A Random Indexing Approach for Web User Clustering and Web Prefetching

Miao Wan[1], Arne Jönsson[2], Cong Wang[1], Lixiang Li[1], and Yixian Yang[1]

[1] Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, P.O. Box 145, Beijing 100876, China

[2] Department of Computer and Information Science, Linköping University, SE-581 83, Linköping, Sweden

wanmiao120@163.com, arnjo@ida.liu.se

Abstract. In this paper we present a novel technique to capture Web users' behaviour based on their interest-oriented actions. In our approach we utilise the vector space model Random Indexing to identify the latent factors or hidden relationships among Web users' navigational behaviour. Random Indexing is an incremental vector space technique that allows for continuous Web usage mining. User requests are modelled by Random Indexing for individual users' navigational pattern clustering and common user profile creation. Clustering Web users' access patterns may capture common user interests and, in turn, build user profiles for advanced Web applications, such as Web caching and prefetching. We present results from the Web user clustering approach through experiments on a real Web log file with promising results. We also apply our data to a prefetching task and compare that with previous approaches. The results show that Random Indexing provides more accurate prefetchings.

Keywords: Web user clustering, User behavior, Random Indexing, Web prefetching.

1 Introduction

Web users may exhibit various types of behaviours associated with their information needs and intended tasks when they are navigating a Web site. These behaviours are traced in Web access log files. Web usage mining [2], which is an important topic in behavior informatics [3] and a branch of Web Mining [1], captures navigational patterns of Web users from log files and has been used in many application areas [8,9,10,11].

Clustering in Web usage mining is used to group together items that have similar characteristics, and user clustering results in groups of users that seem to behave similarly when navigating through a Web site. In recent years, clustering users from Web logs has become an active area of research in Web Mining. Some standard techniques of date mining such as fuzzy clustering algorithms [4], first-order Markov models [5] and the Dempster-Shafer theory [6] have been introduced to model Web users' navigation behaviour and to cluster users based on

L. Cao et al. (Eds.): PAKDD 2011 Workshops, LNAI 7104, pp. 40–52, 2012.

Web access logs. Generally, these techniques capture stand alone user behaviours at the page view level. However, they do not capture the intrinsic characteristics of Web users' activities, nor quantify the underlying and unobservable factors associated with specific navigational patterns. Latent variable models, such as LSA [14], have been widely used to discover the latent relationship from web linkage information and can be utilized to find relevant web pages for improving web search efficiency [7].

Random Indexing [12] is an incremental word space model proposed as an alternative to LSA. Since 2000, it has been studied and empirically validated in a number of experiments and usages in distributional similarity problems [12,13]. However, few of the Random Indexing approaches have been employed into the field of Web mining, especially for the discovery of Web user access patterns.

In this paper we propose a Web user clustering approach to prefetch Web pages for grouped users based on Random Indexing (RI). Specially, segments split by "/" in the URLs are used as the units of analysis in our study. To demonstrate the usability of RI for user cluster detection, we also apply our methodology to a real prefetch task by predicting future requests of clustered users according to the common pages they accessed. Our clustering and prefetching approaches based on Random Indexing are compared to a popular Web user clustering method named FCMdd [4], and a new proposed clustering algorithm called CAS-C [9]. The experimental results show that the RI-based Web user clustering technique present more compact and well-separated clusters than FCMdd and CAS-C, and get higher prefetching accuracy as well.

2 Random Indexing (RI)

Random Indexing is an incremental word space model based on Kanerva's work on sparse distributed representations [12,15]. The basic idea of Random Indexing is to accumulate context vectors based on the occurrence of words in contexts. This technique can be used with any type of linguistic context, is inherently incremental, and does not require a separate dimension reduction phase. The Random Indexing technique can be described as a two-step operation:

Step 1. A unique d-dimensional *index vector* is assigned and randomly generated to each context (e.g. each document or each word). These index vectors are sparse, high-dimensional, and ternary, which means that their dimensionality (d) is on the order of hundreds, and that they consist of a small number(ϵ) of randomly distributed +1s and -1s, with the rest of the elements of the vectors set to 0. In our work each element is allocated with the following probability [13]:

$$\begin{cases} +1 & with\ probability\ \frac{\epsilon/2}{d} \\ 0 & with\ probability\ \frac{d-\epsilon}{d} \\ -1 & with\ probability\ \frac{\epsilon/2}{d} \end{cases}$$

Step 2. *Context vectors* are produced by scanning through the text. As scanning the text, each time a word occurs in a context, that context's d-dimensional

index vector is added to the context vector for the word. Words are thus represented by d-dimensional context vectors that are the sum of the index vectors of all the contexts in which the word appears.

