published as: L. Wang, Christoph Meinel: Recovering Individual Access Behaviors from Web Logs;
In Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE 2005); Taipei (China), pp.590-595.

Recovering Individual Accessing Behaviour from Web Logs

Long Wang[1,2], Christoph Meinel[2]
[1] Computer Science Department, Trier University
Campus II, 54296 Trier, Germany
long.wang@hpi.uni-potsdam.de
[2] Hasso Plattner Institut, Potsdam University
14482 Potsdam, Germany
Meinel@hpi.uni-potsdam.de

Abstract

In this paper, we present a new view on the data preparation in web usage mining. We concentrate on recovering individual usage behaviour from accessing records on web site. We defined five categories of individual behaviours such as granular accessing behaviour, linear sequential behaviour, tree structure behaviour, acyclic routing behaviour and cyclic routing behaviour. The algorithms for recovering different behaviours were also introduced. And the final experimental studies show that our recovery of individual behaviour is very useful and necessary in web usage mining.

1. Introduction

Tracking the traversal of different visitors through the internet is the main aim in web usage analysis. Different web service providers use different methods to record the tracks of their visitors. But for most of the rest web sites which are published for governments, educations, companies and personals, all the usage data are recorded by web servers as web logs or called click-streams in a timestamp order, and in this case, the task to discriminate visitors and their behaviours and interests becomes much harder.

Web usage mining aims to find the characteristic usage patterns in web environment. The traditional mining tools such as association rules, sequential patterns, and classification are necessary to find the usage patterns, but not enough to depict the web characteristics revealed from visitors accessing, such as revisiting and routing. In [6], the maximal forward reference is formed from the start of an access till the occurring of revisiting a previously visited object by the same access. In [1], a web access pattern is the sequence of accesses pursued frequently by many visitors in which repeated pages could happen. The similar definitions can be found in [3]. Each of these efforts is dedicated to define and mine only one type of usage patterns. But the valuable information hidden in web logs is far more than the above defined usage patterns. An individual accessing behaviour not only refers to the contents that were visited, but also the way that the same access performed. The former reveals the visitor's interest on the content and the latter reveals the visiting custom of this visitor, and also reveals the site structure or semantics relations among these visited objects. To get the useful usage pattern that reveals the web characteristics, it is necessary to investigate individual accessing. The aim of recovering individual accessing behaviour is to reconstruct the browsing scenario, which is the necessary foundation to get the useful and meaningful usage patterns from all the accessing behaviours. This task is the further integration and reorganization of individual usage data and it outputs the exact single user data for the further usage mining and knowledge discovery.

In this paper, we give a detailed view on recovering individual accessing behaviour. Based on the target mined patterns, an individual accessing behaviour can be recovered into five different categories: granular accessing behaviour, linear sequential behaviour, tree structure behaviour, acyclic behaviour and cyclic routing behaviour.

The rest of the paper is organized as follows. Section 2 gives the context and the data preparation for "recovering individual accessing behaviour". Section 3 presents the necessary terminology and formally defines this problem. We give the algorithms of recovering individual behaviours in section 4 and analyze our experiment results in section 5. Section 6 provides a conclusion and overview on the future works.

2. Related Works

In [8], Cooley gave an overview on data preparation for web usage mining. In their work, they firstly remove irrelevant items such as scripts and attached files from web logs, and then a single visitor can be identified by IP address, client information and even the direct links from any of the objects visited by the same IP and client information. Two timeouts are used to identify individual sessions from all the objects accessed by each visitor. The web objects (pages) are classified into content page and auxiliary page, and user browsing model is represented by a page sequence. While in [5], proactive strategies like cookie-based identification was used to reconstruct session.

In our experiments, we refereed the method from [8] to identify different visitors. A visitor can be identified by a triple unit <IP Address, Client OS, and Client Browser>. And we use two timeout thresholds to identify different sessions from all the objects accessed by the same visitor. In [5], it is showed that there is no best method for session reconstruction. The problem of session identification is beyond our discussion in this paper, our contribution concerns on recovering individual accessing behaviour from reconstructed session.

Before accessing behaviour recovering, unrelated information should be further removed from sessions. If an object was accessed by the same visitor within a session, we omit the second happening for behaviour recovering. The reason for the continuously revisit of the same object is mainly due to "reloading or refreshing the same object" or "the existence of hyperlink to itself".

