Knowl Inf Syst (2012) 33:89–115
DOI 10.1007/s10115-011-0453-x

REGULAR PAPER

Web user clustering and Web prefetching using Random Indexing with weight functions

Miao Wan · Arne Jönsson · Cong Wang · Lixiang Li · Yixian Yang

Received: 30 June 2010 / Revised: 1 July 2011 / Accepted: 22 October 2011 /
Published online: 30 November 2011

Abstract Users of a Web site usually perform their interest-oriented actions by clicking 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 behaviour. 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 utilised 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 · Random Indexing · Weight functions · Web prefetching

1 Introduction

Web Mining [14] is the area of data mining which deals with the extraction of interesting knowledge from Web data repositories of WWW. Web access logs, available on most servers,

M. Wan (✉) · A. Jönsson
Department of Computer and Information Science, Linköping University, 581 83 Linköping, Sweden
e-mail: wanmiao120@163.com

A. Jönsson
e-mail: arnjo@ida.liu.se

M. Wan · C. Wang · L. Li · Y. Yang
Information Security Center, Beijing University of Posts and Telecommunications,
P.O. Box 145, 100876 Beijing, China

are good examples of such data repositories used in Web Mining. Generally, 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 can be traced in the Web access log files of the Web site that the user visited.

Web usage mining [12], which captures navigational patterns of Web users from log files, has achieved great success in various fields, such as personalisation of Web content and user recommendation [5,32], prefetching and caching pages for user navigation [21,45], improvement of Web design [3,16] and e-commerce [2]. Most of the research efforts of Web usage mining focus on three main paradigms: association rules, sequential patterns and clustering.

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 data mining such as fuzzy clustering algorithms [28,33], first-order Markov models [7] and the Dempster-Shafer theory [47] have been introduced to model Web users' navigation behaviour and cluster users based on Web access logs. Three clustering algorithms have been compared in [36] to analyse their performance. 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 [30], have been widely used to discover the latent relationship from Web linkage information, which can be used to find relevant Web pages to improve Web searching efficiency and effectiveness [19,20]. In addition, some works [15,23,49] have been used to derive user access patterns and Web pages from various types of Web data, by utilising a so-called Probabilistic Semantic Latent Analysis (PLSA), which is based on the maximum likelihood principle from statistics.

Random Indexing [24] 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 [10,24,39]. 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. Moreover, in many NLP tasks, including distributional similarity, statistical weighting has been used to improve performance. The Random Indexing algorithm can be modified to utilise weighted contexts [17].

In this paper, we propose a Web user clustering approach to prefetch Web pages for grouped users based on Random Indexing (RI). Segments split by '/' in the URLs will be used as the unit of analysis in our study. The Random Indexing model is constructed to uncover the latent relationships among segments of different users and extract individual user access patterns from the Web log files. Furthermore, to improve the performance of Random Indexing, we modify it with statistical weighting functions for detecting groups of Web users. Common user profiles can be created after clustering single-user navigational patterns. To demonstrate the usability of weighted-RI for user cluster detection, we also apply our algorithm to a real prefetch task to predict future requests of clustered users according to their common pages. Our clustering and prefetching approaches based on weighted-RI are compared to a popular Web user clustering method named FCMdd [28] and a new proposed clustering algorithm called CAS-C [45]. The experimental results show that the weighted-RI-based Web user clustering techniques present more compact and well-separated clusters than FCMdd and CAS-C and get higher prefetching accuracy as well.

2 Random Indexing using weight functions

In this section, we describe the Random Indexing technique and the various weighting functions used in our work.

2.1 Random Indexing (RI)

Random Indexing is a vector space technique proposed by Kanerva et al. [24] which provides an efficient and scalable approximation to distributional similarity problems. As an alternative to Singular Value Decomposition (SVD) for Latent Semantic Analysis (LSA) [30], Random Indexing is an incremental word space model based on sparse distributed representations [24–26]. In the distributional hypothesis, words that occur in similar contexts have similar meanings so that a *word* is the sum of its contexts and the *context* is the sum of its words, where the *context* can be defined as the surrounding words or the entire document. 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) can be in the order of thousands, 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 one of these values with the following probability [39]:

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

Step 2 *Context vectors* are produced by scanning through the text. Each time a word occurs in a context (e.g. in a document, or within a sliding context window), 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 effectively 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. The number of adjacent words in a context window is called the context window size, l. For example, assume that term t_n appears in a '2+2' sized context window, w_n, as represented by:

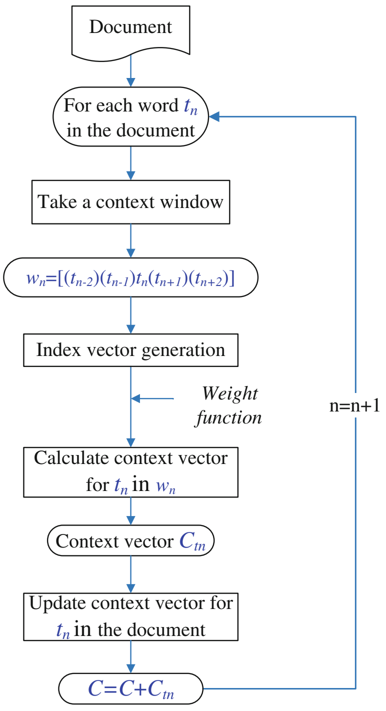
$$w_n = [(t_{n-2})(t_{n-1})t_n(t_{n+1})(t_{n+2})].$$

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

$$C_{t_n} = R(t_{n-2}) + R(t_{n-1}) + R(t_{n+1}) + R(t_{n+2}),$$

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 w_n is encountered again, no new index vector will be generated. Instead the existing index vector for w_n is added to C to produce a new context vector for t_n.

Fig. 1 Working process of Random Indexing based on weight functions


Document
For each word t_n in the document
Take a context window
w_n=[(t_{n-2})(t_{n-1})t_n(t_{n+1})(t_{n+2})]
Index vector generation
Weight function
Calculate context vector for t_n in w_n
Context vector C_{tn}
Update context vector for t_n in the document
$C=C+C_{tn}$
n=n+1

2.2 Weighted-RI

Random Indexing performs poorly on tasks with dramatically increasing volume of raw input data [13]. One way to handle this is to revise Random Indexing to use weight functions. The context vector of a term t is then created by the weighted sum of each of its attributes.

Statistical weighting is used, for instance, to improve performance in many natural language-processing (NLP) tasks. In NLP, the context relation weight function is designed to assign higher values to contexts that are more indicative of the meaning of that word [13]. Following this notation, a context relation is defined as a tuple (t, r, t') where t is a term which occurs in some grammatical relation r with another word t' in some sentence. We refer to the tuple (t, r, t') as an attribute of t.

Weights are generated using the statistical information (e.g. frequency) of each term and its contexts. Rather than the context vector being generated by adding each individual context, it is generated by adding each index vector for each unique context multiplied by its weight.

Thus, the context vector of each term t is calculated as:

$$C_t = \sum_{(r,t') \in (t,*,*)} R(r, t')\text{weight}(t, r, t') \ , \tag{1}$$

where $R(r, t')$ is the index vector of the context (r, t') and $weight(t, r, t')$ is the weight function for term t and its context.

Figure 1 depicts the algorithmic structure of weight-based Random Indexing.