The Random Indexing technique produces context vectors by noting co-occurring events within a context window that defines a region of context around each word, and the number of adjacent words in a context window is called the context window size, l. For example, the term t_n in a '2+2' sized context window, c_m, is represented by:

$$c_m = [(w_{n-2})(w_{n-1})t_n(w_{n+1})(w_{n+2})].$$

Here $l = 2$, and the context vector of t_n in c_m would be updated with:

$$C_m = R(w_{n-2}) + R(w_{n-1}) + R(w_{n+1}) + R(w_{n+2}),$$

where $R(x)$ is the random index vector of x. This process is repeated every time we observe t_n in our data, adding the corresponding information to its existing context vector C. If the context c_m is encountered again, no new index vector will be generated. Instead the existing index vector for c_m is added to C to produce a new context vector for t_n.

3 Random Indexing Based Web User Clustering

The procedure of Web user clustering based on Random Indexing is illustrated in Figure 1 and will be outlined in more detail in this section.

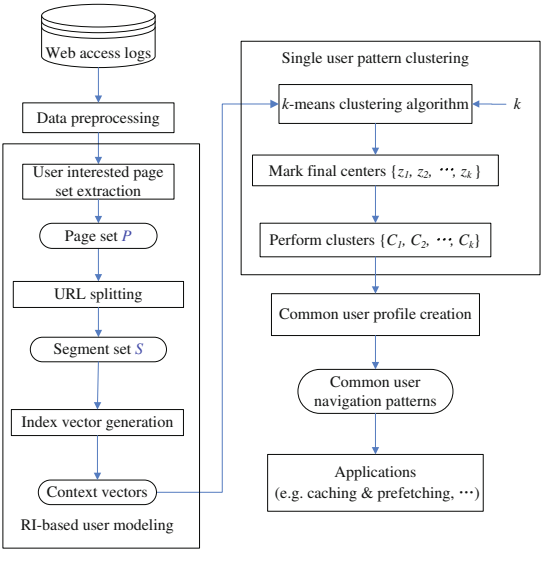

Fig. 1. Working flow of Web user clustering approach based on Random Indexing

3.1 Data Preprocessing

The first part of Web user cluster detection, called preprocessing, is usually complex and demanding. Generally, it comprises three domain dependent tasks: data cleaning, user identification, and session identification.

Data Cleaning. Depending on application and task, Web access logs may need to be cleaned from entry request pages. For the purpose of user clustering, all data tracked in Web logs that are useless, such as graphical page content (e.g. jpg and gif files), which are not content pages or documents, need to be removed.

User Identification. Identifying different users is an important issue of data preprocessing. There are several ways to distinguish individual visitors in Web log data which are collected from three main sources: Web servers, proxy servers and Web clients. In our experiments we use user identification from client trace logs because users in these logs are easily traced via different user IDs.

Session Identification. After individual users are identified, the next step is to divide each user's click stream into different segments, which are called sessions. Most session identification approaches identify user sessions by a maximum time-out. If the time between page requests exceeds a certain limit of access time, we assume a user is starting a new session. Many commercial products use 30 minutes as a default timeout [2]. We will also use 30 minutes in our investigations.

3.2 User Modelling Based on Random Indexing

After all the Web logs are preprocessed, the log data should be further analysed in order to find common user features and create a proper user model for user clustering.

Navigation set of Individual Users. Pages requested by a user only a very small period, such as one session, and not visited anymore, represent temporary user interest and are filtered out in this step. Pages with very low hit rates in the log files that only reflect the personal interest of individual users are also removed based on the pre-set hit number. This leaves us with a user interest page set $P = \{URL_1, URL_2, \ldots, URL_m\}$ composed of the remaining m requested URLs. Each element in P is successfully visited more than the pre-set number of times. Based on P we create a navigation set for individual users, $U = \{U_1, U_2, \ldots, U_n\}$, which contains pages requested by each user.

