published as: L. Wang, Christoph Meinel: Discovering Characteristic Individual Access Behaviors in Web Environment;
Proceedings of 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC 2005); Regina (Canada), LNAI 3642, pp.466-476.

Discovering Characteristic Individual Accessing Behaviors in Web Environment

Long Wang[1,2] Christoph Meinel[2] and Chunnian Liu[3]

[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
[3] Beijing Municipal Key Lab. of Multimedia and Intelligent Software Technology, Beijing University of Technology, 100022 Beijing, China
cslcn@but.edu.cn

Abstract. Discovering diverse individual accessing behaviors in web environment is required before mining the valuable patterns from behaviors of groups of visitors. In this paper, we investigate the data preparation in web usage mining, and especially focus on discovering characteristic individual accessing behaviors and give a systematic and formalized study on this topic. Based on the target usage patterns, individual user behavior through the web site can be discovered into five different categories: granular accessing behavior, linear sequential behavior, tree structure behavior, acyclic routing behavior and cyclic routing behavior. We also give different algorithms for discovering different kinds of behaviors. The experimental studies show that our discovery of individual behavior is very useful and necessary in web usage mining.

1 Introduction

It is necessary to clean and collect usage data for different visitors before mining usage pattern in web environment. Different web service providers use different methods to record the tracks of their visitors, such as web server logs, cookies functions, personalized agents [10] or other interactive scripts. Traditional mining tools such as association rules, sequential patterns, and classification help much to find the usage patterns, but there are some other typical web characteristics that could be revealed from visitors accessing by these tools, such as revisiting and routing.

Many other experts have made great works on mining characteristic patterns in web environment. In [9], the user browsing model is built based on classifying the objects (pages) into content page and auxiliary page. The format of user browsing model is represented by a sequence, so the main contribution of such representation is to mine the usage patterns as association rules and frequent traversal paths. In [7], a maximal forward reference is formed from the start of an access till the occurring of revisiting a previously visited object by the same access. And the large reference sequences are used to depict the path traversal patterns and mined from maximal forward references. In [1,4], web access pattern

D. Ślęzak et al. (Eds.): RSFDGrC 2005, LNAI 3642, pp. 466–476, 2005.

is the sequence of accesses pursued frequently by many visitors which may exist repeated objects.

Characteristic usage patterns are mined from the personalized individual accessing behaviors, so it is necessary to investigate individual accessings, which is far more than the above defined usage patterns. In this paper, we give a new view on discovering individual accessing behavior in web environment. An individual accessing behavior is beyond the visited objects and also the routing process among these objects. Based on the target mined patterns, an individual accessing behavior can be discovered into five different categories: granular accessing behavior, linear sequential behavior, tree structure behavior, acyclic behavior and cyclic routing behavior. In these methods, discovering individual behaviors are combined in the process of session reconstruction.

Individual accessing behaviors are discovered from reconstructed sessions. A session is defined as a group of requests made by a single user for a single navigation purpose [11]. This definition gives a better illustration on "session", but "session" is firstly refereed to a time concept, which means that a session must have happened within a couple of time boundaries.

There are different methods to rebuild sessions from web logs. In [2], timeout method is used to identify different session. In this method, a session shift is identified between two requests if the time interval between two requests is more than a predefined threshold. In [9], 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. And two timeouts are used to identify individual sessions from all the objects accessed by each visitor. While in [6], proactive strategies like cookie-based identification were used to reconstruct session. A method named maximal forward reference is used in [7]. And in [11], Zhang used a statistical language modelling to identify sessions from web logs.

In [6], it is showed that there is no best method for session reconstruction. In our experiments, we refereed the method from [9] to identify different visitors and different sessions. The problem of session identification is beyond our discussion in this paper, our contribution concerns on recovering individual accessing behaviors from reconstructed sessions. In this paper, our work is done based on the assuming that the sessions are already identified.