In the investigations presented in this article, the weight functions include simple frequency functions, approaches from information retrieval and weight functions from existing systems [13]:

$$\begin{array}{llll}
\textit{Identity} & 1.0\ \text{(Original RI)} & \textit{Freq} & f(t,r,t') \\
\textit{RelFreq} & \dfrac{f(t,r,t')}{f(t,*,*)} & \textit{Tf}-\textit{Idf} & \dfrac{f(t,r,t')}{n(*,r,t')} \\
\textit{LogTf}-\textit{Idf} & \dfrac{\log_2(f(t,r,t')+1)}{\log_2\left(1+\frac{N(r,t')}{n(*,r,t')}\right)} & \textit{MI} & \log\left(\dfrac{p(t,r,t')}{p(w,*,*)p(*,r,t')}\right) \\
\textit{Gref94} & \dfrac{\log_2(f(t,r,t')+1)}{\log_2(n(*,r,t')+1)} & \textit{Lin98A} & \log\left(\dfrac{f(t,r,t')f(*,r,*)}{f(t,r,*)f(*,r,t')}\right) \\
\textit{Lin98B} & -\log\left(\dfrac{n(*,r,t')}{N_t}\right) & \textit{Dice} & \dfrac{2p(t,r,t')}{p(t,*,*)+p(*,r,t')}
\end{array}$$

where $f(t,r,t')$ is the frequency of a tuple, that is the number of times a term appears in a context, $f(t,*,*)$ is the instance or token frequency of the contexts in which t appears and $n(t,*,*)$ is the number of attributes of t. f, p, n and N in the weighting functions are calculated as:

$$\begin{aligned}
f(t,*,*) &= \sum_{(r,t')\in(t,*,*)} f(t,r,t') \\
p(t,*,*) &= \frac{f(t,*,*)}{f(*,*,*)} \\
n(t,*,*) &= |(t,*,*)| \\
N_t &= |\{t|n(t,*,*)>0\}|
\end{aligned}$$

More detailed explanations of the applied weight functions are illustrated in [13].

3 Web user clustering based on weighted-RI

There are several preprocessing tasks and modelling techniques that must be performed prior to applying data mining algorithms to the raw data collected from Web logs. In this section, we present the process of Web user clustering based on weighted-RI and express how Web log data for each user is processed.

The procedure of Web user clustering based on weighted-RI is illustrated in Fig. 2 and will be outlined in more detail below.

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.

3.1.1 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) and common scripts (with file name suffixes 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

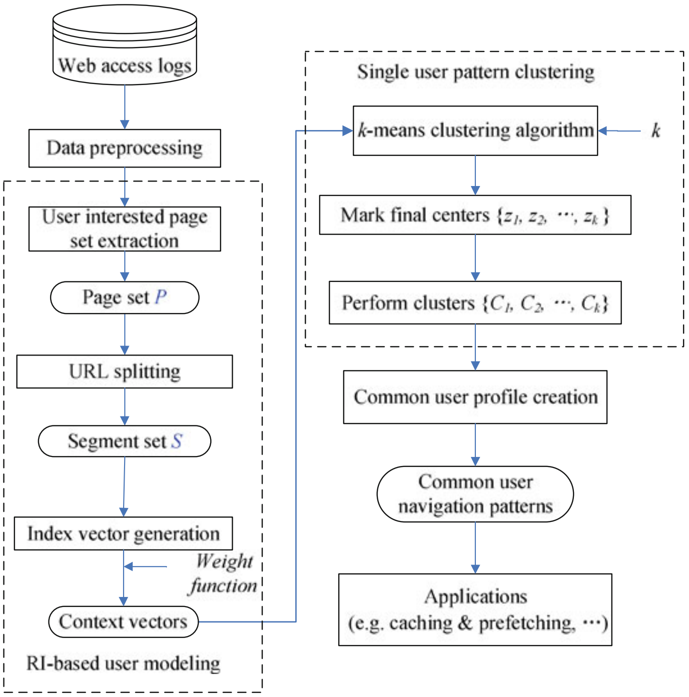

Web access logs
Data preprocessing
User interested page set extraction
Page set P
URL splitting
Segment set S
Index vector generation
Weight function
Context vectors
RI-based user modeling
Single user pattern clustering
k-means clustering algorithm
k
Mark final centers $\{z_1, z_2, \cdots, z_k\}$
Perform clusters $\{C_1, C_2, \cdots, C_k\}$
Common user profile creation
Common user navigation patterns
Applications (e.g. caching & prefetching, ⋯)

Fig. 2 Working flow of Web user clustering approach based on weighted-RI

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 [12]. Duplicated requests are also filtered out in this step, leaving only one entry per page request.

3.1.2 User identification

Identifying different users is an important issue of data preprocessing. There are several ways to distinguish individual visitors in Web log data that 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 is 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 [11], URL rewriting [32] or the requirement for user registration [1]. 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.

3.1.3 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 min [8]. Many commercial products, however, use 30 min as a default timeout [12]. 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' $\langle meta \rangle$ tag to indicate that the page should be refreshed every 30 min [1]. We will also use 30 min in our investigations.

3.2 User modelling based on weighted-RI

After all the Web logs are preprocessed, the log data are further analysed to find common user features to create a proper user model for user clustering. The established matrix of the user model will be the input to the Web user clustering algorithm.

3.2.1 Navigation set of individual users

Based on the results of the user identification, it is possible to assign to every user in the access logs a set of Web pages that s/he has visited. Pages that are requested by a user in only a very small period, such as one session, and not visited anymore, represent temporary user interest and are filtered out. Pages or URLs requested in more than 2 sessions by a user, reflect to some extent the steady-going interests of this user and are selected as user interest pages.

Since pages with very low hit rates in the log file only reflect the personal interest of individual users, these pages should be removed based on the preset number of the user or the host. After the process of low support pages filtering, we will get 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 preset number of times. This user interest page set, P, implicates behaviour features of Web users and is suitable for clustering analysis. Based on the user interest page set, P, we create a navigation set for individual users, $U = \{U_1, U_2, \ldots, U_n\}$, where each element contains pages requested by a single user.

3.2.2 Segmentation of URLs

The form of a Web page's URL contains some useful information. According to the hierarchical structure of most Web sites, URLs can be seen as composed on different levels, which are reflected in the sequence of segments split by '/'. For example, 'http://cs-www.bu.edu/faculty/gacs/courses/cs410/Home.html' may represent the homepage of a course named 'cs410' and that 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.

3.2.3 Random Indexing with different weights for each user

As shown in the previous subsection, user sessions are composed of user interest pages, and all the page URLs are formed by different segments. Each segment can be seen as a word, and each user can be seen as a document. For each segment s_i ($i = 1, 2, \ldots, q$, where q is the total number of segments) in S, a d-dimensional index vector R_i is generated. We then use the 10 statistical functions in Sect. 2.2 to weight each index vector. Here, s_i is taken as the term t in the tuple (t, r, t'), and two kinds of relationships, *before* and *after*, are selected as r in the tuple. For example, (courses, before, cs410) indicates that 'course' is located just before 'cs410' as a pattern of 'courses/cs410' in an URL. As a result, the function symbols in Sect. 2.2 can be explained as follows:

$f(t, *, *)$ is the context frequency for one single user; $f(*, *, *)$ is the total number of segments for an individual user; $n(t, *, *)$ represents the number of users in which t appears; N_t is the total number of $(t, *, *)$.

Thus, for each segment, s_i, appearing in a user's session, we calculate its weight using the statistical weighting functions in Sect. 2.2 and update its zero-initialised context vector u_j ($j = 1, 2, \ldots, n$, where n is the total number of users) by adding its random index vector R_i multiplied by the corresponding weight of each segment in the context window, where the size of the context window is preset. Finally, a set of individual users' navigation patterns, which forms an $n \times d$ matrix $A = \{u_1, u_2, \ldots, u_n\}^T$, is created with each row as the context vector, u_j, of each user.

