ISSN:0975-9646

Random Indexing Based Web User Clustering for Faster Navigation

Dr.J.K.R Sastry, K. Ruth Ramya, M. Devi Kavya Priya
Computer Science Department, KL University
Vaddeswaram, Guntur, District, Andhra Pradesh, India, 522502

Abstract— **Users of a Web site usually perform their interest-oriented actions by click-ing or visiting Web pages, which are traced in access log files. Clustering Web user access patterns may capture common user interests to a Web site, and in turn, build user profiles for advanced Web applications, such as Web caching and prefetching. The conventional Web usage mining techniques for clustering Web user sessions can discover usage patterns directly, but cannot identify the latent factors or hidden relationships among users' navigational behavior. In this paper, we propose an approach based on a vector space model, called Random indexing, to discover such intrinsic characteristics of Web users' activities. The underlying factors are then utilized for clustering individual user navigational patterns and creating common user profiles. The clustering results will be used to predict and prefetch Web requests for grouped users. We demonstrate the usability and superiority of the proposed Web user clustering approach through experiments on a real Web log file. The clustering and prefetching tasks are evaluated by comparison with previous studies demonstrating better clustering performance and higher prefetching accuracy.**

Keywords— **Web user clustering, User behavior, Random Indexing, Web prefetching.**

I. INTRODUCTION

The World Wide Web [1] continues to grow at an astounding rate in both the sheer volume of traffic and the size and complexity of web sites. The complexity of tasks such as web site design, web server design, and of simply navigating through a web site have increased along with this growth. An important input to these design tasks is the analysis of how a web site is being used. Usage analysis [2] includes straightforward statistics, such as page access frequency, as well as more sophisticated forms of analysis such as finding the common traversal paths through a web site.

Web users may exhibit various types of behaviors associated with their information needs and intended tasks when they are navigating a Web site which are traced in Web access log files. There are mainly two issues regarding these user behaviors. First issue is web users are given unique identification each time when they log on to the web. This becomes difficult to identify each user which rises to the solution of assigning a specific identifier for each user that uniquely identifies him every time he logs on to the web.

Second issue is searching for the information according to user's frequent need and intention which is becoming difficult. As a result, it is taking lot of time to find their frequently needed information which lead to the branch of web usage mining. Some of the data mining algorithms that are commonly used in web usage mining are association rule generation, clustering and sequential pattern generation. We should be able to detect more compact or well-separated user groups. Based on common profiles of these detected clusters or groups, prediction and pre-fetching user requests can be done with encouraging results.

II. THEORITICAL SURVEY

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. Some standard techniques of date mining such as **fuzzy clustering [3]** algorithms, **first-order Markov models [4]** and the **Dempster-Shafer theory [5]** have been introduced to find latent factors [14] by modelling Web users' navigation behavior and to cluster users based on Web access logs. Generally, these techniques capture stand alone user behaviors at the page view level.

The common procedure of Web user clustering based on user navigational patterns and their behaviour is illustrated below and shown in the Fig 1:

A.Fuzzy Clustering:

In hard clustering, data is divided into distinct clusters, where each data element belongs to exactly one cluster. In **fuzzy clustering** [3] (also referred to as **soft clustering**), data elements can belong to more than one cluster, and associated with each element is a set of membership levels. These indicate the strength of the association between that data element and a particular cluster. Fuzzy clustering is a process of assigning these membership levels, and then using them to assign data elements to one or more clusters.

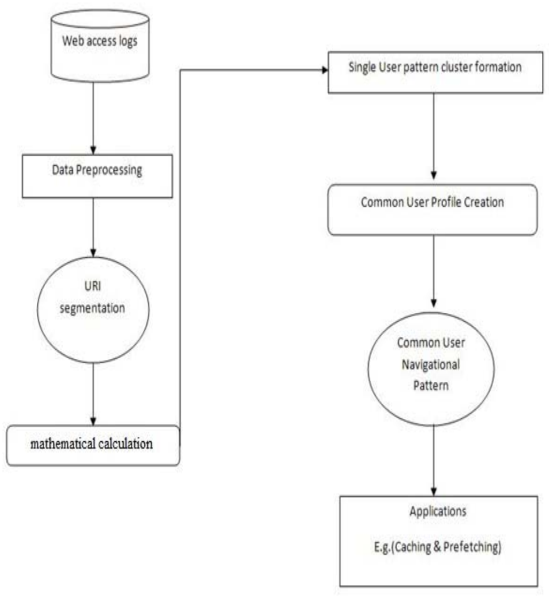
Web access logs
Data Preprocessing
URI segmentation
mathematical calculation
Single User pattern cluster formation
Common User Profile Creation
Common User Navigational Pattern
Applications
E.g.(Caching & Prefetching)