Different from [9] in session reconstruction, we don't care if there is hyperlink between two successively accessed objects, because it is possible for a visitor to use bookmark or URL hints cached by browser within a session. And also unrelated information should be removed from sessions. If an object was accessed *continuously* within a session, we omit the repeat happenings. The reason for the continuously repeat happenings of the same object is mainly due to "reloading or refreshing the same object" or "the existence of hyperlink to itself".

The rest of the paper is organized as follows. In section 2, we present the necessary terminology and formally define individual behavior. Section 3 gives a detailed view on the algorithms of discovering individual accessing behavior. We analysis our experiment results on three different kind of web sites in section 4. Section 5 provides a conclusion and overview on the future works.

2 Problem Statements

Let W be the target web site that gives us the web usage logs, and L be the target cleaned logs. 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 of the logline $l \in L$, and $l.time$ to denote the request time of l, and l.url to denote the URL request in l and $l.obj$ to denote the object requested by $l.visitor$. It is obvious that for each logline l, $l.visitor \in V$, $l.time \in T$, and $l.obj \in O$.

A session s is denoted:

$$s = \{l_1, \ldots, l_m\} : l \leq i < j \leq m, l_i.visitor = l_j.visitor, l_i.time < l_j.time.$$

And also s should satisfy the following conditions:

$$\forall i(1 \leq i < m) : l_{i+1}.time - l_i.time \leq Timeout1, l_m.time - l_1.time \leq Timeout2.$$

$Timeout1$ and $Timeout2$ are the two time thresholds needed to identify sessions, and $Timeout1$ is the threshold that constraints the time delay between two continuously accessed objects and $Timeout2$ aims to constraint the time duration of a session. 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 $linL$, l belongs to exactly one session, and this ensures that S partitions L in an order-preserving way.

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

$$Object(s, j) = l_i.obj : l_i ins.(*)$$

(1) The firstly accessed object in a session s:

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

(2) The last accessed object in a session s:

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

We call the object obj_j "*target object*" of object obj_i and obj_j "*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 s:

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

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

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

(5) An access sequence in a session s:

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

(6) An access path in a session s:

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

From the definition (5) and (6), the difference between accessed sequence and path 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. Even with removing continuously repeatedly accessed objects in the raw session, it doesn't affect the final results of finding sequences and paths in a session.

We use $q.length$ and $p.length$ to denote the lengths of a sequence and a path. And 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 $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, such that $obj'_j = obj_{i_j}$ for $(1 \leq j \leq k)$. 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 $(1 \leq j \leq k)$.

It is well acknowledged that a web site is a 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 among them are edges in this graph. The relationships among the accessed objects in a session, such as tree structure, undirected and directed graph, are hidden in a session which is a form of sequence. 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.

(7) A circle path in a session s:

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

(8) The diverged paths in a session s:

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

(9)The parallel paths in a session s:

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

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. 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 [4,9]. We use a plain term "Action" to uniform these 9 different basic functions, because 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 behavior: *An individual accessing behavior is the combination of several actions performed by a visitor during his session with the web server.*

3 Discovery Algorithms

An individual accessing behavior is the combination of actions extracted from a session and it displays not only the accessed objects and part of site structure, but also the concept hierarchies and the routing activities on these objects.

Individual accessing behavior can be discovered by using several recovery techniques. The choice of proper discovering method is decided by what kind of access patterns we want to mine in the next step. From simple to complex, we show here some strategies of behaviors discovery. We illustrate this problem by using a real session reconstructed from our server logs. The objects here are the web pages, so in the rest of the paper, we replace the term "object" with "page", and every page is titled with its ID. This session is listed as following:

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

0 and 286 were accessed separately as entrance and leaving pages, and $Repeated(s) = \{292, 350, 513, 512\}$. Any piece of session without repeated pages can form a path, for example:

$$Path_1(s) = \text{300-304-350-326-512, and } Path_2(s) = \text{512-515-513-292.}$$