Random Indexing for Each User. The form of a webpage's URL can contain some useful information. According to the hierarchical structure of most Web sites, the URLs at different levels can be reflected in the sequence of segments split by "/". For example, `http://cs-www.bu.edu/faculty/gacs/courses/cs410/Home.html` may represent that it is the homepage of a course named "cs410" and this course is provided by someone called "gacs" who is a faculty of the department of computer science. Based on this assumption, we can split

all the URLs in the user interest page set, P, by "/" and create a set of segments, S, which contains all the segments that have occurred in P. For each segment, a d-dimensional index vector s_i ($i = 1, 2, \ldots, m$, where m is the total number of segments) is generated. Then, as scanning the navigation set U, for each segment appearing in one user, update its zero-initialised context vector u_j ($j = 1, 2, \ldots, n$, where n is the total number of users) by adding the random index vectors of the segments in the context window where the size of the context window is pre-set. Finally, a set of individual users' navigation patterns, which forms a $n \times d$ matrix $A = \{u_1, u_2, \ldots, u_n\}^T$, is created with each row as the context vector u_j of each single user.

3.3 Single User Pattern Clustering

After random indexing of user's transaction data, the single user patterns in matrix A will be clustered by the k-means clustering algorithm. The k-means clustering algorithm [16] is a simple, conventional, clustering technique which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. It is a partition-based clustering approach and has been widely applied for decades.

4 Clustering Validity Measures

In order to evaluate the performance of the proposed clustering algorithm and compare it to other techniques we use two cluster validity measures. Since the cluster number should be pre-set for the clustering algorithms, in this paper we first use a relative criterion named SD to estimate the number of clusters for the clustering algorithms before we evaluate their performances. Furthermore, as the access log is an un-marked data set, we choose another validity measure, called CS, to evaluate the performance of different clustering algorithms without any prior knowledge of the data set.

– The SD index combines the average scattering for clusters and the total separation between clusters. For each k input, the $SD(k)$ is computed as

$$SD(k) = Dis(k_{max}) \cdot Scat(k) + Dis(k), \tag{1}$$

where k_{max} is the maximum number of input clusters and influences slightly on the value of SD [17].

$Scat$ is the average scattering within one cluster and is defined as:

$$Scat(k) = \frac{1}{k} \sum_{i=1}^{k} \frac{\|\sigma(C_i)\|}{\|\sigma(X)\|}, \tag{2}$$

where $\sigma(S)$ represents the variance of a data set S. X is the entire data set.

Dis is the total scattering (separation) between clusters and is given by the following equation:

$$Dis(k) = \frac{D_{max}}{D_{min}} \sum_{i=1}^{k} \left(\sum_{j=1}^{k} \|z_i - z_j\| \right)^{-1}, \tag{3}$$

where $D_{max} = max(\|z_i - z_j\|)$ and $D_{min} = min(\|z_i - z_j\|)$ $(\forall i, j \in 1, 2, 3, \ldots, k)$ are the maximum and minimum distances between cluster centers.

Experiments show that the number of clusters, k, which minimizes the SD index can be considered as an optimal value for the number of clusters present in the data set [17].

– The CS index computes the ratio of *Compactness* and *Separation*.

A common measure of *Compactness* is the intra-cluster variance within a cluster, named *Comp*:

$$Comp = \frac{1}{k} \sum_{i=1}^{k} \|\sigma(C_i)\|. \tag{4}$$

Separation is calculated by the average of distances between the centers of different clusters:

$$Sep = \frac{1}{k} \sum \|z_i - z_j\|^2, \quad i = 1, 2, \ldots, k-1, \quad j = i+1, \ldots, k. \tag{5}$$

It is clear that if the dataset contains compact and well-separated clusters, the distance between the clusters is expected to be large and the diameter of the clusters is expected to be small. Thus, cluster results can be compared by taking the ratio between *Comp* and *Sep* :

$$CS = \frac{Comp}{Sep}. \tag{6}$$

Based on the definition of CS, we can conclude that a small value of CS indicate compact and well-separated clusters.

5 Experiments

In this section, we present an experiment using the RI-based Web user clustering approach, and give a detailed investigation of results using the method. We use MatLab for our experiments.

