Data Mining User Sessions Based on Forward and Backward Reference Patterns

Rung Ching Chen Whei Yue Shen

Department of Information Management Chen-Kuo
Institute of Technology Chang Hua, Taiwan, R.O.C.
Email:crching@cc.ckit.edu.tw

Abstract

In this paper, we propose a novel method to explore the user sessions. After the data mining for the user sessions, the system management can understand which web pages or traversal paths are often referenced. An efficient algorithm is used to convert the user sessions into the forward and backward reference patterns. By the algorithm of analyzing the forward and backward reference patterns, we can find the times of the web pages and the traversal paths were referenced. If the reference times of the web pages or traversal paths are greater than a threshold, also named support value, the system will output them as often reference patterns. Even the support value is changed, the system is unnecessary to recalculate the accessed times of the reference patterns. The simulation results confirm our method is efficient to analyze the traversal information on the web site.

1. Introduction

Data mining the custom behavior has high applicability in retail market. The custom transaction behavior can be extracted from the settle or square accounts front desk. Those transaction data are transferred into association rules. Each association rule records a transaction action. By way of data mining those association rules, we can understand the time of man or woman stay in merchant, the commodity relation of which customer purchase, the relation of the commodity, the customer age and sex, the sell condition of each article, and so on. The manager can then rearrange the goods in the suit location from above information. Many papers are published for the data mining of those traction association rules[1,2,3]. The accessed times of the association rules were recorded. Always the association rules are so large that finding the meaningful association rules is needed. A support value is defined as the number of times a reference sequence has to appear in order to be qualified as large reference sequence. Those that reference times greater than the support values are the information that system required. Another application of data mining is on ordered data, such as stock market and point of sales data [8]. For example, the stock market data mining interesting on the change of strokes' price with the time sequences and the similar price change sequences search. The goal of application domain of data mining effects the kinds of data to be mined.

In recent, the e-commerce is very flourishing. Many companies or persons set up stores on the web sites. A web site is an internet location that contains hyperlinked documents. Many trades are finished on the web sites of internet. When a web site has been set up, the provider can analyze the customer behavior by the log file of the web site. By way of analyzing the log files, the management also can find the information people who always buy what kinds of goods and the relations between the goods and customers. By mining the information, which web page is more suited to place what kind of goods on the web side can be found. A web page is a hypertext or hypermedia document residing on an internet computer that contains text, graphics, video, or sound, where a hypertext document contains text hyperlinks to other documents; a hypermedia document contains text, graphics, image, sound hyperlinks that connect to other documents. How to analyze the data that customer on the web is an important topic. Also many researches have been published on this topics recently [4,5,6,7]. On this topic, one kind of approach is to analyze the user traversal path. A traversal path is defined as a user login the web site, by which the web pages they have arrived and the order they went on the web site. The traversal path will be recorded on the log file of the web site. Also, the WWW server management hope to have the report, about which pages and traversal paths are the more often referenced, to decide the prices of the location the advertisement on the WWW server. A user on the web site who browses the pages during a period of time is defined as a user session. A user session is organized as a series of patterns, where a pattern is a web page. For example, the user session {A-> B-> C-> D-> C-> E-> A-> B-> F-> G-> A-> H-> I-> J-> I->K} stands for the user arriving the web site and the order they travel on the web site is web page 'A' first then web pages 'B', 'C', ……, 'K' by order. During a period of time, lots of user sessions will be recorded on the log file.