3. Problem Statements

Let W be the target web site that gives us the web usage logs. After removing the errors and useless information from logs, we take some part of cleaned logs as our target denoted as L. Visitors, Times and Objects are the three key parameters in web usage mining. Let V be the set of all visitors identified from L and T be the set of all time requests recorded in L, and O be the set of possible object requests in W. An object is a page, media file or a visitor's action captured by the web server.

L is a list of actual object requests from V, and each of these object requests is recorded as a logline entry termed as l. So L can be regarded as a set, but all the loglines are ordered by the timestamp of their invocation. We use $l.visitor$ to denote the visitor in the logline $l \in L$, and $l.time$ to denote the request time in l, and $l.url$ to denote the URL request in l and $l.obj$ to denote the object requested by $l.visitor$. It is obviously that for each logline l, $l.visitor \in V$, $l.time \in T$, and $l.obj \in O$.

L is denoted as:

$$L = \{l_1, l_2, \ldots, l_n\},\ 1 \le i < j \le n,\ l_i.time < l_j.time.$$

A session s is denoted:

$$s = \{l_1, l_2, \ldots, l_m\},\ 1 \le i < j \le m,$$
$$l_i.visitor = l_j.visitor,\ l_i.time < l_j.time.$$

This means that a session s preserves the same order of requests as in L, and s satisfies the following conditions:

$$\text{For } 1 \le i < i+1 \le m,\ l_{i+1}.time - l_i.time \le Timeout1,$$
$$l_m.time - l_1.time \le Timeout2.$$

Timeout1 and *Timeout2* are the two time thresholds needed to identify sessions. As the same definition for logline l, we also denote *s.visitor* the visitor of this session, and *s.length* the number of objects in s.

With the above definition, we can map web logs L into a session set S, for each logline $l \in L$, l belongs to exactly one session, and this ensures that S partitions L in an order-preserving way.

Additionally, a hyperlink within a web site is a binary relation on the object set O. We define *Hyperlink (O)* as the set of all the hyperlinks among the elements in O, and as well, *hyperlink* (obj_i, obj_j) as the link from object o_i to o_j:

$$Hyperlink\ (O) \subseteq O \times O,$$
$$\forall\ hyperlink(o_i, o_j) \in Hyperlink\ (O)\text{: there is a hyperlink from } o_i \text{ to } o_j.$$

We give a function to get the i^{th} accessed object in a session:

$$Object(s, i) = l_i.obj.\ (*)$$

Given a session s, we define some functions on it:

(1) The firstly accessed object in a session:

$$EntranceObject(s) = l_1.obj.$$

(2) The last accessed object in a session:

$$ExistObject(s) = l_{s.length}.obj.$$

We call the object obj_j "*target object*" of object obj_i and obj_i "*source object*" of obj_j, if obj_j is accessed after obj_i. And we also call the last accessed object "*final target object*" of a session.

(3) The set of repeated objects in a session:

$$Repeated(s) = \{obj_1, \ldots, obj_k\},$$
$$\forall i (1 \le i \le k),\ \exists i', j' (1 \le i' <> j' \le s.length.)\text{: } Object(s, i') = Object(s, j') = obj_i.$$

(4) The set of access objects in a session:

$$ObjectSet(s) = \{obj_1, \ldots, obj_k\},$$
$$\forall\ i, j (1 \le i <> j \le k),\ \exists i', j' (1 \le i' <> j' \le s.length.)\text{: } Object(s, i') = obj_i,\ Object(s, j') = obj_j,\ obj_i <> obj_j.$$

(5) An access sequence in a session:

$$Sequence(s) = obj_1\ obj_2 \ldots obj_k,$$

$\forall i, j(1 \le i < j \le k)$, $\exists i', j'(1 \le i' < j' \le s.length.)$:
$Object(s, i') = obj_i$, $object(s, j') = obj_j$, $obj_i <> obj_j$.

(6) An access path in a session:

$$Path(s) = obj_1\, obj_2 \ldots obj_k,$$
$$\exists\, i'(1 \le i' < s.length),\ \forall\, i, j(1 \le i <> j \le k.):$$
$$Object(s, i + i') = obj_i,$$
$$Object(s, j + i') = obj_j,\ obj_i <> obj_j.$$

The difference between accessed sequence (5) and path (6) is that all the objects in a sequence are accessed in the same time order as in the original session, while in a path, all the objects must be *continuously* accessed one by one as the order in the session. So access path can be seen as the special access sequence. The lengths of accessed sequence and path are defined as the same as the length of a session, and *q.length* and *p.length* are used for these two lengths. And also the definitions for the i^{th} object in a sequence and a path are the same as in (*), which only the parameter "*s*" is replaced by "*q*" or "*p*".