3.1 Simple Behaviors Discovery

This strategy overlooks all the repeated pages in a session. The behavior of this visitor can be simply discovered into the largest set of accessed objects, and the longest access sequence. We also call the largest set of accessed objects "granular behavior" and longest access sequence "linear sequential behavior". These two kinds of behaviors are the extensions of the definitions of "set of access objects" and "access sequence" in section 3; and the former is defined as $ObjectSet_L(s)$ and the latter is $Seq_L(s)$. To discover the longest accessed sequence, we choose the first request time as the time for the same repeated pages in a session. For the above session, we remove the 10th, 12th, 13th, and 15th pages:

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

We can see that any sub set of $ObjectSet_L(s)$ is one of the sets of accessed objects by this visitor. Any subsequence of $Seq_L(s)$ is one of the accessed sequences in this session.

Motivated by other data mining applications in [4,9], given a large group of accessed objects and accessed sequences by different sessions, we can mine the most popular set of accessed objects and the most popular accessed sequences.

3.2 Tree Structure Behaviors Discovery

The tree structure behavior is characterized by diverged paths in a session defined in (8). From this definition, some paths in a session can form diverged path because they share the start accessed object. Though all the accessed objects are ordered by timestamp in a sequence, we can find those repeated objects that lead to different target objects. Tree structure behaviors displayed not only the visiting patterns, but also some conceptual hierarchy on site semantics.

To discover tree structure t from a session s, a pointer pr is used to point to the last read node in t. Every page is read in the same order as in s and this page is inserted as the child node of pr if it firstly happens in t; but if the same page already exits in t, we do nothing but only setting pr point to this page in t.

The tree structure behaviors for the above session can be discovered with our strategy as the figure 1.

Based on this algorithm, there is some property in the discovered tree structure behaviors:

Property 1: Given a discovered tree structure behavior, the nodes that lead to diverged paths are the repeated objects in this session.

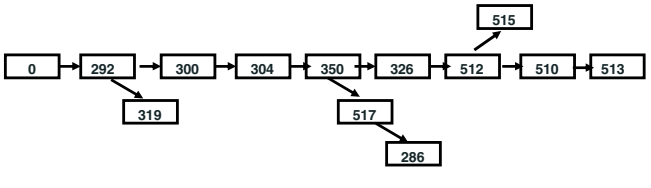

Fig. 1. Tree Structure Behavior

The diverged path in this session is:

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

Similar from [8], given a large group of discovered individual tree behaviors, the most popular tree structure access patterns can be mined.

3.3 Acyclic Routing Behaviors Discovery

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

The final discovered behavior is like a lattice structure defined as l, and we used pr pointing to the last read node in l. The pages are read in the same sequence as in s, and for every page, we check if the same page exits in l. If this page firstly happens in l, we insert this as a new child node of pr, and let pr point to this new node. If this page already exits in l, there are four different relations between this page and the last read page:

- This page is the same as pr:
  1. Do nothing.
- This page can be backward tracked from pr:
  1. Set pr point to this page.
- This page can be forward tracked from pr:
  1. Build new edge directed from pr to this page, if there is not directed edge from pr to this page,
  2. Set pr point to this page.
- This can not be tracked from pr:
  1. Build new edge directed from pr to this page,
  2. Set pr point to this page.

Figure 2 displays the recovered acyclic routing behavior from the above session. It is clear that if an acyclic routing behavior can be discovered from a session, the session must have the following property:

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

The parallel paths in this session are:

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

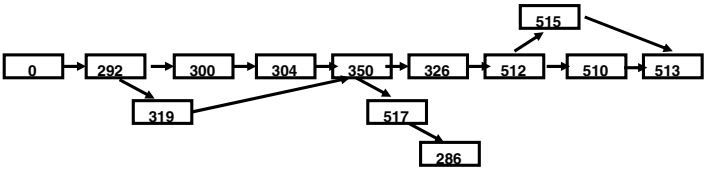

Fig. 2. Acyclic Routing Behavior

3.4 Cyclic Routing Behaviors Discovery

If there are back tracked or revisited objects in a session, directed links will be built from every revisited objects to one of its source object. From the semantic level, we call these two object can be mutually heuristically evoked. So in this situation, the individual behavior can be discovered as cyclic routing behavior and this kind of behavior is characterized by the circle path hidden in the session.