The time period is used to segment the user session from the log file[5]. In this paper, we assumed that the user sessions have been extracted from the log file of a web site. We transfer the user session into forward and backward reference patterns. A forward reference pattern stands for the traversal path that the user on the web site does not backward to the web page that has been arrived. Reference [5] transfer the user session into a maximum forward reference patterns. The maximum forward reference patterns will record some web pages accessed times repeated to miss the real accessed times on the web site. For example, the maximum forward reference will transfer Figure 1 visited order {A-> B-> C-> D-> C-> E-> A-> B-> F-> G-> A-> H-> I-> J-> I->K} into the maximum forward reference patterns {{A-> B-> C-> D}, {A-> B-> C-> E}, {A-> B-> F-> G}, {A-> H-> I-> J},{A->H->I->J}}. We can find the web page "A" and web page "B" in the set of maximum forward reference patterns appeared five and three times respectively but the pure visiting on the user session is three and two times. Based on above appearance, we find that the maximum forward reference patterns can not truly appear the user visiting on the web site.

Improving the weakness of maximum forward reference patterns, which loses pure web pages accessed information, we propose a new method to represent the user session. When the user backward from a web page, we record the forward browsing path. Next, a backward browsing path is start. The backward browsing path termination when the user start a new reference pattern. Repeating the same steps mentioned above, we can find a new forward browsing path is happened on that start from a backward terminal web page through new forward pages and terminal at backward web page. Reversing, a backward reference patterns are extracted from a forward terminal web page through visited pages and terminal at a new web page. In Figure 1, the visiting order are {A-> B-> C-> D-> C-> E-> A-> B-> F-> G-> A-> H-> I-> J-> I->K}, the same as above traversal order. The user session will be converted into the forward reference patterns {{A-> B-> C-> D}, {C->E}, {A-> B-> F->G}, {A-> H-> I-> J}, {I-> K}} and the backward reference patterns {{D-> C}, {E-> A}, {G-> A}, {J-> I}} respectively. The forward reference web pages "A" and "B" is visited three and two times respectively. This truly appears the user accessed information. After the forward and backward reference patterns are generated, we can analyze the travel paths and the reference web pages accessed times. The information can provide the web site management to make policy decision.

The remainder of the paper is organized as follows. The section 2 described an algorithm to convert the user session into forward and backward reference patterns. The reference times of the reference patterns are found in section 3. A simulation based on Java and C language is presented in section 4. We also make conclusions and discussion in section 5.

2. The generation of forward and backward reference patterns.

In the session, we propose a Bi-Direction_Reference_ Patterns_Extraction algorithm, which will be called BDRPE algorithm short, to convert the user session into the forward and backward reference patterns. The means of the symbols and the operation of the algorithm were described as follows.

The US[i,j] stands for the input of user session where i is the index of input user session and j is the index of web page in the "i" user session. The FRP[n] stands for the forward reference patterns and BRP[m] stands for the backward reference patterns where "n" and "m" are the index of forward and backward reference pattern respectively. A user session can be converted into more than one forward and backward reference patterns. The FSB and BSB stand for forward and backward string buffers are used to store the alphabets extracted from a user session. Two flags organize four conditions includes to detect the condition of writing a alphabet to the FSB, writing the FSB to FRP[n], writing a alphabet to the BSB and writing the BSB to BRP[m]. Table 1 shows the relation of action with Flag1 and Flag2. For example, outputting a forward reference patterns is happened on the "Flag1" equals to zero and "Flag2" equals to one. The algorithm is expressed as follows.