Access sequence $Seq' = obj'_1\, obj'_2 \ldots obj'_k$ is called a subsequence of access sequence $Seq = obj_1\, obj_2 \ldots obj_n$ and Seq a super-sequence of Seq', denoted as $Seq' \subseteq Seq$, if and only if there exist $1 \le i_1 < i_2 < \ldots < i_k \le n$, such that $obj'_j = obj_{ij}$ for $(1 \le j \le k)$. Access path $P' = obj'_1\, obj'_2 \ldots obj'_k$ is called a sub path of access path $P = Obj_1\, obj_2 \ldots obj_n$ and P a super-path of P', denoted as $P' \subseteq P$, if and only if there exist a const c, such that $obj'_j = obj_{j+c}$ for $(1 \le j \le k)$.

(7) A path with hyperlinks in a session:

$$LinkedPath(s) = obj_1, \ldots, obj_k,$$
$$\exists\, i'(1 \le i' < s.length),\ \forall\, i(1 \le i < i+1 \le k.):$$
$$Object(s, i + i') = obj_i,\ Object(s, i + i+1') = obj_{i+1},$$
$$hyperlink(obj_i, obj_{i+1}) \in Hyperlink(O).$$

(8) A path without hyperlinks in a session:

$$UnlinkedPath(s) = obj_1, \ldots, obj_k,$$
$$\exists\, i'(1 \le i' < s.length),\ \forall\, i(1 \le i < i+1 \le k.):$$
$$Object(s, i + i') = obj_i,\ Object(s, i + i+1') = obj_{i+1},$$
$$hyperlink(obj_i, obj_{i+1}) \notin Hyperlink(O).$$

From the above defined functions, we can find some accessing behaviour revealed from a session. For example, the repeated accessed objects may play great importance in deciding the visitor's routing from web structure or semantic level. And also the unlinked accessed object sequence attracts us to find how and why the visitor suddenly jumps to another unlinked object.

It is well acknowledged that a web site is complex graph, and any kind of information can find its position in this graph. In our study, we take the objects accessed by visitors as the basic unit, then these objects are the vertices and the hyperlinks are edges in this graph. The accessing of every visitor is a directed routing process of the sub graph. Regardless of the repeat and backwards tracking, we only care about the relationships among the accessed objects in a session, so maybe tree structure, undirected and directed graph relationships are hidden in a session. From the point of visitor tracking, the tree structure relationship is characterized by the nodes that lead to different objects in a session, which mean divert or different paths after an object. Thus we call this tree structure relationship "divert path tracking" in web usage. Furthermore, in the routing on a graph, it is popular that there is more than one path between two selected vertices, which looks like a rhombus structure. And we call this relationship "parallel path tracking" in web usage.

In the following, we give the formally definitions of "circle path", "divert path tracking" and "parallel path tracking" in web usages.

(9) A circle path in a session:

$$CirclePath(s) = obj_1, \ldots, obj_k,$$
$$\exists\, i'(1 \le i' < s.length),\ \forall\, i(1 \le i \le k.):$$
$$Object(s, i + i') = obj_i,\ obj_1 = obj_k.$$

(10) The diverged paths in a session:

$$DivertPath(s) = \{Path(s)_1, \ldots, Path(s)_k\},$$
$$\forall\, i, j(1 \le i <> j \le k.):\ Object(P_i, 1) = Object(P_j, 1).$$

(11) The parallel paths in a session:

$$ParallelPath(s) = \{Path(s)_1, \ldots, Path(s)_k\},$$
$$\forall\, i, j(1 \le i <> j \le k.):\ Object(P_i, 1) = Object(P_j, 1),$$
$$Object(P_i, P_i.length) = Object(P_j, P_j.length).$$

The above definitions reveal the diversities of the usage activities endowed with the special characteristics in web environment, such as entrance page, hyperlinks, backtracking, revisiting and so on. And it is possible for an individual accessing session to be interpreted with one of these definitions or the combination of several definitions, which is far more than object sets and sequences patterns as discussed in [2]. To mine the concrete and meaningful usage patterns among groups of visitors, it is necessary to investigate the activities of single visitors. We use a plain term "*Action*" to uniform these 11 different basic functions, for each of them characterizes the unique activity perfumed by a visitor on some objects in a session. So an action of a visitor is embodied with the organization of some objects in a session.