The strategy for discovering cyclic routing behavior is similar to but more complicate than discovering acyclic routing. The following figure shows cyclic routing behaviors discovered from the same example.

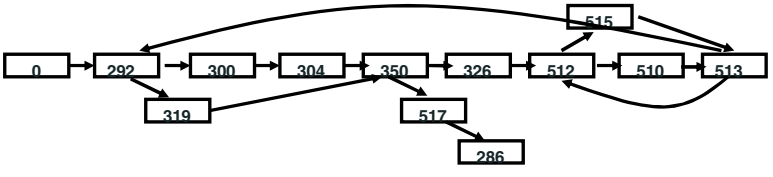

Fig. 3. Cyclic Routing Behavior

The circle paths in this session are:

$$CirclePath_1(s) = 292 - 300 - 304 - 350 - 326 - 512 - 510 - 513 - 292,$$
$$CirclePath_2(s) = 512 - 510 - 513 - 512.$$

4 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.

The distribution of repeated pages is closely related with the web structure. It is clear that a structure with many pages has more repeated pages than those with small number of pages. It can be seen from the different distribution that TTK has a small number of repeated pages than INFO and HPI. But for the ratio of repeated pages, the simple structure has a higher possibility of repeated

Table 1. Pages, Sessions and Repeated Pages on INFO, HPI and TTK

	INFO	HPI	TTK
Pages	728	253	52
Sessions	5828	28308	33894
Number of Repeated Pages(Ratio)	29(∼3.9%)	21(∼8.3%)	6(∼11.5%)

happenings than complicate structure; the reason is that a visitor could have more choices in a site with many pages which could reduce the happening of repeated visits. Another factor is related site structures, the difference of the number of repeated pages from INFO and HPI is not as so big as the number of total pages on this two sites, this is because the a big part of pages from INFO are those with few links in their contents and are seen as the bottom pages or the leaf pages on the site graph.

The average of the length of session is 3∼4. It has been discussed in the previous parts that for every session a granular behavior and linear sequence behavior can be discovered, but for tree structure behavior there must be repeated objects in a session, and for semi-lattice structure behavior, 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 discover tree and semi-lattice behaviors. We also give the ratio of these numbers with respect to the session set. The following table gives the statistic results.

Table 2. Statistics of Repeated Pages and Recovered Behaviors

	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-Behavior (ratio)	642(∼11%)	3911(∼13.8%)	1085(∼3.2%)
Sessions with L-Behavior (ratio)	72(∼1.2%)	402(∼1.4%)	22(∼0.06%)

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

Based on the discovered individual behaviors, we further mine the popular and characteristic web usage patterns. By analyzing the accessing information of entrance page and exit page, these three different sites have the same accessing distribution: the entrance pages concentrate on a very few number of pages, which have much higher popularities than other pages accessed as entrance page.

The exit pages concentrated on a number of pages, while the difference on accessing popularity is not as violently as on entrance page. This phenomenon gives us the hint that the exit page plays more important on discriminating visitors from different interests. The analysis on mining popular co-accessed pages and popular sequences from discovered simple behavior set is the same as that from other researchers, so we don't give the mined examples here. Mining complex patterns from discovered complex behaviors such as tree behaviors and semi-lattice behaviors, we build a complex tree structure named DIXT (Double Index Three-Dimension Tree) to index and agglomerate individual behaviors, and this will be detailed discussed in another paper. Based on the properties analysis in the above sections, one required conditions for recovering tree structure behavior is the existence of at least 1 repeated object in a session; and one required conditions for recovering lattice behavior is the existence of at least 2 different repeated objects in a session.

We now give some general statistics on the mined popular tree patterns and parallel patterns on these tree sites separately with different support thresholds.