3.3 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 [31] 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 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.

4 Clustering validity

The problem of common clustering can be formally stated as follows. Given a sample data set $X = \{x_1, x_2, \ldots, x_n\}$, determine a partition of the objects into k clusters $C_1, C_2, \ldots, C_k$. z_i is the centre of cluster C_i, 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.

4.1 Clustering validity measures

In general terms, there are three approaches to investigate cluster validity [40]. The first is based on external criteria, which evaluates the results of a clustering algorithm by comparing it to a prespecified 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 $Separation$ [4]. The third technique of clustering validity can also be used to choose the number of clusters in a data set.

Since the number of clusters should be preset for the k-means clustering algorithm, we use a relative criteria named SD in this paper 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 two internal validity measures, called SSE and β, to evaluate the performance of the proposed clustering algorithms with different parameter values. However, some internal indices depend on the problems' parameters [22], such as the number of patterns, features and clusters. Square error, for example, naturally decreases as the number of clusters increases. Thus, it is unfair to use the SSE and β indices for evaluating performance of clustering algorithms with different numbers of clusters. Instead, we introduce a relative index called CS for comparing the clustering results of different algorithms. In what follows we present the definitions for each validity measure used in this paper.

– 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

$$\mathrm{SD}(k) = Dis(k_{\max}) \cdot \mathrm{Scat}(k) + Dis(k), \tag{2}$$

where $k_{\max}$ is the maximum number of input clusters and influences slightly on the value of SD [18].
Scat is the average scattering within one cluster and is defined as:

$$\mathrm{Scat}(k) = \frac{1}{k}\sum_{i=1}^{k} \|\sigma(C_i)\| / \|\sigma(X)\|, \tag{3}$$

where $\sigma(S)$ represents the variance of a data set S.
Dis is the total scattering (separation) between clusters and is given by the following equation:

$$Dis(k) = \frac{D_{\max}}{D_{\min}} \sum_{i=1}^{k} \left(\sum_{j=1}^{k} \|z_i - z_j\| \right)^{-1}, \tag{4}$$

where $D_{\max} = \max(\|z_i - z_j\|)$ $(\forall i, j \in 1, 2, 3, \ldots, k)$ is the maximum distance between cluster centres and $D_{\min} = min(\|z_i - z_j\|)$ $(\forall i, j \in 1, 2, 3, \ldots, k)$ is the minimum distance between cluster centres.
Experiments show that the number of clusters, k, which minimises the SD index can be considered as an optimal value for the number of clusters present in the data set [18].

– Sum of Squared Error (SSE) is the common criteria of evaluating clustering results that sums the squared error of each data together. SSE is computed as

$$\mathrm{SSE} = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - z_i\|^2. \tag{5}$$

For each data in the given set, the error is the distance to the nearest cluster. Experiments show that the smaller the value of SSE, the better results the clustering approach will get [41].

– The beta index (β) computes the ratio of total variation and within class variation [37] and is defined as

$$\beta = \frac{\sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(X_{ij} - \overline{X}\right)^2}{\sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(X_{ij} - \overline{X_i}\right)^2}, \tag{6}$$

Table 1 Function description of each validity index

Index	Value	Function
SD	Smallest	Best k
SSE	Smallest	Best clustering results
β	Largest	Best clustering results
CS	Smallest	Best clustering results

where $\overline{X}$ is the mean of all the data points and $\overline{X_i}$ is the mean of the data points that belong to cluster C_i. X_{ij} is the jth data point of the ith cluster, and n_i is the number of data points in cluster C_i. Since the numerator of β is constant for a given data set, the value of β is dependent on the denominator only. The denominator decreases with homogeneity in the formed clusters. Therefore, for a given data set, the higher the value of β, the better is the clustering.

– The CS index computes the ratio of $Compactness$ and $Separation$.
$Compactness$ means that the members of each cluster should be as close to each other as possible. A common measure of compactness is the intra-cluster variance within a cluster, which should be minimised [4]. We take the average of variances of all clusters and call it $Comp$ which is defined as

$$Comp = \frac{1}{k}\sum_{i=1}^{k} \|\sigma(C_i)\|. \tag{7}$$

$Separation$ means that the clusters themselves should be widely spaced. There are three common ways measuring the distance between two different clusters [4]:
Single linkage that measures the distance between the closest members of the clusters.
Complete linkage that measures the distance between the most distant members.
Comparison of centroids that measures the distance between the centres of the clusters.
In this paper, we use the latter as measure of $Separation$. We calculate the average of all of these distances as follows

$$Sep = \frac{1}{k}\sum \|z_i - z_j\|^2, \quad i = 1, 2, \ldots, k-1, \quad j = i+1, \ldots, k. \tag{8}$$

A good clustering method should produce clusters with high intra-class similarity and low inter-class similarity. Cluster results can be compared by taking the ratio between the $Compactness$ ($Comp$) and the $Separation$ (Sep):

$$\text{CS} = \frac{Comp}{Sep}. \tag{9}$$

It is clear that if the data set 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. Therefore, based on the definitions of CS, we can conclude that a small value of CS indicates compact and well-separated clusters.

To summarise, Table 1 presents the evaluation function of each validity index.

4.2 Methods for comparison

We take the popular Web user clustering algorithm FCMdd [28] as a comparison with 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 [45] is also employed for comparison. This method solves clustering problems from the perspective of chaotic optimisation and presents better Web user clustering performance than the *k*-means clustering algorithm [45]. 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.

In Sect. 5, the implemented RI-based Web user clustering will be presented and common user profiles will be created based on the clustering results. We will compare user profiles acquired by FCMdd, CAS-C and weighted-RI-based Web user clustering and analyse the results of them.

A prefetch application will be introduced in Sect. 6 to employ FCMdd, CAS-C and weighted-RI and evaluate their performance. Experiments based on the common user profiles will be set up to describe prefetch result comparisons between weighted-RI-based user clustering, FCMdd and CAS-C.

5 Experiments

In this section, we present our experiments of clustering Web users using RI weighted by various statistical functions and give a detailed investigation of the results. We use MatLab for our experiments.

5.1 Preprocessing of the 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 [9]. It was collected by the Oceans Research Group [35] at Boston University. The log file is available at The Internet Traffic Archive [42] sponsored by ACM SIGCOMM. It contains a total of 1,143,839 requests for data transfer, representing a population of 762 different users. The raw data in the access log have the following format:

$\langle$ *machine name*, *timestamp*, *user id*, *requested URL*, *size of document*, *bytes sent in reply* $\rangle$

We use the part of the logs during the period of January and February 1995. For session identification, we set the maximum elapsed time to 30 min, which is used in many commercial applications. 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 1,005 sessions from these 100 users. The User IDs are renumbered, and each one of them has been assigned an identification number between 1 and 100.

5.2 Parameter and weight function investigations

In this subsection, we present results from our investigations on the impacts of some key parameters and assign initial values for them. We also investigate 10 different statistical weight functions for Random Indexing.

5.2.1 Cluster number

Each single-user pattern matrix generated by the 10 different weighted-RI approaches will be clustered by the *k*-means clustering algorithm. First, we need to find the proper *k* value for the *k*-means algorithm for each user pattern matrix.

We have conducted k-means clustering experiments for 10 different user pattern matrixes by measuring SD values using various values of k. The maximum value of the cluster number $k_{\max}$ can be chosen as $\sqrt{n}$ (n is the data size) [48]. So we set $k_{\max} = 10$ in our work. Each 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 Fig. 3.