The results produced by our Web user clustering algorithm can be used in various ways. In this section we will illustrate how it can be used for prefetching and caching, which means that URL objects can be fetched and loaded into the Web server cache before users request them.

5.1 Preprocessing of Data Source

The data source for the Web user clustering algorithm is the Web site access log of the Computer Science department at Boston University [18] which is available at The Internet Traffic Archive [19]. It contains a total of 1,143,839 requests for data transfer, representing a population of 762 different users.

We use the part of the logs during the period of January and February 1995. According to the item of 'user id' in the log data, we selected 100 users in the step of user identification. After access log preprocessing, we get 1005 sessions from these 100 users. The User IDs are renumbered, and each one of them have been assigned an identification number between 1 and 100.

5.2 Parameter Setting Investigations

In this subsection we present results from our investigations on the impacts of some key parameters and assign initial values for them.

Cluster Number. First we need to find the proper k values for each clustering algorithm in our work.

We have conducted k-means clustering experiments by measuring SD values for various values of k. The experiment is performed 50 times with 9 different values of k (from 2 to 10) and 6 different dimensions (from $d = 100$ to 600). The results of the k-investigations are given in Figure 2.

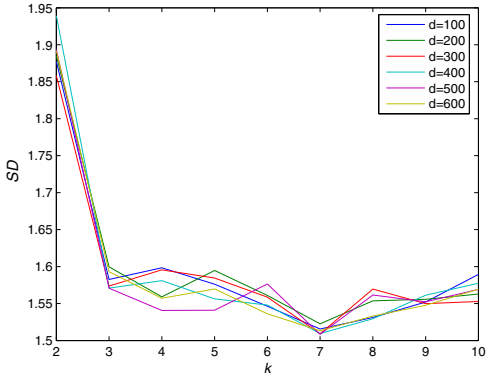

Fig. 2. Comparisons of SD measures for k-means clustering in RI-based tasks

As seen in Figure 2, the SD index is marginally influenced by the dimension of the index vectors. The distribution of SD values for different values of d is similar for most user pattern matrixes and the minimum SD is found at the same k for all dimension settings. Thus, we set $k = 7$ as the optimal cluster number for the k-means clustering algorithm.

We perform similar experiments for the FCMdd and CAS-C algorithms as depicted in Figures 3 and 4. We use $k = 8$ for FCMdd and $k = 7$ for CAS-C.

Dimensionality. In order to evaluate the effects of increasing the dimensionality to the performance of Random Indexing in our work, we computed the values of CS with d ranging from 50 to 600. The performance measurers are reported using average values over 30 different runs. The results are depicted in Figure 5.

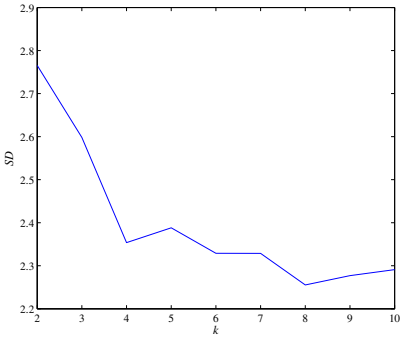

Fig. 3. Comparison of SD for FCMdd in Web user clustering

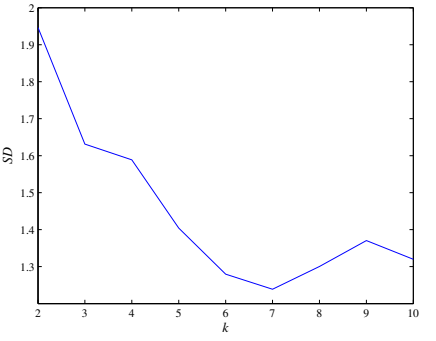

Fig. 4. Comparison of SD for CAS-C in Web user clustering

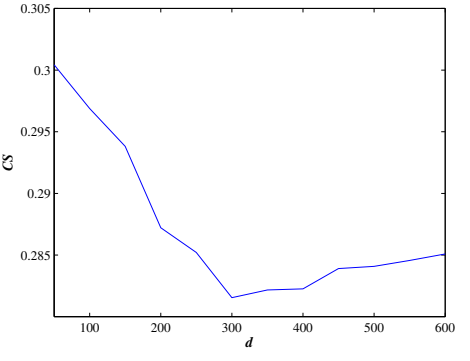

Fig. 5. The influence of various d values to RI-based Web user clustering

From Figure 5 we can see that for $d = 300$ we have the smallest CS. Thus, $d = 300$ is chosen as the dimension of the index vectors used by Random Indexing.