We now give the definition of individual accessing behaviour: An individual accessing behaviour is the combination of several actions performed by a visitor during his session with the web server.

4. Recovery Algorithms

An individual accessing behaviour is the combination of actions extracted from a session, which reveal not only the required objects during his visiting, but also some of site structure and concept hierarchies, and also the routing activities on these structures and hierarchies characterized with revisiting, back tracking and so on.

Individual accessing behaviour can be recovered using several recovery techniques. The choice for proper recovering method is decided by what kind of access patterns we want to mine in the following step, and the recovered individual access behaviour is characterized by some actions illustrated in the former sections. From simple to complex, we show here some strategies of behaviours recovery. We illustrate this problem from a real session reconstructed from our server logs. The objects we investigate in our study are the web pages, so in the rest of the paper, when our statement is related with our example, we replace the term "object" with "page", and also for simplicity, every page is titled with its ID.

This session is listed as following:

$$s = <0, 292, 300, 304, 350, 326, 512, 510, 513, 512, 515, 513, 292, 319, 350, 517, 286>$$

There are 17 page/times requests in this session from its visitor. 0 and 286 were accessed separately as entrance and leaving pages. And there are two kinds of pages, one group includes 300, 304, 326, 510, 319, 517, and 515, which were accessed only once; and the other includes 292, 350, 513 and 512, which were accessed more than once:

$$Repeated(s) = \{292, 350, 513, 512\}.$$

4.1 Simple behaviours recovery

This strategy overlooks all the repeated pages in a session. The behaviour of this visitor can be simply recovered into the largest set of accessed objects, or the longest access sequence. We call the largest set of accessed objects "granular behaviour" and longest access sequence "linear sequential behaviour". The former is defined as $ObjectSet_L(s)$ and the latter is $Sequence_L(s)$.

For the above session, the granular behaviour is recovered as:

$$ObjectSet_L(s) = \{0, 292, 300, 304, 350, 326, 512, 510, 513, 515, 319, 517, 286\}.$$

We can see that any sub set of $ObjectSet_L(s)$ is one of the sets of accessed objects by this visitor.

To recover the longest accessed sequence, we choose the first request time as the time for the same repeated pages in a session. So we remove the 10^{th}, 12^{th}, 13^{th}, and 15^{th} pages in the above session.

The linear sequential behaviour is recovered as:

$$Sequence_L(s) = <0 – 292 – 300 – 304 – 350 – 326 – 512 – 510 – 513 – 515 – 319 – 517 - 286>.$$

Any subsequence of the longest accessed sequence is one of the accessed sequences in this session.

Motivated by other data mining applications, given a large group of different sets of accessed objects and accessed sequences by different session, we can mine the most popular set of accessed objects and the most accessed sequence.

4.2 Tree structure behaviours recovery

The tree structure behaviour is characterized by diverged paths in a session defined in (10). Some paths in a session can form diverged path because they share the start accessed object, which means that an object can attract visitor to different targets. Tree structure behaviours not only depicted the visiting patterns, but also revealed some conceptual hierarchy on site semantics.

To recover access tree t from session s, we used a page set named P to store the unique pages that already exist in t, and we also used a pointer pr pointing to the last recovered node in t. The recovery strategy is as:

*1) Set t = **NULL**;*

2) Read the first entrance page in s as the tree root r, let pr pointing to r and insert this page to P;

3) Read new page from s and judge if the same page exist in P;

i) Exist in P,

4) Find this already existing node n in t and set pr point to this node,

5) Go to step 3.

ii) Not exist in P,

4) Insert this new page to P,

5) Create a new node and insert this new node as a new child for pr,

6) Let pr point to this new node,

7) Go to step 3.