As can be seen in Fig. 3, 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 $k = 7, k = 5, k = 8, k = 8, k = 9, k = 4, k = 4$ and $k = 3$ as the optimal cluster number for $Identity$, $Freq$, $RelFreq$, $TfIdf$, MI, $Lin98A$, $Lin98B$ and $Dice$, respectively. For the matrixes processed by $LogTfIdf$ and $Gref$, we get more than one suitable k value. Since 4 lines reach the bottom at $k = 7$ and $d = 600$ gets the smallest SD value among the 6 lines in Fig. 3g, we select $k = 7$ for $Gref$. We also select $k = 7$ for $LogTfIdf$ as the minimum value of SD appears at $k = 7$ for 3 lines ($d = 300$, 500 and 600) and $d = 500$ acquires the best SD in Fig. 3e.

We perform similar experiments for the FCMdd and CAS-C algorithms as depicted in Figs. 4, 5. We use $k = 8$ for FCMdd and $k = 7$ for CAS-C.

5.2.2 Weights

We calculate values of SSE and β to compare user clustering performance using Random Indexing with the different weight functions presented in Sect. 2.2. Figure 6 shows the comparison results of the 10 statistical functions for RI.

From Fig. 6, we can see that approaches that used MI, $RelFreq$ and $Gref$ have smaller SSE values and larger β values than $Identity$ at all dimensions, and MI gets the smallest SSE and the largest β among all the weight functions.

From the above results we can conclude that given a certain dimension of Random Indexing, utilising MI, $RelFreq$ and $Gref$ can improve the Web user clustering performance of the original Random Indexing technique. In Sect. 6, we will apply RI with these three weighting functions as well as $Identity$ (i.e. the original Random Indexing) for the prefetch application.

5.2.3 Dimensionality

As four weighting functions ($Identity$, MI, $RelFreq$ and $Gref$) have been selected in Sect. 5.2.2, we want to choose the proper dimension of index vectors for these four weighted Random Indexing techniques. In theory, the random projection family of dimension reduction techniques should give a better approximation to the original data matrix as the dimensionality of the random matrix increases [27]. In order to evaluate the effects of increasing the dimensionality of Random Indexing, we computed values of β and SSE using 12 different vector dimensions, with d ranging from 50 to 600. In these experiments, the performance measurers are reported using average values over 30 different runs. The results are depicted in Fig. 7.

From Fig. 7a, b we can see that the 4 weighted-RI methods reached the largest β and almost the smallest SSE at $d = 300$. As a result, $d = 300$ is chosen as the dimension of the index vectors used by the 4 weighted-RI techniques in our Web user clustering task.

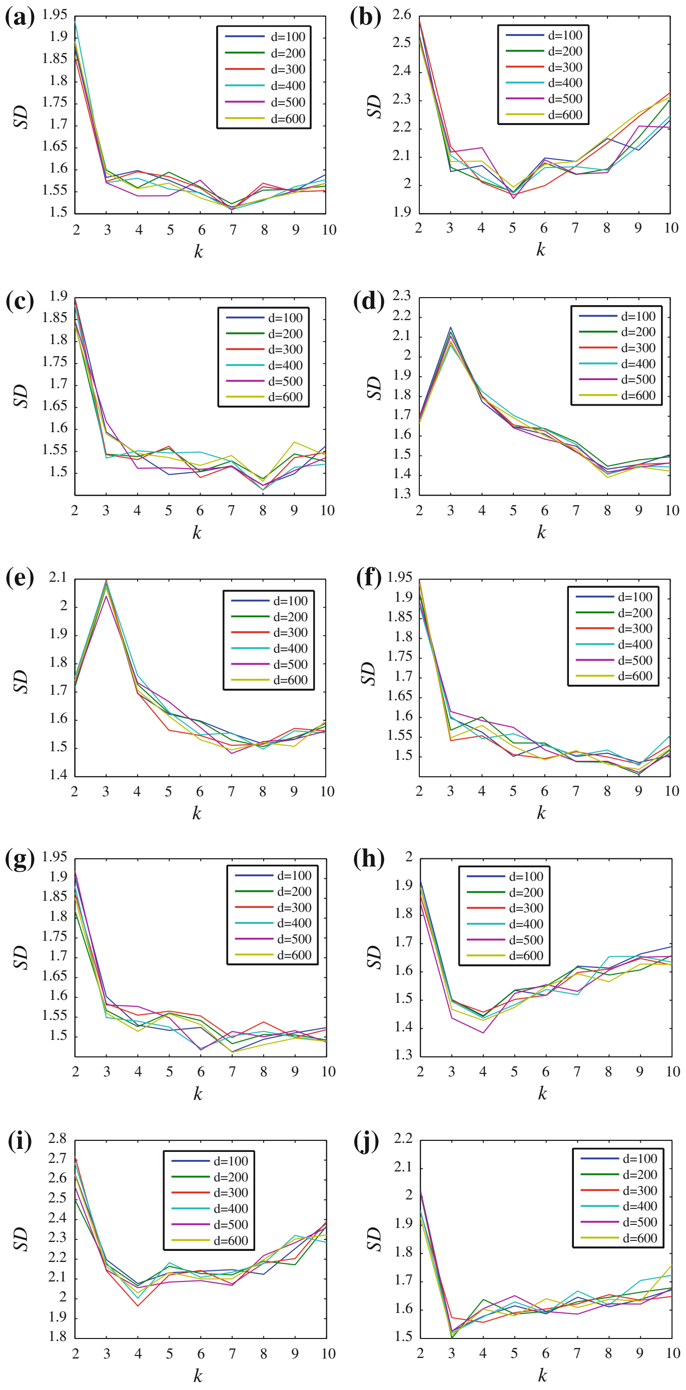


Fig. 3 Comparisons of SD measures for different weighting functions of Random Indexing in Web user clustering tasks. **a** *Identity*, **b** *Freq*, **c** *RelFreq*, **d** *TfIdf*, **e** *LogTfIdf*, **f** *MI*, **g** *Gref*, **h** *Lin98A*, **i** *Lin98B*, **j** *Dice*

