of the probability density), quite simply the goal of clustering is to assign patterns that are *close* to one cluster and to assign patterns that are not close to different clusters. Of course, the notion of closeness requires some way of finding the distance between two patterns and we shall comment on this slightly later in this Section.

Nonetheless, based on a given measure of closeness (distance), when the partitions are disjoint, one obtains the so-called *exclusive classication* (i.e.each data point belongs to only one cluster). On the other hand, when the partitions are overlapping, one obtains the so-called *non-exclusive classication* [42]. Clustering methods which seek to provide exclusive classication can be broadly classied into *partitioning* and *hierarchical*.

Hierarchical clustering algorithms come in two variations. The so-called *agglomerative methods* (e.g. *average linkage method*) start with as many cluster centers as there are patterns and merge the closest cluster centers as one proceeds higher up in the hierarchy. *Divisive methods* (e.g. *polythetic divisive method*) proceed by grouping all patterns together in a cluster and splitting cluster centers higher up in the hierarchy.

Within the context of access log mining, a pattern may be thought of as a sequence or a sub-sequence corresponding to a particular user (session based or even from multiple sessions) and the result of clustering is a paritioning of the set of users into segments. Such segmentation (based on clickstream as opposed to, say, attribute such as purchase history based segmentation) has the advantage that user segmentation is possible even when a user may not logged in or is not registered. Since an overwhelming percentage of users (by some estimates, almost 90% of registered users do not log in until they are ready to checkout or require their customized settings, clickstream derived segmentation assumes great importance. Despite this importance and the relative popularity of clustering for access log mining, there have not been many new algorithms. Much of the work has centered on defining a measure of distance that is required during clustering. Since user paths (sub-sequences) may be thought of as symbolic sequences, finding the distance between two clickstreams (patterns) is not straightforward.

In [43], the distance (or similarity) between two clickstreams is taken as the length of the logest common subsequence. The measure proposed has a "similarity component" which reflects how similar the two paths are in the region of their overlap and an "importance component" which uses the geometric mean of the fraction of time occupied by the respective clickstreams in the region of overlap. The similarity is then defined as the product of the two components. However, the measure defined by the authors still uses a base LCS extractor [36] which uses the assumption that any two pages that are not identical are dissimilar to the same extent. Often, at the level of a web page many user paths may have little similarity. Often, it is helpful to first categorize the pages (into so-called concepts) and the original clickstream (of web pages) can be mapped into a stream of concepts. In a sense, this attempts to approach the desirable (and difficult) content based analysis of web pages to find the dissimilarity between two web pages.

As discussed previosuly, in [13] the joint probability of cooccurrence of web pages is used an indicator of the distance between two web pages. Somewhat similar in spirit is the algorithm based on hypergraph partitioning [44, 45] in which frequent itemsets are found. The web pages represent the nodes and the edges reflect the frequent itemsets that are found. A hypergraph partitioning algorithm [46], which attempts to partition such that the weights of the edges cut by the partitioning is minimized, is used to find clusters. The use of frequent item sets weakly captures the notion of "cooccurence implies similarity," though it cannot take the sequence, time and additional information that the approach in [13] can.

In [44], an additional method that represents each web page by a feature vector formed from the word frequencies is used. The algorithm attempts to partition the feature space by a hyperplane that is normal to the *principal component* and passes through the aithmetic mean of the features (pages). The process is repeated as many times as the desired number of clusters. The principal component is the direction of most variance and the heuristic that the authors use is that a direction normal to the principal component would have less variance (more similarity) and hence they belong in one cluster. The authors approach the computation by first computing the principal component and then finding the normal, though there are methods which directly provide the minor component (direction of least variance) [47].

Somewhat surprisingly, the so-called *subspace clustering* methods have not been used for clustering within the context of clustering usage paths [48, 49, 50]. In subspace clustering a cluster is represented by a subspace and different clusters reside in different subspaces. When the subspaces are linear combinations of the original basis, efficient algorithms can be used to find the subspaces. However, when the clusters are of complex geometry then a non-linear kernel can be used to map the data to some high dimensional space. Subspace clustering in

this high dimensional space corresponds to clustering with a complex geometry in the original feature space.

7 Conclusions

In this manuscript, we presented an overview of the methods and algorithms for mining of web access logs. While there has been a considerable amount of activity in terms of the methods uses in such analysis, it is likely that the combination of content and navigation will allow higher quality inferences to be made than is possible from navigation patterns alone. To some extent the movement towards increasingly richer forms of document representations (e.g. XHTML) also creates additional opportunities.