Table 3. Statistic on Tree Patterns

	INFO	HPI	TTK
Sessions with T-Behavior	642	3911	1085
Tree Patterns (sup=0.005)	21	48	23
Tree Patterns (sup=0.01)	6	11	8

Table 4. Statistic on Lattice Patterns

	INFO	HPI	TTK
Sessions with L-Behavior	72	402	22
Lattice Patterns (sup=0.02)	64	7	14
Lattice Patterns (sup=0.05)	3	0	3

For further analysis, lattice behavior surely includes its corresponding tree structure if we keep only one incoming connection for a node that is repeatedly accessed from different nodes. A tree pattern is shared by two individual tree structure behaviors, if both of them have one same node that connects to at least two same nodes. If one parallel path in a discovered individual lattice behavior shares the same start node and also the same end node with another parallel path in another individual behavior, we call these two behaviors share the same parallel path, while we don't care if the middle nodes on separate path for two parallel path are the same or not. From above two tables about distribution of tree patterns and parallel patterns, we find that both of them are the rare patterns, which can be seen as the unexpected patterns in web usage mining; and also the complexity of such behaviors makes it less possible to find tree patterns or semi-lattice patterns. We can also further find from these two tables that the number of semi-lattice patterns is larger than that of tree patterns while the former support number is also larger than the later, one reason for this phenomenon is that our constraint for a semi-lattice patterns is less stricter than that of tree patterns, and the second reason is that more complex of individual behavior, and more impossible to find dominated popular complex patterns, although the size of discovered tree behavior set is larger than that of discovered

semi-lattice behavior set. So the support number on thousands of individual tree behaviors is set smaller than that of hundreds or less than hundred of individual semi-lattice behaviors. A circle path is a special sequence, which the start node is the same as the end node, so mining circle paths is the same as mining the largest forward paths from other researchers.

5 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 behavior.

In this paper, we discuss the individual access behavior through the web, and how to discover these behaviors from web logs. The complexity of web structure and the variety of visitors, and also the target patterns pursued decide that access behaviors can not be simplified into one category, and we define five different categories of individual access behavior. Before these behaviors were given, we also define 9 different basic actions that could be performed during a session. And individual access behavior is the combination of these basic actions. The experiment results show that our defined actions and behavior 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. B. Berendt and M. Spiliopoulou: Analysis of navigation behavior in web sites integrating multiple information systems, The VLDB Journal, (2000).
2. D. He and A. Goker: Detecting Session Boundaries from Web User Logs, 22nd Annual Colloquium on IR Research, (2000).
3. J. Srivastava, R. Cooley, M. Deshpande and P. Tan: Web Usage Mining: Discovery and Application of Usage Patterns from Web Data, ACM SIGKDD, (2000).
4. J. Pei, J. Han, B. Mortazavi-Asl and H. Zhu: Mining Access Patterns Efficiently from Web Logs, PAKDD, (2000).
5. J. Pei, J. Han and W. Wang: Mining Sequential Patterns with Constraints in Large Databases, ACM CIKM, (2002).
6. M. Spiliopoulou, B. Mobasher, B. Berendt and M. Nakagawa: A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis, INFORMS Journal on Computing, (2003).
7. M. Chen, J. Park and P. Yu: Data Mining for Path Traversal Patterns in a Web Environment, 16th International Conference on Distributed Computing Systems, (1996).
8. M. J. Zaki: Efficiently Mining Frequent Trees in a Forest, ACM SIGKDD, (2002).
9. R. Cooley, B. Mobasher and J. Srivastava: Data Preparation for Mining World Wide Web browsing Patterns, Knowledge and Information System, Journal of Knowledge and Information System, (1999).
10. T. Joachims, D. Freitag and T. Mitchell: WebWatcher: A Tour Guide for the World Wide Web, IJCAI, (1997).
11. X. Huang, F. Peng, A. An and D. Schuurmans: Dynamic Web Log Session Identification with Statistical Language Models, Journal of Information Science and Technology, (2004).