Fig. 1 Generic cluster formation flow

Problems:

- The algorithm minimizes intra-cluster variance as well, but has the same problems as *k*-means; the minimum is a local minimum, and the results depend on the initial choice of weights.
- Using a mixture of Gaussians along with the expectation-maximization algorithm is a more statistically formalized method which includes some of these ideas: partial membership in classes.
- Fuzzy c-means has less accuracy of clustering under noise.

B.First-Order Markov Models:

Markov model [4] and clustering are two frameworks used for predicting the next page to be accessed by the Web user. Markov models are becoming very commonly used in the identification of the next page to be accessed by the Web site user based on the sequence of previously accessed pages.

Let P = {p1, p2, …, pm} be a set of pages in a Web site. Let W be a user session including a sequence of pages visited by the user in a visit. Assuming that the user has visited ***l*** pages, then prob(pi|W) is the probability that the user visits pages pi next. Page p***l***+1 the user will visit next is estimated by:

$$P_{l+1} = \arg\max_{p\in P}\{P(P_{l+1} = p|W)\} = \arg\max_{p\in P}\{P(P_{l+1} = p|pl, pl-1, \ldots, p1)\}$$

Problems:

However, longer k causes the following two problems: The coverage of model is limited and leaves many states uncovered; and the complexity of the model becomes unmanageable. Therefore, the following are three modified Markov models for predicting Web page access.

- All kth Markov model: This model is to tackle the problem of low coverage of a high order Markov model. For each test instance, the highest order Markov model that covers the instance is used to predict the instance.
- Frequency pruned Markov model: Though all kth order Markov models result in low coverage, they exacerbate the problem of complexity since the states of all Markov models are added up. The removal of these low frequency states affects the accuracy of a Markov model. However, the number of states of the pruned Markov model will be significantly reduced.
- Accuracy pruned Markov model: Frequency pruned Markov model does not capture factors that affect the accuracy of states. A high frequent state may not present accurate prediction. When we use a means to estimate the predictive accuracy of states, states with low predictive accuracy can be eliminated. One way to estimate the predictive accuracy using conditional probability is called confidence pruning. Another way to estimate the predictive accuracy is to count (estimated) errors involved, called error pruning.

C.Dempster-Shafer theory:

Dempster-Shafer's theory [5] of combining evidence has attracted considerable attention as a promising methodfor dealing with some problems arising in combining of evidence and data fusion. It starts by assuming a Universe of Discourse U, also called Frame of Discernment, which is a set of mutually exclusive alternatives. The frame of discernment can consist of the possible values of an Iatt tgriibvuetse .t o each subset A of U a **basic probability assignment** (***bpa***) *m*(A), which represents the strength of some evidence. For the empty set, *m* is 0; the sum of *m* over all subsets of U is 1. That is:

$$m(\phi) = 0 \quad \text{and} \quad \sum_{A_i \subseteq U} m(A_i) = 1.$$

The basic probability assignment *m* is referred to as *mass distribution* to distinguish it from the probability distribution.

Problems:

This work still has several research issues, which we plan to address in the future. First, usage data by itself is not sufficient for recommendation. The personalization and recommendation process needs to have specific knowledge about the particular domain to do anything besides filtering based on statistical attributes of the discovered rules or patterns. Another problem is the scalability problem. Usage data collection on the Web is incremental. Hence, there is a need for mining algorithms to be scalable. They should be able to take as input the existing data, and mined knowledge, as well as the new data, and develop a new model in an efficient manner. Our future work will address these problems.