Algorithm:
Bi-Direction_Reference_Patterns_Extraction

```
Input : The string of user session patterns US[i,j]; i is index
        of user session; j is index of user session pattern.
        i<NUS;
Output : The forward reference patterns FRP[n] and
         backward reference patterns BRP[m] Step 1:
         Initialization. Let m=n=i=0.
Step 2: While (i <NUM) do Step 3 to Step 8   // NUS is
number of user sessions.
Step 3: Detecting the length of the US_i , then storing the
        value to the variable "length"; let j=0; Flag1=0;
Step 4: While (j<length) do Step 5 to Step 7.
Step 5: Reading a alphabet from the US[i,j] and checking
          if US[i,j] ∈ {US[i, 0], US[i,1], ……, US[i,j-1]}
              Flag2=1;
          else
              Flag2=0;
Step6: do case
      Case Flag1=0 and Flag2=1
            {Adding the character of ser session US[i,j] to
            FSB
            Remainder_Flag=1;
            Go to Step 7.
            }
      Case Flag1=1 and Flag2=1
            {Writing forward string buffer FSB to FRP[n]
            Writing US[i,j-1] and US[i,j] to the backward
            buffer string BSB in order.
            Discard the alphabets in the {US[i, 0],
            US[i,1], ……, US[i,j-1]}
            from alphabet US[i, j-1] to US[i,j-r]. Where
            US[i,j-r] is the
            alphabet the same as alphabet US[i,j] and 1 < r
            ≤ j.
            n++;
            Flag1=0;
            Go to Step 7.
            }
        Case Flag1=1 and Flag2=0
            { Adding the character of user session US[i,j]
```

```
                to BSB
                Remainder_Flag=0;
                  Discard the alphabets in the {US[i, 0], US[i,1], ……, US[i,j-1]}
                  from alphabet US[i, j-1] to US[i,j-r]. Where US[i,j-r] is the
                  alphabet the same as alphabet US[i,j] and 1 < r ≤ j
                  Go to Step 7;
                  }
          Case Flag1=0 and Flag2= 0
                  {Writing backward buffer string BSB to BRP[m]
                  Clearing the backward buffer string BSB.
                  Writing US[i,j-1] and US[i,j] to the forward buffer string FSB by order.
                  m++;
                  Flag1=1;
                  }
Step 7 : j++;
Step 8: i++;
Step 9: The remainder reference pattern writing out:
             if(Remainder_Flag==1)
                Writing the remainder characters in FSB to FRP[n]
             else if(Remainder_Flag==0)
                Writing the remainder characters in BSB to BRP[m]
Step 10: End.
```

Figure 2 shows a example of web structure with user session {A-> B-> C-> D-> C-> D-> C-> D-> C-> B-> E-> G-> H-> G-> W-> A-> O-> U-> O->V} to generate the forward and backward reference patterns by the BDRPE algorithm which is described as follows.

First, the character 'A' is read, the Flag1 and Flag2 will be set as 0,1 respectively, then character 'A' is stored to FSB. The work is doing the same until the fifth character 'C' is read, which will set both Flag1 and Flag2 to 1, then write out the characters "ABCD" from FSB to the FRP[n] and write out the characters "D" and "C" to BSB. When the characters "D" and "C" were written to BSB, the fourth character 'D' and the third character 'C' are cut from the user session patterns. Next, the sixth character in the user session pattern is read both of the Flag1 and Flag2 will be set to 0. The characters "DC" in the BSB will be written to the BRP[m] and the system wrote out the characters "C" and "D" to the FSB. Others execution of the orders filed on Table 2 are doing the same. Table 2 shows the detail actions of the algorithm Bi-Direction_Reference_Patterns_ Extraction. The most right character of gray lattice in the field user session stands for the character is processing.

We focus on the order 10, where Flag1 is '1' and Flag1 is '0', that stands for the character is in backward condition which will do until both Flag1 and Flag2 are '0'. On the order 21, the algorithm wrote the characters which were kept in the forward buffer to FRP[n].

From the example, we can find the characters in FSB are written to FRP[n] when both Flag1 and Flag1 are '0' and the characters in BSB are written to BRP[m] when both Flag1 and Flag2 are '1'. The remainder characters in FSB or BSB will be written to FRP[n] or BRP[m] when the program is terminal. The user session will be transferred into the forward reference patterns {{A-> B-> C-> D}, {C-> D}, {C-> D}, {B-> E-> G-> H}, {G->W}, {A-> O-> U}, {O->V}} and the backward reference patterns {{D-> C}, {D-> C}, {D-> C-> B}, {H-> G}, {W-> A}, {U-> O}}.