Though access logs are mild in terms of the type of data they contain, they are often done alongside other data exchange, and clearly stated policies along with responsible use of information are consistent with best business practices. Under such conditions, web access log analysis is a valuable tool for continually optimizing a web site and for making a visit to the site more relevant and satisfying to a user.

References

[1] W3C: The World Wide Web Consortium, "Logging control in W3C HTTPD," 1995.

[2] K.-P. Huber C. Theusinger, "Analyzing the footsteps of your customers: A case study by ASK—net and SAS Institute GmbH," in *WebKDD-2000*, 2000.

[3] T. Kamdar and A. Joshi, "On creating adaptive web servers using weblog mining," Tech. Rep. TR-CS-00-05, University of Maryland, Baltimore County, 2000.

[4] N. Modani, P. Mittal, A. Nanavati, and B. Srivastava, "Series of dynamic targeted recommendations," in *Proc. International Conference on ECommerce and Web Technologies (ECWeb) (LNCS 2455)*, 2002, pp. 262–272.

[5] Y. Fu, K. Sandhu, and M.-Y. Shih, "Fast clustering of web users based on navigation patterns," in *Proc. World Multiconference on Systemics, Cybernetics and Informatics (SCI/ISAS'99)*, Orlando, USA, 1999, pp. 560–567.

[6] B. Hay, G. Wets, and K. Vanhoof, "Clustering navigation patterns on a website using a sequence alignment method," in *Proc. IJCAI's Workshop on Intelligent Techniques for Web Personalisation*, 2001.

[7] R. Kosala and H. Blockeel, "Web mining research: A survey," *ACM SIGKDD Explorations*, vol. 2, no. 1, pp. 1–15, 2000.

[8] R. Cooley, B. Mobasher, and J. Srivastava, "Data preparation for mining world wide web browsing patterns," *Knowledge and Information Systems*, vol. 1, no. 1, pp. 5–32, 1999.

[9] C. Shahabi, A. Zarkesh, J. Adibi, and V. Shah, "Knowledge discovery from users web-page navigation," in *Proc.* 7^{th} *International Conference on Research Issues in Data Engineering*, 1997, pp. 20–29.

[10] L. Catledge and J. Pitkow, "Charcacterzing browsing strategies in WWW," *Computer Networks and ISDN Systems*, vol. 27, no. 6, pp. 1065–1073, 1995.

[11] J. Pitkow, "In search of reliable usage data on the WWW," in *Proceedings of the* 6^{th} *WWW Conference*, 1997, pp. 451–463.

[12] M.-S. Chen, J. S. Park, and P. S. Yu, "Data mining for path traversal patterns in a web environment," in *Proc.* 16th *International Conference on Distributed Computing Systems*, 1996, pp. 385–392.

[13] R. Kothari, P. Mittal, V. Jain, and M. Mohania, "On using page cooccurrences for computing clickstream similarity," in *Proceedings of the SIAM Conference on Data Mining*, 2003.

[14] Z. Huang, J. Ng, D. W. Cheung, M. Ng, and W.-K. Ching, "A cube model for web access sessions and cluster analysis," in *Proc. WEBKDD*, 2001.

[15] O. R. Zaïane, M. Xin, and J. Han, "Discovering web access patterns and trends by applying OLAP and data mining technology on web logs," in *Advances in Digital Libraries*, 1998, pp. 19–29.

[16] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge And Data Engineering*, vol. 8, pp. 866–883, 1996.

[17] Han J, Y. Cai, and N. Cercone, "Data-driven discovery of quantitative rules in relational databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, pp. 29–40, 1993.

[18] J. Han and Y. Fu, "Exploration of the power of attribute oriented induction in data mining," in *Advances in Knowledge Discovery and Data Mining*. 1996, pp. 399–421, AAAI/MIT Press.

[19] S. Schechter, M. Krishnan, and M. D. Smith, "Using path profiles to predict HTTP requests," in *Proc.* 7th *International World Wide Web (WWW) Conference*, 1998.

[20] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *Computer Communication Review*, vol. 26, pp. 22–36, 1996.

[21] A. Bestravos, "Using speculation to reduce server load and service time on the WWW," in *Proc. ACM Conference on Information and Knowledge Management (CIKM)*, Baltimore, MD, 1995.