III. IMPLEMENTATION

The main implementation process consists of 3 steps. They are data preprocessing, random indexing and clustering. This can be explained in Algorithm 1: as per the process, algorithm can be explained as follows: firstly, URLs are split into segments S and the users are formed as {U1, U2,..., Un}. Now a N*d context window is formed using random indexing. Then a clustering approach (k-means) is applied on the window producing a group of users having similar interest patterns.

Algorithm 1: Random Indexing Based Clustering
Input: User IDs, Session IDs, URLs
Output: Common User Profile containing Common User Navigation Patterns
1: Obtain the user interested set of URLs P= {URL1, URL2....URLm} from web access log.
2: Based on P obtain a navigation set for individual users, U = {U1, U2, . . ., Un}, which contains pages requested by each user.
3: Obtain n*d Context Window by implementing Random Indexing Approach
4: Perform k-means clustering and obtain common user profiles containing set of users having similar interest patterns.

A.Data preprocessing

The first part of Web user cluster detection, called preprocessing [2], is usually complex and demanding. Generally, it comprises three domain dependent tasks: data cleaning, user identification, and session identification.

Step 1: Data cleaning

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) and common scripts (with filename sufixes such as js, css or cgi), which are not content pages or documents, need to be removed. In general, a user does not explicitly request all of the graphics that are on a Web page and automatically downloaded. Since the main intent of Web Usage Mining is to get a picture of the uses' behaviour, it does not make sense to include file requests that the user did not explicitly request. Duplicated requests are also filtered out in this step, leaving only one entry per page request.

Step 2: 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. The most obvious assumption is that a single user in Web logs acquired from the server and proxy sides are identified by the same IP address. However, this is not very accurate because, for example, a visitor may access the Web from different computers, or many users may use the same IP address (if a proxy is used). This problem can be partially solved by the use of cookies, URL rewriting, or the requirement for user registration. User identification from client-side logs is much easier because these logs are traced via different user IDs. Since we take a log file from the client side, users are identified according to their IDs.

Step 3: 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 timeout. If the time between page requests exceeds a certain limit of access time, we assume a user is starting a new session. Based on empirical investigations this time limit has been found to be 25.5 minutes. Many commercial products, however, use 30 minutes as a default timeout. Besides, web browsers may also request content on a regular time frequency based on requests from the page. For example, www.cnn.com uses the \http-equiv" html tag to indicate that the page should be refreshed every 30 minutes. We will also use 30 minutes in our investigations.

B.Random indexing

Random Indexing (RI) [12], [15] is a word co-occurrence based approach to statistical semantics. RI uses statistical approximations of the full word co-occurrence data to achieve dimensionality reduction. This results in a much quicker running time and fewer required dimensions. 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(€) 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 :

$$\begin{array}{l} +1 \text{ for the probability} \\ 0 \text{ for the probability} \\ -1 \text{ for the probability} \end{array} \left\{ \begin{array}{l} (€/2)/d \\ d\text{-}€/d \\ (€/2)/d \end{array} \right.$$

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 *tn* in a '2+2' sized context window, *cm*, is represented by:

$$cm = [(wn-2)(wn-1)tn(wn+1)(wn+2)].$$

Here $l = 2$, and the context vector of *tn* in *cm* would be updated with:

$$Cm = R(wn-2) + R(wn-1) + R(wn+1) + R(wn+2),$$

where $R(x)$ is the random index vector of x. This process is repeated every time we observe *tn* in our data, adding the corresponding information to its existing context vector *C*. If the context *cm* is encountered again, no new index vector will be generated. Instead the existing index vector for *cm* is added to *C* to produce a new context vector for *tn*.the following algorithm 2 shows the 2 step process of random indexing including the URL generation and segment generation. {URL1, URL2, ..., URLn} are the URLs from the web access log after data preprocessing. {U1, U2,.....,Un} are the users and S is the one that contains segments of URLs. From these, obtain index vectors si(1,2, ...m) containing segments used by different URLs. As in next step context vectors are generated, these are done by G=(Si U Uj) =n × d matrix A = {u1, u2, . . . , un}T where each row as the context vector uj of each single user.