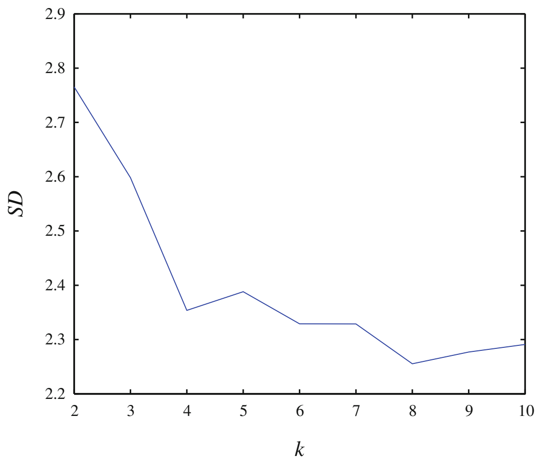


Fig. 4 Comparison of SD measures for FCMdd in Web user clustering tasks

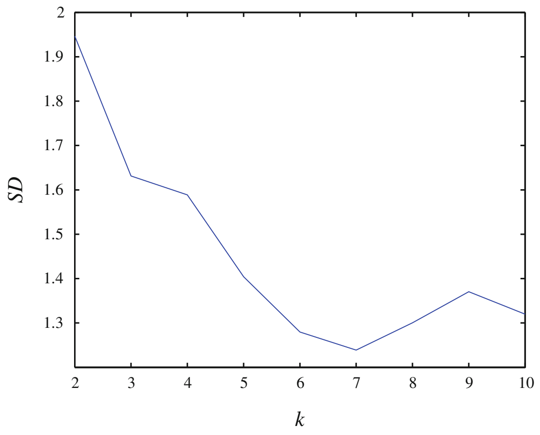


Fig. 5 Comparison of SD measures for CAS-C in Web user clustering tasks

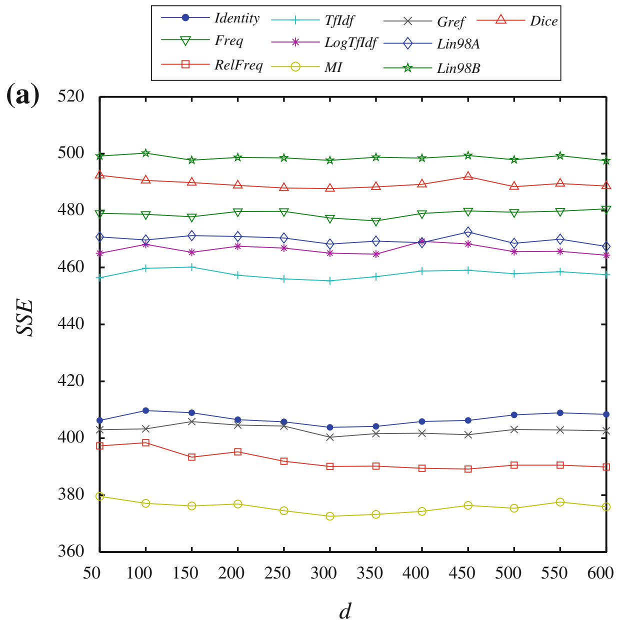
5.2.4 *Other parameters*

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 Gorman and Curran [17] and $l = 1$ as the URLs are rather short.

To summarise, we will use the values of the parameters for different weighted-RI in our experiments as presented in Table 2.

5.3 Single-user pattern matrix

After preprocessing and performing Random Indexing for the Web log data, we get a user interest page set P containing 97 requested URLs. These URLs are split by "/" to get the segment set S, which comprise 152 different segments. As the dimension of the index vector in



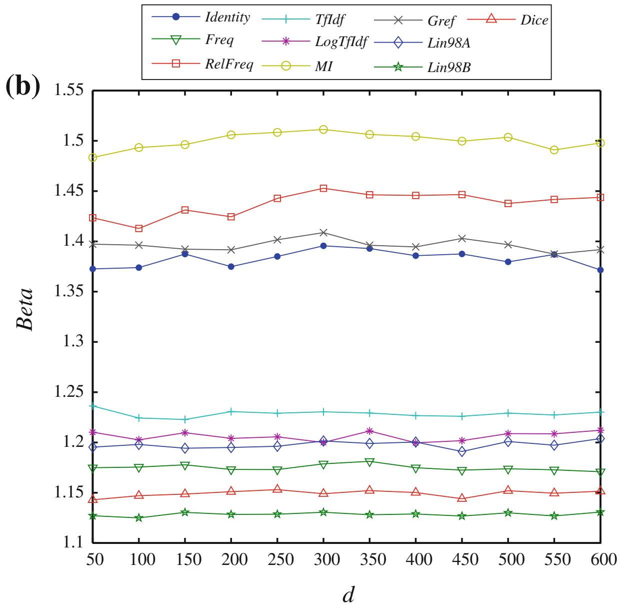
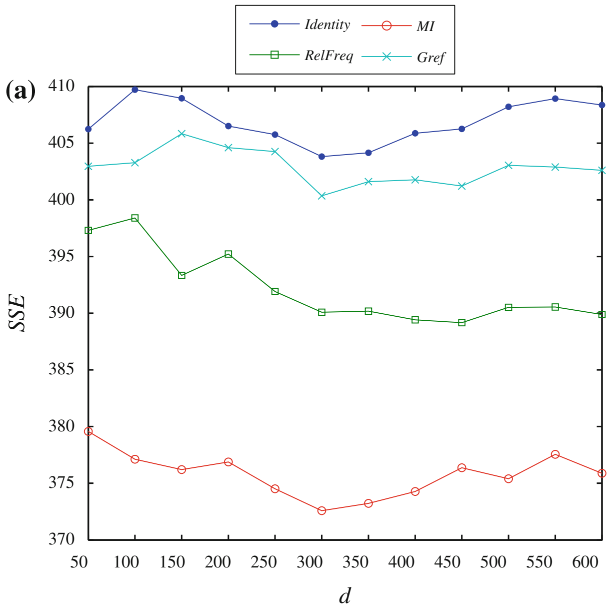


Fig. 6 The influence of different weight functions in Random Indexing on Web user clustering performance. **a** Comparison of SSE measures for different weight functions. **b** Comparison of β measures for different weight functions

RI is selected to 300 in Sect. 5.2.3, we construct a 100×300 matrix $A = \{A_1, A_2, \ldots, A_{100}\}^T$ as the single-user pattern matrix for each weighted-RI method and take it as input to the k-means clustering algorithm.



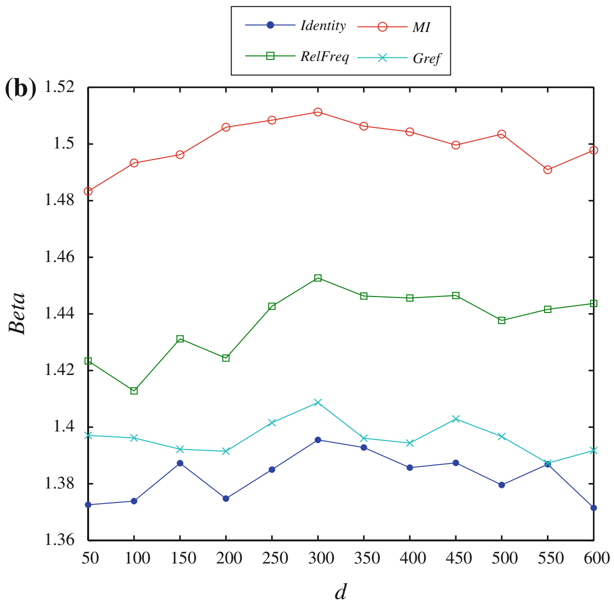


Fig. 7 The influence on Web user clustering performance with different dimension settings in weighted-RI techniques. **a** The influence of various d values to SSE. **b** The influence of various d values to β

5.4 User clustering results

After the log data are processed by Random Indexing using different weighting strategies ($Identity$, $RelFreq$, MI and $Gref$), the single-user navigation pattern matrix A will be clustered by the k-means clustering algorithm. Based on the validity measures in Sect. 4.1,

Table 2 Parameter values used in the experiments

Weight function	k	d	l	ϵ
$Identity$	7	300	1	10
$RelFreq$	8	300	1	10
MI	9	300	1	10
$Gref$	7	300	1	10

Table 3 Values of CS for different clustering approaches

Methods	k	$Comp$	Sep	CS
$FCMdd$	8	2.1423	5.7272	0.3741
$CAS-C$	7	2.2380	7.1574	0.3127
$Gref-RI$	7	2.2295	7.5605	0.2949
$Identity-RI$	7	2.2950	7.8054	0.2940
$RelFreq-RI$	8	2.2544	8.6938	0.2593
$MI-RI$	9	2.1116	10.2503	0.2060

we want to identify the best clustering scheme for each method. As we have mentioned in Sect. 4, CS index (Eq. 9) is used for comparing the performance of different clustering methods. The various weighted-RI methods for Web user clustering are compared to that generated using FCMdd [28] and CAS-C [45]. Table 3 presents the values of CS for the different clustering techniques.