3. Counting the accessed times of forward and backward reference patterns

The user sessions had been transferred into forward and backward reference patterns in session 3. Always the number of forward and backward reference patterns are large. In references[5,6], they detect the reference times based on selective scan and full scan. Both of them have problems that existed when the support value is changed the system must recalculate the large database and the database is scanned too many times. The support value is the number of times a reference sequence has to appear in order to be qualified as large reference sequence. In this session, we propose an effective algorithm to find accessed times of the forward reference patterns. The algorithm also applies to find the accessed times of various lengths of sequences of the backward reference patterns. Before the algorithm is described, the method of decomposing forward reference patterns is described previously as follows.

Assuming a forward reference pattern is (M_1->M_2 ->M_3 ->M_4->……->M_N) with length N. We then can extract the information include number of N with one alphabet patterns, which are { M_1, M_2, M_3, M_4,……,M_N }; number of N-1 with two alphabet patterns, which are { M_1->M_2, M_2->M_3, M_3->M_4->……,M_{N-1}->M_N }; the number of N-2 with three alphabet patterns which are { M_1->M_2->M_3, M_2->M_3->M_4,……, M_{N-2}->M_{N-1}->M_N };……, and so on. The final of pattern (M_1->M_2->M_3->M_4->……->M_N) has length N. Figure 3 showed the example of decomposing expression of the patterns{A->B->C->D} into {{A, B, C, D}, {A->B, B->C, C->D}, {A->B->C, B->C->D}, {A->B->C->D}}. We now can add up the one alphabet, two alphabets, three alphabets,….., until N alphabets appearance times respectively. All the accessed times will be sorted. The reference times of the alphabets and the traversal paths can be sorted respectively by decrement, then we can find the top N more often reference pages from one alphabet accessed times. The same as above described, we can find the various length of traversal paths of top N reference times. If provider defines a support value, we also can extract the pages and traversal paths that accessed times greater than the threshold by the sorting of accessed times, even the support value which changed the system can efficiently extracted the information that the provider needed.

Link lists are used to present the data structure. Each length of the alphabets is corresponding to a link list. The nodes in each link-list include two fields. One is used to store the alphabet or alphabets and the other record the accessed times of the alphabet or alphabets. Figure 4 shows the structure of the link lists. The operation of the Reference_Patterns_Accessed_Times_Counted algorithm and the Link_List_Accessed_ Times_Counted algorithm are described as follows.

First, the Reference_Patterns_Accessed_Times_Counted algorithm reads a reference pattern from FRP[n] and finds the length of this pattern, then decomposes the reference pattern with various lengths from 1 to the length of this pattern. Each length of pattern will call the Link_List_Accessed_Times_Counted algorithm to decide whether the reference times of alphabet or alphabet pattern must be incremented or a new reference alphabet or pattern has to be added to the link list. The next forward reference pattern is read and done the same action as above mention until all reference patterns have been processed. The reference times in each various length link-lists will be sorted by decrement. The two algorithms are described as follows.

Algorithm: Reference_Patterns_Accessed_Times_Counted

```
Input: Forward or backward reference patterns
Output: The reference times of each alphabet and various
        length alphabets.
Step 1: Initialize: Let i=0; Max_Patterns_Length=0;
Step 2: While i<Max do Step 3 to Step 5; // Max is the
        number of forward reference patterns.
Step 3: Read FRP[i] and store the length of FRP[i] to
        variable Len
        If Len >Max_Patterns_Length
            Max_Patterns_Length=Len
Step 4: For r=1 to Len
        Call the Link_List_Accessed_Times_Counted(r,i)
        Algorithm
Step 5: i=i+1
Step 6: for n=1 to Max_Patterns_Length
        Sorting the link list of LINK_LIST[n] index on
        accessed times.
Step 7: End.
```