The tree structure behaviours for the above session is recovered by our strategy as the following figure:

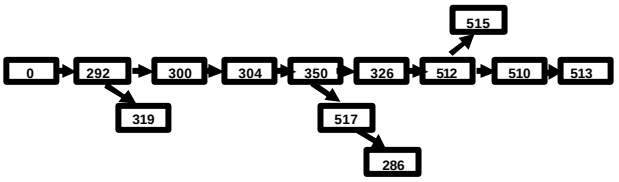

Figure 1: Tree Structure behaviour

Based on this algorithm, there is some property in the recovered tree structure behaviours:

Property 1: *Given recovered tree structure behaviour, the nodes that lead to diverged paths are the repeated objects in this session.*

The diverged path in this session is:

$DivertPath_1(s) = \{<292\text{-}300\text{-}304\text{-}350>, <292\text{-}319>\}$,
$DivertPath_2(s) = \{<350\text{-}326\text{-}512><350\text{-}517\text{-}286>\}$,
$DivertPath_3(s) = \{<512\text{-}510\text{-}513>, <512\text{-}515>\}$.

Tree structure behaviours can help to mine those access patterns with tree structure [7] and to mine the large reference sequences from maxim forward references [6].

4.3 Acyclic Routing behaviours recovery

"Acyclic routing behaviour" means that in a session, there exist at least two different pages between which there were at least two different access paths. This kind of behaviour is characterized by the parallel paths in a session. It shows that visitor can access the same target object from the same start object but via different paths. With acyclic routing behaviours, we can further query the shortest path and most popular path between two pages. We also call this recovered behaviour "semi-lattice behaviour".

The final recovered behaviour is like a lattice structure defined as l, and we used P to store unique pages in l, and pr pointing to the last recovered node in l. We used the following strategy to rebuild the acyclic routing in a session.

*1) Set l = **NULL**;*
2) Read the first entrance page in s as the top node t, led pr pointing to t and insert this page to P,
3) Read new page from s and judge if the same page exist in P;
i) Exist in P:
4) Find this same existing node n in l and judge the relation between n and pr,
a) n can be backward tracked from pr
5) Set pr point to n,
6) Go to step 3.
b) n can be forward tracked from pr
5) Build new edge directed from pr to n, if there is not directed edge from pr to n.
6) Set pr point to n,
7) Go to step 3.
c) n can not be tracked from pr in a single direction
5) Build new edge directed from pr to n,
6) Set pr point to n,
7) Go to step 3.
ii) Not exist in P:
4) Insert this new page to P,
5) Create a new node and insert this new node as a new child for pr,
6) Let pr point to this new node,
7) Go to step 3.

The following figure displays the recovered acyclic routing behaviour from the above session:

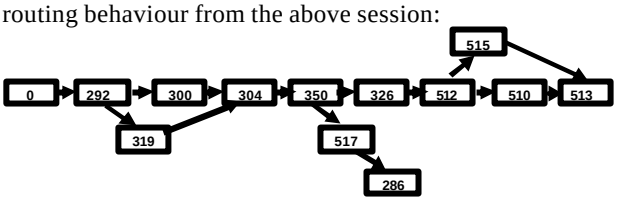

Figure 2: Acyclic routing behaviour

It is clear that if an acyclic routing behaviour can be recovered from a session, the session must have the following property:

Property 2: *An acyclic routing behaviour can be recovered from a session s iff there exist obj_i, obj_m, obj_j, obj_v, obj_k, obj_w ($1 \le i < m < j < v < k < w \le s.length$) in s, and $obj_i == obj_v$; $obj_j == obj_w$; $obj_m <> obj_k$.*

The parallel path in this session is:

$ParallelPath_1(s) = \{<292\text{-}300\text{-}304\text{-}350>, <292\text{-}319\text{-}350>\}$,
$ParallelPath_2(s) = \{<512\text{-}515\text{-}513>, <512\text{-}510\text{-}513>\}$.

4.4 Cyclic routing behaviours recovery

Within a session, different accessed objects are the targets chosen by this visitor, and they are linked by the accessing sequence, which forms a directed graph. If there is back tracked or revisited objects in a session, a directed link will be built from the target object to one of its source object. From the semantic level, we call these two object can be mutually heuristically evoked. In this meaning, the individual behaviour can be recovered as cyclic routing behaviour and such behaviour is characterized by the circle path hidden in the session. The strategy is similar to but more complicate than recovering acyclic routing.

The following figure shows cyclic routing behaviours recovered from the same example.

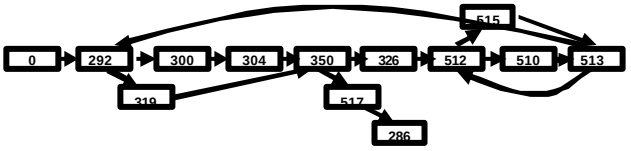

Figure 3: Cyclic routing behaviour

The circle path in this session is:

$CirclePath_1(s) = 292\text{-}300\text{-}304\text{-}350\text{-}326\text{-}512\text{-}510\text{-}513\text{-}292$,

$CirclPath_2(s) = 512\text{-}510\text{-}513\text{-}512$.