[22] J. Pitkow and P. Pirolli, "Mining longest repeating subsequences to predict World Wide Web surfing," in *Proc.* 2nd *USENIX Symposium on Internet Technologies & Systems (USITS)*, Boulder, CO, 1999.

[23] J. Borges and M. Levene, "A fine grained heuristic to capture web navigation patterns," *ACM SIGKDD Explorations*, vol. 2, no. 1, pp. 40–50, 2000.

[24] B. D. Davison, *The Design And Evaluation Of Web Prefetching and Caching Techniques*, Ph.D. thesis, Department of CS, Rutgers University, 2002.

[25] W. Lin, S. Alvarez, and C. Ruiz, "Efficient adaptive-support association rule mining for recommender systems," *Data Mining and Knowledge Discovery*, vol. 6, pp. 83–105, 2002.

[26] B. Mobasher, N. Jain, E.-H. Han, and J. Srivastava, "Web mining: Pattern discovery from World Wide Web transactions," Tech. Rep., Department of CS, University of Minnesota, Minneapolis, 1996.

[27] M. Spiliopoulou and L. C. Faulstich, "WUM: a Web Utilization Miner," in *Workshop on the Web and Data Bases (WebDB98)*, 1998, pp. 109–115.

[28] F. Masseglia, P. Poncelet, and M. Teisseire, "Using data mining techniques on web access logs to dynamically improve hypertext structure," *ACM SigWeb Letters*, vol. 8, no. 3, pp. 1–19, 1999.

[29] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, Eds. 1994, pp. 487–499, Morgan Kaufmann.

[30] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[31] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Manteo, California, 1993.

[32] M. Dong and R. Kothari, "Look-ahead based fuzzy decision tree induction," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 461–468, 2001.

[33] R. Srikant and R. Agrawal, "Mining generalized association rules," *Future Generation Computer Systems*, vol. 13, no. 2–3, pp. 161–180, 1997.

[34] J.-S. Park, M.-S. Chen, and P. S. Yu, "An effective hash based algorithm for mining association rules," in *Proc. ACM SIGMOD*, 1995, pp. 175–186.

[35] D. W. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintainence of discovered association rules in large databases: An incremental updating technique," in *Proc. Interantional Conference on Data Engineering ICDE*, 1996.

[36] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[37] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 2001.

[38] E. Ukkonen, "On approximate string matching," in *Proc. International Conference on Foundations of Computational Theory (LNCS 158)*. 1983, pp. 487–495, Springer-Verlag.

[39] E. Ukkonen, "Algorithms for approximate string matching," *Information and Control*, vol. 64, pp. 100–118, 1985.

[40] F. Y. L. Chin and C. K. Poon, "A fast algorithm for computing longest common subsequences of small alphabet size," *Journal of Information Processing*, vol. 13, no. 4, pp. 463–469, 1990.

[41] W. J. Masek and M. S. Paterson, "A faster algorithm for computing string edit distances," *Journal of Computer and System Sciences*, vol. 20, no. 1, pp. 18–31, 1980.

[42] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, USA, 1988.

[43] A. Banerjee and J. Ghosh, "Clickstream clustering using weighted longest common subsequences," in *Proc. Web Mining Workshop at the* 1st *SIAM Conference on Data Mining*, Chicago, USA, 2001.

[44] J. Moore, E.-S. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher, "Web page categorization and feature selection using association rule and principal component clustering," in *Proc.* 7th *Workshop on Information Technologies and Systems*, 1997.

[45] E.-S. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering based on association rule hypergraphs," in *SIGMOD'97 Workshop on Research Issues in Data Mining and Knowledge Discovery*. 1997, pp. 9–13, ACM Press.

[46] G. Karypis, R. Aggarwal, V. Kumar, and S. Sekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," *Proc. ACM/IEEE Design Automation Conference*, 1997.

[47] R. Kothari and S. Swaminathan, "On minor component and active learning," in *Intelligent Engineering Systems Through Artificial Neural Networks*, C. Dagli, M. Akay, O. Ersoy, B. Fernandez, and A. Smith, Eds. 1997, vol. 7, pp. 93–98, ASME Press.

[48] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, Letchworth, UK, 1983.

[49] T. Balachander and R. Kothari, "Introducing locality and softness in subspace classification," *Pattern Analysis and Applications*, vol. 2, no. 1, pp. 53–58, 1999.

[50] T. Balachander and R. Kothari, "Oriented soft localized subspace classification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1999, pp. 1017–1020.