Algorithm: Link_List_Accessed_Times_Counted(r, i)

```
Input: The FRP[i,j] and the index r, where r is a global
        variable get from
        Reference_Patterns_Accessed_Times_Counted
        Algorithm ,0<r≤Len. 0<=i<Max
Output: Increment of the accessed times of LINK_LIST[r]
        or add a new element to Link List.
Step1: Initialization: let Exist_flag=0;
Step 2: For p=0 to p<Len do Step 3 to Step 5
Step 3: Extracting sub-string Pattern $PS_{p..p+r-1}$ from FRP[i, p]
        to FRP[i,p+r-1] of FRP[i]
Step 4: For s=0 to Link_List_length[r] do
        {If $PS_{p..pr+1}$ == $LINK\_LIST_r$ [s].alphabets
            { $LINK\_LIST_r$ [s].times= $LINK\_LIST_r$ [s].times
              +1
             Exist_Flag=1;}
        }
Step 5: IF Exist_Flag equal to 0
            {Adding pattern $PS_{p..p+r-1}$ to $LINK\_LIST_r$
              [s].alphabets
             Link_List_length[r]++;
            }
Step 6: Return
```

The Reference_Patterns_Accessed_Times_Counted algorithm calls the Link_List_ Accessed_Times_Counted algorithm in Step 4. Exist_Flag is used to check whether the alphabet or alphabets in the link list already? The $PS_{p..p+r-1}$ stands for a sub-pattern extract from index p to p+r-1 of the FRP[i,j], where p is start of extraction alphabet index and r is the length of the alphabets.

The accessed times of backward reference pattern can be found only change the input array FRP[n] as BRP[m]. From the accessed times of backward reference pattern we can find which web pages are often backward and which backward path of web pages are usually happened. The information of backward reference times also can help the web-site manager make policy decisions.

4. Simulation

The simulation of web site structure and generation the user sessions are implemented by Java language. Three major algorithms are implemented by C language. All the functions are tested on K7-700 with M RAM. Since the home page is the most frequently visited, we make a reasonable high probability criteria that each user session often start from the home page designated by root node. In the following experiment, a web site structure is created first. Next, a set of user sessions based on the web site structure will be established to simulate the user browse order on this web site. Those user sessions will work as the input of the algorithm Bi-Direction_Forward_ Reference_Patterns_Extraction. The output of the algorithm will be analyzed by Reference_Patterns_Accessed_ Times_Counted and Link_ List_ Accessed_Times_Counted algorithms. Then we can find the top T more often and top L less seldom reference pages and reference path patterns for the forward and backward reference pattern.

Constructing the web structure, two parameters will be given first. One is the depth of the web site "D", the other is the maximum number of the child nodes for each parent node "M". A web site structure is like a tree structure. These two parameters will determine the total nodes of the web tree. In the system, also two optional choices N and L. N stand for the minimum number of the child nodes for each parent node. L denotes for the maximum levels whose nodes are backward hyper-linked. Each parent node will have the number of child nodes between N and M. For simplicity, we reduce the total nodes to 50.

In the program, we suppose that the root node could be backward hyper-linked by each child node. In other words, any child node can directly return to the root node. Once the intermediate node is entered, each of the following downstream nodes could be selected with equally probability. For example, the intermediate node N which belongs to level L4 can link to either all its child nodes N[5][k],where k is the number of child nodes of node N, those nodes belongs to level L5. The index i in N[i][k] stands for level Li. Also the node N can connect itself. Therefore, node N as depicted in Figure 5 will have the same opportunity in choosing one of its downstream nodes { A, B, C, D, E, F, G, H, N, e, f, g } as the next candidate node. The length of the user session is a Poisson Distribution with $\lambda=6$. The root node A is not the first entry in each user session. Consequently, the root node of "A" will have high probability to work as start nodes. We generate 100000 user sessions to detect the three algorithm mentioned in session 2 and session 3. The processing time of converting the one hundred thousands user sessions into forward and backward reference patterns only takes 0.6 sec. The analysis of the forward and backward reference patterns using Reference_Patterns_Accessed_Times_ Counted and Link_List_Accessed_Times_Counted algorithms which only takes 22.2 sec.