Other Parameters. Two more parameters need values: the number of +1s and -1s in the index vector, ϵ, and the context window size, l. We will use $\epsilon = 10$ as proposed by [20] and $l = 1$ as the URLs are rather short.

To summarise, we will use initial values of parameters in our RI-based experiments as presented in Table 1.

Table 1. Parameter selection in the RI-based approach

k	d	l	ϵ
7	300	1	10

5.3 Common User Profile Creation

After the log data are processed by Random Indexing, the single user navigation pattern matrix A will be clustered by the k-means algorithm.

The RI-based Web user clustering is compared to that generated using FCMdd and CAS-C. Table 2 presents the values of CS for different clustering techniques.

Table 2. Values of CS for different clustering approaches

Methods	k	*Comp*	*Sep*	CS
$FCMdd$	8	2.2401	5.8465	0.3832
$CAS-C$	7	2.2380	7.1574	0.3127
RI	7	2.1978	7.8054	0.2816

As shown in Table 2, the RI-based clustering algorithm gets the smallest $Comp$ with the largest Sep, and of course, the best CS value. We can conclude that the RI-based approach performs better than FCMdd and CAS-C for clustering Web users.

Based on the clustering results, we build a common user profile as seen in Table 3.

5.4 Prefetching for User Groups

Web caching and Web prefetching are two important techniques used to reduce the noticeable response time perceived by users [21]. For an effective prefetching scheme, there should be an efficient method to predict users' requests and proper prefetching and caching strategies. Various techniques, including Web Mining approaches [22,23], have been utilised for improving the accuracy of predicting user access patterns from Web access logs, making the prefetching of Web objects more efficient. Most of these techniques are, however, limited to predicting requests for a single user only. Predicting groups of users interest have caught little attention in the area of prefetching.

Prefetching Rule. According to the common user profiles created by the RI-based technique, FSMdd and CAS-C, we set up prefetching experiments to prefetch URL requests for users in each cluster. The prefetch rule is defined as follows:

For each cluster, let $P = \{p_1, p_2, \ldots, p_m\}$ be a set of Web pages in the Web server. In this paper, the prefetch rule is defined as an implication of the form $\{p_1, p_2, \ldots, p_i\} \xrightarrow{c} \{q_1, q_2, \ldots, q_j\}$, where $P_1 = \{p_1, p_2, \ldots, p_i\}$ is the page set that users requested in January and February, $P_2 = \{q_1, q_2, \ldots, q_j\}$ is the page set to be prefetched in March, $P_2 \subseteq P_1 \subseteq P$, and c is the portion (or ratio) of users who have requested P_2 in January and February.We use $c = 0.5$ [9] for our prefetch task, which means that more than or equal to 50% of the users pages in one cluster that have been requested in January and February will be prefetched for March.

Table 3. Common user profile created by Web user clustering algorithm using the RI-based approach. The CN column represents the cluster number.