Algorithm 2: context vector generation

Input: User IDs, Session IDs, URLs

Output: A n*d Context Window A

```
1: Obtain the user interested set of URLs
       P={URL1,URL2....URLm} from web access log.
2: Based on P obtain a navigation set for individual users, U
       = {U1, U2, . . ., Un}, which contains pages requested
       by each user.
3: Obtain S which contains all the segments that have
       occurred in P by splitting all the URLs in the user
       interest page set, P, by "/".
4: For each segment, obtain a d-dimensional index vector si
       (i = 1, 2, . . ., m, where m is the total number of
       segments).
5: Scanning the navigation set U, for each segment
       appearing in one user, update its zero- initialised
       context vector uj (j = 1, 2, . . ., n, where n is the total
       number of users).
6: G=(Si U Uj) =n × d matrix A = {u1, u2, . . . , un}T where
       each row as the context vector uj of each single user.
```

C. Single user pattern clustering

After random indexing of a 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 [6] partitions 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 of years. The k-means clustering technique can be described as follows:

Firstly, k initial centroids are randomly chosen. Each data point is then assigned to the closest centroid, and each collection of points assigned to a centroid forms a cluster. The centroid of each cluster is then updated as the mean of points assigned to the cluster. The assignment and update steps are repeated until no point changes clusters, or equivalently, until the centroids remain the same. Euclidean distance is used in our k-means experiments. Here is the algorithm 3 that explains k-means clustering taking K as number of clusters, chosen cluster centers {z1, z2,... ,zk} and final clusters formed as result are placed into {C1, C2,....Ck}.

```
Algorithm 3: K-Means Clustering
Input: n*d Context Window A
Output: Common User Profile containing Common User Navigation Patterns
1: Choose a value for K, the total number of clusters.
2: Randomly choose K points as final centers {z1, z2, , zk }.
3: Perform clusters {C1, C2, , Ck} by assigning the remaining instances to their closest cluster center.
4: Calculate a new cluster center for each cluster.
5: Repeat steps 3-5 until the cluster centers do not change.
```

IV. Cluster Validation

The problem of common clustering can be formally stated as follows. Given a sample data set X = {x1; x2; : : : ; xn}, determine a partition of the objects into k clusters C1;C2; : : : ;Ck. zi is the center of cluster Ci, which is represented by the average(mean) of all the points in the cluster. One of the most important issues of cluster analysis is the evaluation of clustering results to find the partitioning that best fits the underlying data. The procedure of evaluating the results of a clustering algorithm is known as cluster validity.

A. *Clustering validity measures*

In general terms, there are three approaches to investigate cluster validity. The first is based on external criteria, which evaluates the results of a clustering algorithm by comparing it to a pre-specified class label for the data set. The second is based on internal criteria, which evaluates the clustering results without any prior knowledge of the data sets. The third approach is based on relative criteria, which performs comparisons between cluster partitions by the same algorithm, that can be used to set various parameter values. There are two basic relative criteria proposed for clustering evaluation and selection of an optimal clustering scheme: Compactness and Separation. The third technique of clustering validity can also be used to choose the number of clusters in a data set.

B. *Methods for comparison*

We use the popular Web user clustering algorithm FCMdd as a comparison to RI-based Web user clustering. FCMdd is a fuzzy clustering based approach for Web user grouping and represents state-of-the-art using fuzzy clustering. The new optimisation based clustering algorithm called CAS-C is also employed for comparison. This method solves clustering problems from the perspective of chaotic optimisation and presents better Web user clustering results than the k-means clustering algorithm. Moreover, CAS-C represents an approach that differs from the other two, RI being a vector space based method and FCMdd being a fuzzy clustering method.

V. Conclusion

Previous methods focused on single user interested patterns which did not consider and find the hidden relationships between users. This paper focuses on discovering latent factors (hidden relationships) of users browsing behaviours using Random Indexing based Web Clustering that detects 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.

References

[1] Etzioni, O.: The world-wide Web: quagmire or gold mine? Communications of theACM 39(11), 65–68 (1996)

[2] Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wideweb browsing patterns. J. Knowl. Inf. Syst. 1(1), 5–32 (1999)

[3] 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)

[4] Cao, L.: In-depth Behavior Understanding and Use: the Behavior Informatics Approach.Information Science 180(17), 3067–3085 (2010)

[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., Z˙ ytkow, 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)