5. Conclusions

In this paper, we have proposed a novel method that transferred the user traversal path to forward and backward reference patterns. The forward and backward reference patterns can response the traversal page of the visitor browsing in the web-site. Bi-Direction_Reference_ Patterns_Extraction algorithm is used to transfer the user traversal path into forward and backward reference patterns. Using the Reference_Patterns_ Accessed_Times_Counted algorithm and Link_List_Accessed_Times_Counted algorithms, we can find the accessed times of the forward and backward reference patterns in various lengths. The reference times will be sorted then we can find which pages and traversal paths are top N more often be reference of the forward and backward reference patterns. In the future work, we will mine the real log file of NT WWW server for the service provider to get more information to make policy decisions.

References

[1] R. Agrawal and R., T. Imielinski, and A.Swami, "Mining Association Rules between Sets of Items in Large Database," Proc. ACM SIGMOD, pp. 207-216, May,1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for mining association rules in large databases," Proc. ACM SIGMOD, pp. 478-499, Sept. 1994.

[3] J.Han and Y. Fu, "Discovery of Multiple-level association rules from large databases, " Proc. 21th Int'l Conf. Very Large Data Bases, pp. 420-431, Sept. 1995.

[4] D.L. Yang and S.H. Yang, " A study on mining session path patterns," Fourth conference on artificial intelligence and applications, pp. 37-44,1999m Taiwan. .

[5] J.S. Park, M.S. Chen, and P.S. Yu, "Using a hash-based method with transaction trimming for mining association rules", IEEE Transactions on knowledge and data engineering, Vol.9.No.5. pp. 813-825, 1997

[6] M-S Chen, J.S. Park, and P.S. Yu, "Efficient Data Mining for path traversal patterns", IEEE Transactions on knowledge and data engineering, Vol.10.No.2. pp. 209-221, 1998

[7] Fu-Ren Lin, Shih-ta Chang, "Mining user accessed patterns from network flow on the Internet," the Eleventh National Conference on the Information Management, Taiwan, 2000.

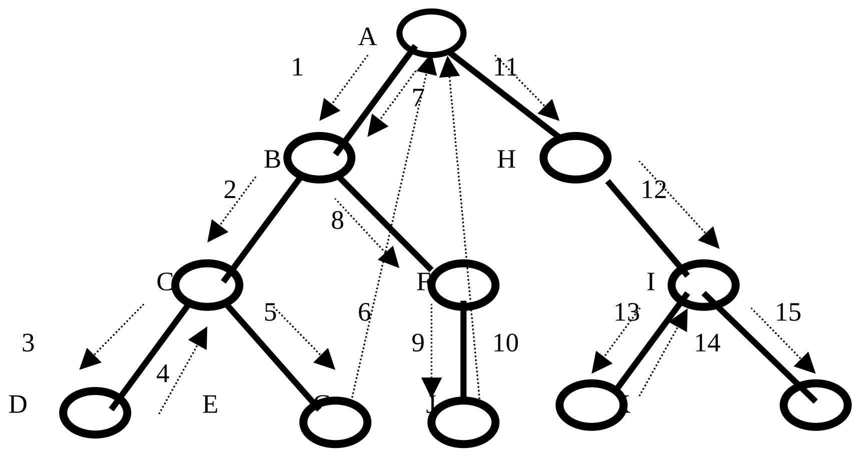


Figure 1. An example of a user session {A-> B-> C-> D-> C-> E-> A-> B-> F-> G-> A-> H-> I-> J-> I->K}.