CN	Members	Common user requests
1	4, 19, 33, 40, 67, 90	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu/faculty/heddaya/CS103/HW/1.html, cs-www.bu.edu/faculty/heddaya/CS103/HW/2.html, cs-www.bu.edu/faculty/heddaya/CS103/Home.html, cs-www.bu.edu:80/, sundance.cso.uiuc.edu/Publications/Other/Zen/zen-1.0_toc.html, www.ncsa.uiuc.edu/SDG/Software/Mosaic/StartingPoints/ NetworkStartingPoints.html, www.ncsa.uiuc.edu/demoweb/url-primer.htm
2	6, 61, 71, 83	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu/staff/Home.html, cs-www.bu.edu/staff/TA/biddle/www/biddle.html, cs-www.bu.edu/staff/TA/dmc/www/dmc.html, cs-www.bu.edu/staff/TA/joyceng/home.html, cs-www.bu.edu/staff/people.html, cs-www.bu.edu:80/
3	7, 8, 11, 21, 26, 30, 66, 89	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu/pointers/Home.html, cs-www.bu.edu/students/grads/Home.html, cs-www.bu.edu/students/grads/oira/Home.html, cs-www.bu.edu/students/grads/oira/cs112/hmwrk1.html, cs-www.bu.edu/students/grads/oira/cs112/hmwrk2.html, cs-www.bu.edu/students/grads/oira/cs112/hmwrkgd.html, cs-www.bu.edu/students/grads/oira/cs112/node1.html, cs-www.bu.edu:80/, cs-www.bu.edu:80/students/grads/oira/cs112/
4	1, 9, 12, 17, 24, 25, 31, 34, 35, 42, 50, 59, 72, 76, 78, 81, 82, 84, 97, 99, 100	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu:80/
5	10, 44, 56, 74, 87, 93	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu/faculty/Home.html, cs-www.bu.edu/students/grads/Home.html, cs-www.bu.edu/students/grads/tahir/CS111/, cs-www.bu.edu:80/
6	2, 3, 5, 14, 16, 20, 22, 23, 27, 28, 29, 32, 36, 37, 38, 39, 41, 43, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 57, 58, 60, 62, 63, 64, 68, 69, 70, 73, 75, 77, 79, 80, 85, 86, 91, 92, 94, 95, 96, 98	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu/students/grads/tahir/CS111/
7	13, 15, 18, 65, 88	cs-www.bu.edu/, cs-www.bu.edu/faculty/Home.html, cs-www.bu.edu/faculty/crovella/Home.html, cs-www.bu.edu/faculty/crovella/courses/cs210/, cs-www.bu.edu/faculty/crovella/courses/cs210/hwk1/hwk1.html, cs-www.bu.edu/faculty/crovella/courses/cs210/reading.html, cs-www.bu.edu/pointers/Home.html , cs-www.bu.edu:80/, cs-www.bu.edu:80/faculty/crovella/courses/, cs-www.bu.edu:80/faculty/crovella/courses/cs210/

Results and Analysis. Four parameters are used to investigate the performance of our prefetching task: (1) *hits* which indicate the number of URLs that are requested from the prefetched URLs, (2) *precision* which is the ratio of hits to the number of URLs that are prefetched, (3) *recall* which is the ratio of hits

to the number of URLs that are requested, and (4) $F_{0.5}$ which considers both the precision and the recall to test the accuracy. Since our prefetch strategy only predicts common URLs within one user cluster, we can't make sure that all requests from a single user are prefetched. Therefore precision is valued higher than recall for prefetching. As a result, we choose $F_{0.5}$ to measure the prefetching accuracy which weights precision twice as much as recall.

Table 4 gives the overall experimental comparison of prefetching for FCMdd, CAS-C and Random Indexing.

Table 4. Overall result comparisons for FCMdd, CAS-C and RI-based prefetchings

Algorithms	*Number of cluster detected*	*Overall precision*	*Overall recall*	$F_{0.5}$
FCMdd	8	0.7039	0.4434	0.6299
CAS-C	7	0.7062	0.4168	0.6201
RI	7	0.7540	0.5311	0.6956

Comparing the top two lines to the last row of Table 4, we can see that the results in the proposed prefetching task achieve a total average precision of 75.40% and a total recall of 53.11%, which are all higher than 70.62% of CAS-C and 44.34% using FCMdd. The $F_{0.5}$ value from RI, 0.6956, is therefore larger than 0.6201 of CAS-C and 0.6299 of FCMdd. We can thus conclude that prefetching based on Random Indexing provides a user request prediction that is better than using FCMdd or CAS-C.

To summarize, Random Indexing can improve the quality of user request prediction and perform better results than FCMdd and CAS-C.

6 Conclusions

This paper focuses on discovering latent factors of user browsing behaviours based on Random Indexing and detecting clusters of Web users according to their activity patterns acquired from access logs. Experiments are conducted to investigate the performance of Random Indexing in Web user clustering tasks. The experimental results show that the proposed RI-based Web user clustering approach could be used to detect more compact and well-separated user groups than previous approaches. Based on common profiles of detected clusters, prediction and prefetching user requests can be done with encouraging results.