As shown in Table 3, the MI-based RI clustering algorithm gets the smallest $Comp$ with the largest Sep, and of course, the best CS value. The other five methods get similar $Comp$ values but different Sep values. The $RelFreq$-based RI approach is second best because of its larger Sep value. The clustering algorithms FCMdd and CAS-C get the smallest values of Sep and the largest CS. Just from the CS comparison, we can see that the RI-based approaches perform better than FCMdd and CAS-C for clustering Web users. Moreover, the weighting-based RI techniques produce more compact clusters and separate better between clusters than the original RI approach, which demonstrates that the weighting functions have improved Random Indexing for Web user clustering. In the following section, we will set up a prefetch application using these six clustering approaches to further study Random Indexing and its weighting techniques.

6 Application: prefetching

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. Web caching and Web prefetching are two important techniques used to reduce the noticeable response time perceived by users [43]. The caching technique exploits the temporal locality, whereas the prefetching technique utilises the spatial locality of Web objects. An efficient caching and prefetching scheme effectively reduces the load and response time of Web servers. 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 [6,29,34], 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 [44,46]. Predicting groups of users interest have caught little attention in the area of prefetching.

6.1 Prefetching rule

Our prefetch task tries to exploit the advantage of spatial locality within groups of users. First, according to the clustering results in Sect. 5, we create common user profiles for FCMdd, CAS-C and the proposed RI-based Web user clustering algorithms. Then, for each algorithm, we identify commonly requested pages of most users in each cluster as the prefetching objects. As before, we will use the Web access logs of January and February for user clustering and request prediction. The accuracy of our prefetching scheme will be verified by comparing the predicted URLs with the access logs of March.

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. To compare our results with previous studies, we use the same $c = 0.5$ as in [45] for our prefetch task, which means that more than or equal to 50% of the users pages in one cluster which have been requested in January and February will be prefetched for March.

6.2 Experiments

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 cannot 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 that weights precision twice as much as recall.

Based on the clustering results, we build the common user profile for each RI method. Moreover, in order to evaluate RI with weighting functions for the Web user clustering task, we compare their user profiles to the ones generated using FCMdd and CAS-C. The common user profile created by *MI-RI* (the method with best clustering performance shown in Sect. 5.4) can be found in Table 5 of Appendix A.

From the common user profile acquired by the *MI*-based RI approach, we can find some information: (1) Different clusters of users can be identified by the common teachers or courses they selected, such as Clusters 2, 4 and 9 in Table 5; (2) Some groups of users are clustered by their common interests, such as Clusters 1, 5, 6, 7 and 8 in Table 5; (3) Many users only access the homepage of the department and the entry of courses to check information, such as Cluster 3 in Table 5; (4) About half of the users visited the course page

Table 4 Overall comparison of prefetch results for FCMdd, CAS-C and 4 weighted-RI techniques

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
$RelFreq$-RI	8	0.7812	0.4641	0.6873
$Identity$-RI	7	0.7540	0.5311	0.6956
$Gref$-RI	7	0.7994	0.4810	0.7059
MI-RI	9	0.8095	0.4678	0.7063

’cs-www.bu.edu/students/grads/tahir/CS111/’, ’Introduction to Computer Science’, which means that they selected this course or were interested in it; (5) Course pages gained the most attention at this Web site, because almost everyone had visited the homepage of courses; and (6) The entry page of this Web site, ‘cs-www.bu.edu’, had been accessed by all the clusters.

According to the common user profiles created by the four weighted-RI techniques, FSMdd and CAS-C, we set up prefetching experiments to prefetch URL requests for users in each cluster. We calculate the accuracy of the prefetch hits by comparing the predicted URLs with the access logs of March.

Table 4 gives the overall experimental comparison of prefetching for FCMdd, CAS-C and the weighted-RI techniques.

Comparing the top two lines to the last four rows of Table 4, we can see that the results in the proposed prefetching tasks achieve a total average precision of 75.40–80.61% and a total recall of 46.41–53.11%, which are all higher than 70.62% of CAS-C and 44.34% using FCMdd. Even the lowest $F_{0.5}$ value of the RI-based techniques, 0.6873 from *RelFreq-RI*, is 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 predicting service that is better than using FCMdd or CAS-C.

Then, we focus on the 4 different weighted-RI methods (rows 2–5 in Table 4). We can find that the *MI-RI* approach gets the highest total precision and the largest $F_{0.5}$ value, while the original RI (*Identity-RI*) method has the best recall. It is clear that the three weighted-RI techniques achieve higher precision than the *Identity-RI*-based approach although they get lower recall rate. The *MI-RI* and *Gref-RI* methods acquire the largest $F_{0.5}$ values (0.7063 and 0.7059) that are higher than 0.6956 of *Identity-RI*. As we have mentioned at the beginning of this subsection, we value precision higher than recall and use $F_{0.5}$ to measure the accuracy of prefetching. The *MI-RI*- and *Gref-RI*-based approaches then perform better than the original RI technique. Furthermore, *MI-RI* achieves the best performance among all the prefetching methods. The detailed prefetching results of *MI-RI* are shown in Table 6 of Appendix B.

To summarise, Random Indexing of user navigation patterns can improve the quality of user request prediction and show better results than FCMdd and CAS-C. Moreover, weighting functions further improve Random Indexing’s prefetch performance.

7 Conclusion

This paper focuses on discovering latent factors of user browsing behaviours based on Random Indexing with various weight functions and detecting clusters of Web users according to their activity patterns acquired from access logs. Experiments are conducted to investigate the performance of weighted-RI in Web user clustering tasks. The results show that the proposed RI-based Web user clustering approach could be used to detect user groups that are hardly found by other clustering algorithms. Based on common profiles of detected clusters, our approach is employed to predict and prefetch user requests with encouraging results.

Acknowledgments I would like to thank the editor and all the reviewers for their great supports to our work. This work is supported by the Program for New Century Excellent Talents in University of the Ministry of Education of China (Grant No. NCET-10-0239), the Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (Grant No. 121062), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20100005110002), the National Natural Science Foundation of China (Grant No. 60805043), the National Key Technologies R&D Program (Grant No. 2009BAH42B02) and Santa Anna IT Research Institute.

Appendices

A Common user profile created by *MI-RI* (Table 5)

Table 5 Common user profile created by Web user clustering algorithm using the MI-based RI approach

CN	Members	Common user requests
1	4, 19, 33, 40, 67, 76, 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/, www.ncsa.uiuc.edu/demoweb/url-primer.htm
2	13, 15, 18, 44, 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/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/
3	1, 12, 17, 25, 32, 42, 50, 72, 76, 77, 81, 82, 84, 88, 97, 99	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu:80/
4	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,

Table 5 continued

CN	Members	Common user requests
		cs-www.bu.edu/staff/people.html, cs-www.bu.edu:80/
5	2, 5, 20, 22, 23, 27, 29, 36, 37, 38, 39, 41, 43, 46, 47, 49, 51, 52, 53, 54, 56, 57, 58, 60, 62, 63, 64,68, 69, 73, 75, 79, 80, 85, 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/
6	9, 16, 24, 28, 31, 45, 55, 77, 78, 86	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, www.ncsa.uiuc.edu/SDG/Software/Mosaic/StartingPoints/ NetworkStartingPoints.html
7	3, 10, 14, 25, 48, 70, 87, 93	cs-www.bu.edu/, cs-www.bu.edu/courses/Home.html, cs-www.bu.edu/pointers/Home.html, cs-www.bu.edu/students/grads/tahir/CS111/, cs-www.bu.edu:80/
8	34, 35, 59, 100	akebono.stanford.edu/yahoo/, akebono.stanford.edu/yahoo/Entertainment/, cs-www.bu.edu/, cs-www.bu.edu/faculty/Home.html, cs-www.bu.edu/faculty/best/Home.html, cs-www.bu.edu/faculty/best/crs/cs101/Home.html, cs-www.bu.edu/pointers/Home.html, cs-www.bu.edu:80/
9	7, 8, 11, 21, 26, 30, 66, 74, 89	cs-www.bu.edu/, cs-www.bu.edu/courses/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/node1.html, cs-www.bu.edu:80/, cs-www.bu.edu:80/students/grads/oira/cs112/