5. Experiment Results

Our experiment data was taken from three web sites: www.informatik.uni-trier.de (INFO), www.hpi.uni-potsdam.de (HPI) and www.tele-task.de (TTK). These three sites are chosen because of their differences: INFO is a well frame based site, and TTK is a site dedicated to multimedia lectures and HPI has launched a new version. The time durations for these logs are one month for INFO and HPI, and 12 months for TTK because of the small access count.

	INFO	HPI	TTK
Pages	728	253	52
Sessions	5828	28308	33894

Table 1: Pages and Sessions on INFO, HPI and TTK

The average of the length of session is 3~4. It has been discussed in the previous parts that for every session a granular behaviour and linear sequence behaviour can be recovered, but for tree structure behaviour there must be repeated objects in a session, and for semi-lattice structure behaviour, two or more different objects must be repeated. We compute the number of the sessions with 1 or 2 repeated pages, and also the number of the sessions that can recover tree and semi-lattice behaviours. We also give the ratio of these numbers with respect to the session set. The following table gives the statistic results.

	INFO	HPI	TTK
Sessions with 1 R-Page (ratio)	701 (~12%)	3998 (~14.1%)	1319 (~3.9%)
Sessions with 2 R-Pages (ratio)	350 (~6%)	2493 (~8.8%)	237 (~0.7%)
Sessions with T-Behaviour (ratio)	642 (~11%)	3911 (~13.8%)	1085 (~3.2%)
Sessions with L-Behaviour (ratio)	72 (~1.2%)	402 (~1.4%)	22 (~0.06%)

Table 2: Statistics of Repeated Pages and Recovered Behaviours

From the *table2*, we found that tree behaviours and semi-lattice behaviours exist in some of the sessions, and their hosts are the most valuable visitors for the sites. And the visitor behaviour is closely related with site structure and content. Firstly, the more complex of the web site, the more complex of the behaviour; secondly, most of the sessions with repeated pages can recover tree behaviour, but great drawdown of semi-lattice behaviour from sessions with 2 or more pages, which is due to the constraints among objects in a lattice behaviour. And also, HPI has a much smaller link depth than INFO, so in the lattice behaviour happens frequently than in INFO.

6. Conclusion

The bottleneck of enlarging the mining applications in web usage field is the exploding of web knowledge and the specialities of web environment, which attract us to deeply investigate the individual access behaviour.

In this paper, we discuss the individual access behaviour through the web, and how to recover these behaviours from web logs. The complexity of web structure and the variety of visitors, and also the target patterns pursued decide that access behaviours can not be simplified into one category, and we define five different categories of individual access behaviour. Before these behaviours were given, we also define 11 different basic actions that could be performed during a session. And individual access behaviour is the combination of these basic actions. The experiment results show that our defined actions and behaviour universally exist in many websites. And they are the necessary for mining the useful usage patterns with web characteristics in the following mining steps.

References

[1] Bettina Berendt, Myra Spiliopoulou: Analysis of navigation behaviour in web sites integrating multiple information systems. The VLDB Journal, (2000)

[2] J. Srivastava, R. Cooley, M. Deshpande and P. Tan: Web Usage Mining: Discovery and Application of Usage Patterns from Web Data, ACM SIGKDD, (2000)

[3] Jian Pei, Jiawei Han and etc.: Mining Access Patterns Efficiently from Web Logs, PAKDD, (2000)

[4] Jian Pei, Jiawei Han and Wei Wang: Mining Sequential Patterns with Constraints in Large Databases, ACM CIKM, (2002)

[5] M. Spiliopoulou, B. Mobasher, B. Berendt and M. Nakagawa: A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis, INFORMS, (2000)

[6] Ming-Syan Chen, Jong Soo Park and etc.: Data Mining for Path Traversal Patterns in a Web Environment. Proceedings of the 16th International Conference on Distributed Computing Systems (1996)

[7] Mohammed J. Zaki: Efficiently Mining Frequent Trees in a Forest. In SIGKDD'02 (2002)

[8] R. Cooley, B. Mobasher and J. Srivastava: Data Preparation for Mining World Wide Web browsing Patterns, Knowledge and Information System, (1999)