Acknowledgments. This work is supported by the National Basic Research Program of China (973 Program) (Grant No. 2007CB311203), the National Natural Science Foundation of China (Grant No. 60805043), the Natural Science Foundation of Beijing, China (Grant No. 4092029), Huo Ying-Dong Education Foundation of China (Grant No. 121062), Specialized Research Fund for the Doctoral Program of Higher Education (No. 20100005110002) and Santa Anna IT Research Institute.

References

1. Etzioni, O.: The world-wide Web: quagmire or gold mine? Communications of the ACM 39(11), 65–68 (1996)
2. Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wide web browsing patterns. J. Knowl. Inf. Syst. 1(1), 5–32 (1999)
3. Cao, L.: In-depth Behavior Understanding and Use: the Behavior Informatics Approach. Information Science 180(17), 3067–3085 (2010)
4. Krishnapuram, R., Joshi, A., Nasraoui, O., Yi, L.: Low-complexity fuzzy relational clustering algorithms for web mining. IEEE Transaction of Fuzzy System 4(9), 596–607 (2003)
5. Cadez, I., Heckerman, D., Meek, C., Smyth, P., Whire, S.: Visualization of Navigation Patterns on a Website Using Model Based Clustering. Technical Report MSR-TR-00-18, Microsoft Research (March 2002)
6. Xie, Y., Phoha, V.V.: Web User Clustering from Access Log Using Belief Function. In: Proceedings of K-CAP 2001, pp. 202–208 (2001)
7. Hou, J., Zhang, Y.: Effectively Finding Relevant Web Pages from Linkage Information. IEEE Trans. Knowl. Data Eng. 15(4), 940–951 (2003)
8. Paik, H.Y., Benatallah, B., Hamadi, R.: Dynamic restructuring of e-catalog communities based on user interaction patterns. World Wide Web 5(4), 325–366 (2002)
9. Wan, M., Li, L., Xiao, J., Yang, Y., Wang, C., Guo, X.: CAS based clustering algorithm for Web users. Nonlinear Dynamics 61(3), 347–361 (2010)
10. Berendt, B.: Using site semantics to analyze, visualize, and support navigation. Data Mining and Knowledge Discovery 6(1), 37–59 (2002)
11. Ansari, S., Kohavi, R., Mason, L., Zheng, Z.: Integrating e-commerce and data mining: Architecture and challenges. In: Proceedings of ICDM 2001, pp. 27–34 (2001)
12. Kanerva, P., Kristofersson, J., Holst, A.: Random Indexing of text samples for Latent Semantic Analysis. In: Proceedings of the 22nd Annual Conference of the Cognitive Science Society, p. 1036 (2000)
13. Sahlgren, M., Karlgren, J.: Automatic bilingual lexicon acquisition using Random Indexing of parallel corpora. Journal of Natural Language Engineering, Special Issue on Parallel Texts 6 (2005)
14. Landauer, T., Dumais, S.: A solution to Plato problem: the Latent Semantic Analysis theory for acquisition, induction and representation of knowledge. Psychological Review 104(2), 211–240 (1997)
15. Kanerva, P.: Sparse distributed memory. The MIT Press, Cambridge (1988)
16. MacQueen, J.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
17. Halkidi, M., Vazirgiannis, M., Batistakis, Y.: Quality Scheme Assessment in the Clustering Process. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 265–276. Springer, Heidelberg (2000)
18. Cunha, C.A., Bestavros, A., Crovella, M.E.: Characteristics of WWW Client Traces, Boston University Department of Computer Science, Technical Report TR-95-010 (April 1995)
19. The Internet Traffic Archive. `http://ita.ee.lbl.gov/index.html`
20. Gorman, J., Curran, J.R.: Random indexing using statistical weight functions. In: Proceedings of EMNLP 2006, pp. 457–464 (2006)

21. Teng, W., Chang, C., Chen, M.: Integrating Web Caching and Web Prefetching in Client-Side Proxies. IEEE Trans. Parallel Distr. Syst. 16(5), 444–455 (2005)
22. Nanopoulos, A., Katsaros, D., Manolopoulos, Y.: Effective Prediction of Web-User Accesses: A Data Mining Approach. In: Proceeding of Workshop WEBKDD (2001)
23. Wu, Y., Chen, A.: Prediction of web page accesses by proxy server log. World Wide Web 5, 67–88 (2002)