The CN column represents the cluster number

B Prefetching results of *MI-RI* (Table 6)

Table 6 Prefetching results based on common profiles via the MI-based RI approach

Cluster	U_id	$Requests$	Pre_urls	$Hits$	$Precision$	Pre_avg	$Recall$	Rec_avg
	4	32		8	1.000		0.250	
	19	27		8	1.000		0.296	
	33	4		4	0.500		1.000	
1	40	5	8	5	0.625	0.839	1.000	0.584
	67	6		6	0.750		1.000	
	76	116		8	1.000		0.069	
	90	17		8	1.000		0.471	
	13	301		10	1.000		0.033	
	15	6		6	0.600		1.000	
2	18	51	10	10	1.000	0.933	0.196	0.459
	44	20		10	1.000		0.500	
	65	13		10	1.000		0.769	
	88	39		10	1.000		0.256	
	1	6		2	0.667		0.333	
	12	18		3	1.000		0.167	
	17	11		3	1.000		0.273	
	32	9		2	0.667		0.222	
	42	56		3	1.000		0.054	
3	50	16	3	3	1.000	0.788	0.188	0.281
	72	30		3	1.000		0.100	
	82	2		1	0.333		0.500	
	84	3		2	0.667		0.667	
	97	4		2	0.667		0.500	
	99	36		3	1.000		0.083	
	6	101		14	1.000		0.139	
4	61	9	14	5	0.357	0.625	0.556	0.518
	71	2		2	0.143		1.000	
	83	37		14	1.000		0.378	
	2	3		3	1.000		1.000	
	5	103		1	0.333		0.010	
	20	3		3	1.000		1.000	
	22	3		3	1.000		1.000	
	23	3		3	1.000		1.000	
	27	8		3	1.000		0.375	
	29	15		3	1.000		0.200	
	36	3		3	1.000		1.000	
	37	16		3	1.000		0.188	
	38	3		3	1.000		1.000	
	39	5		3	1.000		0.600	

Table 6 continued

Cluster	*U_id*	*Requests*	*Pre_urls*	*Hits*	*Precision*	*Pre_avg*	*Recall*	*Rec_avg*
	41	52		3	1.000		0.058	
	43	3		3	1.000		1.000	
	46	17		1	0.333		0.059	
	47	113		3	1.000		0.027	
	49	3		3	1.000		1.000	
	51	3		3	1.000		1.000	
	52	25		3	1.000		0.120	
5	53	3	3	3	1.000	0.917	1.000	0.603
	54	16		3	1.000		0.188	
	56	108		3	1.000		0.028	
	57	3		3	1.000		1.000	
	58	3		3	1.000		1.000	
	60	4		1	0.333		0.250	
	62	3		3	1.000		1.000	
	63	3		3	1.000		1.000	
	64	3		3	1.000		1.000	
	68	4		3	1.000		0.750	
	69	6		3	1.000		0.500	
	73	2		2	0.667		1.000	
	75	14		3	1.000		0.214	
	79	20		3	1.000		0.150	
	80	14		3	1.000		0.214	
	85	17		2	0.667		0.118	
	91	3		3	1.000		1.000	
	92	6		1	0.333		0.167	
	94	3		3	1.000		1.000	
	95	3		3	1.000		1.000	
	96	17		3	1.000		0.176	
	98	4		3	1.000		0.750	
	9	6		2	0.667		0.333	
	16	38		1	0.333		0.026	
	24	79		2	0.667		0.025	
	28	3		2	0.667		0.667	
	31	6		2	0.667		0.333	
6	45	3	3	2	0.667	0.636	0.667	0.249
	55	13		1	0.333		0.077	
	77	60		2	0.667		0.033	
	78	6		2	0.667		0.333	
	81	11		2	0.667		0.182	
	86	49		3	1.000		0.061	
	3	71		5	1.000		0.070	
	10	3		3	0.600		1.000	
	14	23		3	0.600		0.130	

Table 6 continued

Cluster	*U_id*	*Requests*	*Pre_urls*	*Hits*	*Precision*	*Pre_avg*	*Recall*	*Rec_avg*
7	25	34	5	4	0.800	0.825	0.118	0.431
	48	3		3	0.600		1.000	
	70	69		5	1.000		0.072	
	87	9		5	1.000		0.556	
	93	10		5	1.000		0.500	
	34	28		9	1.000		0.321	
8	35	21	9	9	1.000	1.000	0.429	0.453
	59	12		9	1.000		0.750	
	100	29		9	1.000		0.310	
	7	5		5	0.556		1.000	
	8	75		9	1.000		0.120	
	11	104		9	1.000		0.087	
	21	6		2	0.222		0.333	
9	26	59	9	9	1.000	0.691	0.153	0.632
	30	4		4	0.444		1.000	
	66	6		6	0.667		1.000	
	74	5		5	0.556		1.000	
	89	7		7	0.778		1.000	

The *Pre_urls* gives the number of URLs prefetched by the prefetching scheme; *Pre_avg* and *Rec_avg* represent the average *precision* and *recall* within one cluster, respectively

References

1. Anderson CR (2002) Amachine learning approach to web personalization. Ph.D. thesis, University of Washington
2. Ansari S, Kohavi R, Mason L, Zheng Z (2000) Integrating e-commerce and data mining: architecture and challenges. In: Proceedings of the 2001 IEEE international conference on data mining (ICDM 2001), pp 27–34
3. Berendt B (2002) Using site semantics to analyze, visualize, and support navigation. Data Min Knowl Discov 6(1):37–59
4. Berry MJA, Linoff G (1996) Data mining techniques for marketing, sales and customer support. Wiley, London
5. Bezerra BLD, de Assis Tenório de Carvalho F (2010) Symbolic data analysis tools for recommendation systems. Knowl Inf Syst (on-line)
6. Bundschus M, Yu Sh, Tresp V, Rettinger A, Dejori M, Kriegel H-P (2009) Hierarchical bayesian models for collaborative tagging systems. In: Proceedings IEEE international conference on data mining (ICDM 2009), pp 728–733
7. Cadez I, Heckerman D, Meek C, Smyth P, Whire S (2002) Visualization of navigation patterns on a website using model based clustering. Technical Report MSR-TR-00-18, Microsoft Research
8. Catledge LD, Pitkow JE (1995) Characterizing browsing strategies in the world-wide web. Comput Netw ISDN Syst 27:1065–1073
9. Characteristics of WWW Client Traces, Cunha CA, Bestavros A, Crovella ME (1995) Boston University Department of Computer Science. Technical Report TR-95-010. http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html
10. Chatterjee N, Mohan S (2008) Discovering word senses from text using Random Indexing. In: Gelbukh A (ed) Computational linguistics and intelligent text processing (Lecture Notes in Computer Science), CICLing 2008, LNCS 4919, pp 299–310
11. Cooley R (2000) Web usage mining: discovery and application of interesting patterns from web data. Ph.D. thesis, University of Minnesota