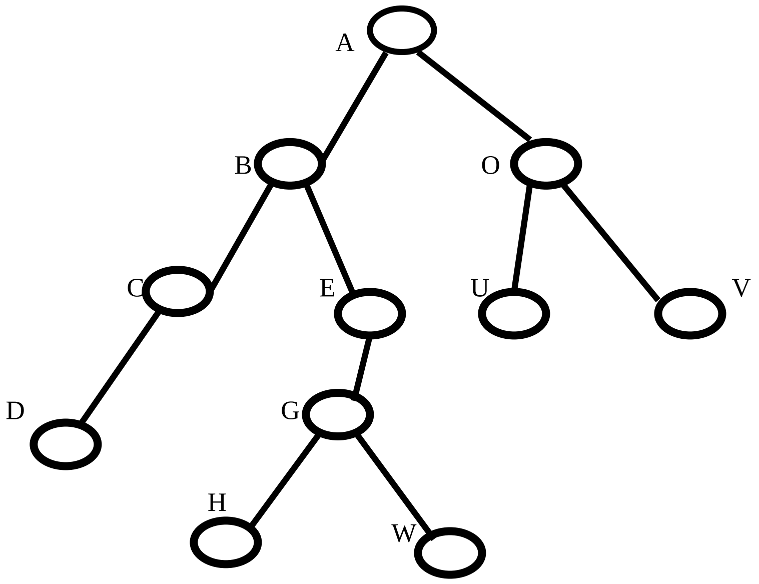


Figure 2. A web structure has session {{A-> B-> C-> D-> C-> D-> C-> D-> C-> B-> E-> G-> H-> G-> W-> A-> O-> U-> O->V}

$$\begin{tikzcd}
 & \{a,b,c,d\} \quad \text{One alphabet} \\
 & \{ab,bc,cd\} \quad \text{two alphabets} \\
\{abcd\} \arrow[ur] \arrow[uur] \arrow[r] \arrow[dr] & \{abc,bcd\} \quad \text{three alphabets} \\
 & \{abcd\} \quad \text{four alphabets}
\end{tikzcd}$$

Figure 3. The decompose of a forward or backward reference pattern

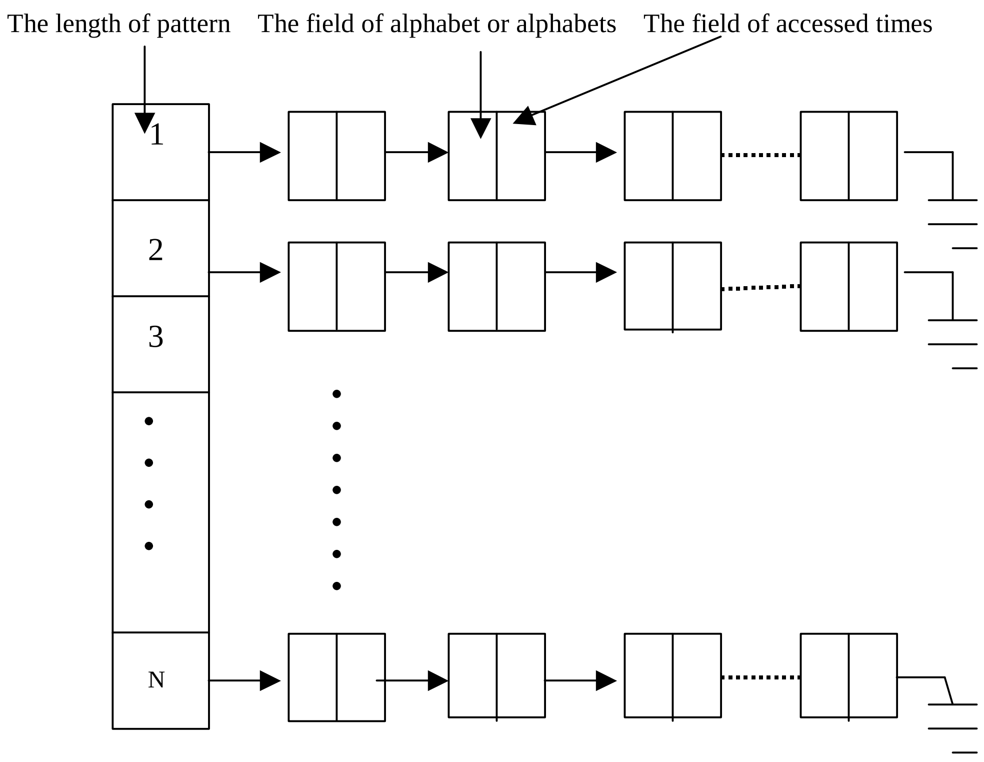
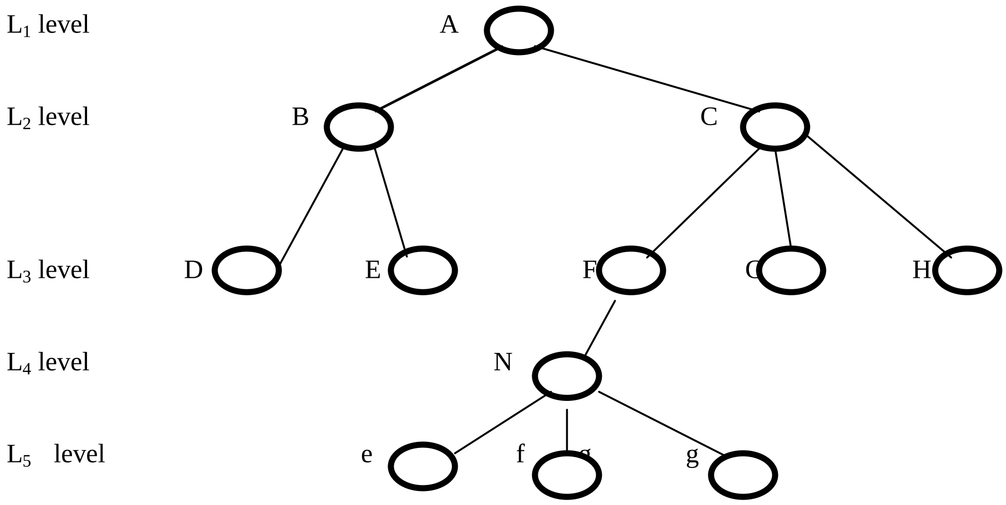

The length of pattern
The field of alphabet or alphabets
The field of accessed times
1
2
3
N

Figure 4. The structure of link list.



N[5][k]={e,f,g} N[4][k]={N} N[3][k]={D,E,F,G} N[2][k]={B,C} N[1][0]={A}

Figure 5. The possible link pages of node N

Table 1. The action control of Flag1 and Flag2.

Flag1	Flag2	Action
0	0	Writing BSB to BRP[m]
0	1	Writing the character to FSB
1	0	Writing the character to BSB
1	1	Writing FSB to FRP[n]

Table 2. The process of the algorithm of Bi-direction reference patterns extraction

order	User Session	Flag1g1	Flag1lag	FRP[n]	BRP[m]
1	ABCDCDCDCBEGHGWAOUOV	0	1		
2	ABCDCDCDCBEGHGWAOUOV	0	1		
3	ABCDCDCDCBEGHGWAOUOV	0	1		
4	ABCDCDCDCBEGHGWAOUOV	0	1		
5	ABCDCDCDCBEGHGWAOUOV	1	1	ABCD	
6	AB**CDCDCBEGHGWAOUOV	0	0		DC
7	AB**CDCDCBEGHGWAOUOV	1	1	CD	
8	AB****CDCBEGHGWAOUOV	0	0		DC
9	AB****CDCBEGHGWAOUOV	1	1	CD	
10	AB******CBEGHGWAOUOV	1	0		
11	A********BEGHGWAOUOV	0	0		DCB
12	A********BEGHGWAOUOV	0	1		
13	A********BEGHGWAOUOV	0	1		
14	A********BEGHGWAOUOV	1	1	BEGH	
15	A********BE**GWAOUOV	0	0		HG
16	A********BE**GWAOUOV	1	1	GW	
17	***************AOUOV	0	0		WA
18	***************AOUOV	0	1		
19	***************AOUOV	1	1	AOU	
20	***************A**OV	0	0		UO
21	***************A****	1	1	OV	