12. Cooley R, Mobasher B, Srivastava J (1999) Data preparation for mining world wide web browsing patterns. J Knowl Inf Syst 1(1):5–32
13. Curran JR (2004) From distributional to semantic similarity. Ph.D. thesis, University of Edinburgh
14. Etzioni O (1996) The world-wide Web: quagmire or gold mine. Commun ACM 39(11):65–68
15. Feng S, Wang D, Yu G, Gao W, Wong K (2010) Extracting common emotions from blogs based on fine-grained sentiment clustering. Knowl Inf Syst 24(1). doi:10.1007/s10115-010-0325-9
16. Fu Y, Creado M, Ju C (2001) Reorganizing web sites based on user access patterns. In: Proceedings of the tenth international conference on information and knowledge management, pp 583–585
17. Gorman J, Curran JR (2006) Random Indexing using statistical weight functions. In: Proceedings of the conference on empirical methods in natural language processing (EMNLP 2006), Sydney, Australia, pp 457–464
18. Halkidi M, Vazirgiannis M, Batistakis I (2000) Quality scheme assessment in the clustering process. In: Proceedings of the 4th European conference on principles and Practice of Knowledge Discovery in Databases (PKDD 2000), Lyon, France
19. Hou J, Zhang Y (2002) Constructing good quality web page communities. In: Proceedings of the 13th Australasian database conferences (ADC2002), vol 36. ACS Inc, Melbourne, pp 65–74
20. Hou J, Zhang Y (2003) Effectively finding relevant web pages from linkage information. IEEE Trans Knowl Data Eng 15(4):940–951
21. IBM (2003) SurfAid analytics. http://surfaid.dfw.ibm.com
22. Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice Hall
23. Jin X, Zhou Y, Mobasher B (2004) A unified approach to personalization based on probabilistic latent semantic models of web usage and content. In: Proceedings of the AAAI 2004 workshop on semantic web personalization (SWP'04), San Jose
24. Kanerva P, Kristofersson J, Holst A (2000) Random Indexing of text samples for latent semantic analysis. In: Proceedings of the 22nd annual conference of the cognitive science society. Erlbaum, New Jersey, p 1036
25. Kanerva P (1988) Sparse distributed memory. The MIT Press, Cambridge
26. Kanerva P, Sjödin G, Kristofersson J, Karlsson R, Levin B, Holst A, Karlgren J, Sahlgren M (2001) Computing with large random patterns. In: Uesaka Y, Kanerva P, Asoh H (eds) Foundations of real-world intelligence. CSLI Publications, Stanford
27. Kaski S (1999) Dimensionality reduction by random mapping: fast similarity computation for clustering. In Proceedings of the international joint conference on neural networks (IJCNN98), IEEE Service Center
28. Krishnapuram R, Joshi A, Nasraoui O, YI L (2003) Low-complexity fuzzy relational clustering algorithms for web mining. IEEE Trans Fuzzy Syst 4(9):596–607
29. Lan B, Bressan S, Ooi BC, Tan K (2000) Rule-assisted prefetching in web server caching. In: Proceedings of 2000 ACM international conference on information and knowledge management (Virginia, USA), vol 1. ACM, New York, pp 504–11
30. Landauer T, Dumais S (1997) A solution to Platos problem: the latent semantic analysis theory for acquisition, induction and representation of knowledge. Psychol Rev 104(2):211–240
31. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, pp 281–297
32. Mobasher B, Cooley R, Srivastava J (2000) Automatic personalization based on web usage mining. Commun ACM 8(43):142–151
33. Nasraoui O, Frugui H, Krishnapuram R, Joshi A (2000) Extracting web user profiles using relational competitive fuzzy clustering. Int J Artif Intell Tools 4(9):509–526
34. Nanopoulos A, Katsaros D, Manolopoulos Y (2001) Effective prediction of web-user accesses: a data mining approach. In Proceedings of workshop web usage analysis and user profiling (WebKDD'01) (San Francisco, USA). ACM, New York
35. Oceans Research Group. Department of Computer Science, Boston University. http://cs-www.bu.edu/groups/oceans/Home.html
36. Paliouras G, Papatheodorou C, Karkaletsis V, Spyropoulos CD (2000) Clustering the users of large web sites into communities. In: Proceedings of the international conference on machine learning (ICML), pp 719–726
37. Pal SK, Ghosh A, Uma Shankar B (2000) Segmentation of remotely sensed images with fuzzy thresholding and quantitative evaluation. Int J Remote Sens 21(11):2269–2300
38. Sahlgren M, Karlgren J (2005) Automatic bilingual lexicon acquisition using Random Indexing of parallel corpora. J Nat Lang Eng (Special Issue on Parallel Texts)
39. Sahlgren M, Karlgren J (2005) Automatic bilingual lexicon acquisition using Random Indexing of parallel corpora. J Nat Lang Eng Special Issue Parallel Texts 11(3):1–14
40. Theodoridis S, Koutroumbas K (2006) Pattern recognition, 3rd edn. Academic Press, New York

41. Tan P-N, Steinbach M, Kumar V (2006) Introduction to data mining. Pearson Addison-Wesley, Reading
42. The Internet Traffic Archive. http://ita.ee.lbl.gov/index.html
43. Teng W, Chang C, Chen M (2005) Integrating web caching and web prefetching in client-side proxies. IEEE Trans Parallel Distrib Syst 16:444–455
44. Tian W, Choi B, Phoha VV (2002) An adaptive web cache access predictor using neural network. In: Proceedings of 15th international conference on IEA/AIE (Cairns, Australia), vol 2358. Springer, Berlin, pp 450–459
45. Wan M, Li L, Xiao J, Yang Y, Wang C, Guo X (2010) CAS based clustering algorithm for Web users. Nonlinear Dyn 61(3):347–361
46. Wu Y, Chen A (2002) Prediction of web page accesses by proxy server log. World Wide Web 5:67–88
47. Xie Y, Phoha VV (2001) Web user clustering from access log using belief function. In: Proceedings of the 1st international conference on Knowledge capture, pp 202–208
48. Yang S, Li Y, Wu X, Pan R (2006) Optimization study on k value of K-means algorithm. J Syst Simul 18(3):97–101
49. Zhou Y, Jin X, Mobasher B (2004) A recommendation model based on latent principal factors in web navigation data. In: Proceedings of the 3rd international workshop on web dynamics. ACM Press, New York

Author Biographies

Miao Wan graduated in automation from Beijing University of Posts and Telecommunications, Beijing, China, in 2006. In 2008, she joined the Information Security Center at Beijing University of Posts and Telecommunications as a Ph.D. candidate and received the Ph.D. degree in signal and information processing in July of 2011. Since 2006, her research interests have been data mining, pattern recognition and intelligent information processing.

Arne Jönsson is professor at the Computer and Information Science Department, Linköping University and senior researcher at Santa Anna IT Research Institute AB. He received the Ph.D. degree in computer science from Linköping University in 1993. He is the head of the Human Centered Systems Division. His research activities include multimodal human computer interaction, especially in the area of natural language dialogue systems, and the use of vector space models for language technology.

Cong Wang graduated in Computer Science from Hunan University at Changsha, China, in 1982. From 1982 to 2002, she was a researcher-level senior engineer at China Academy of Machinery Science and Technology (CAM). She received her Ph.D. degree in 2002. Since 2002, she has been a Professor in Beijing University of Posts and Telecommunications. At present, she is deputy director of National Engineering Laboratory for Disaster Backup and Recovery, and deputy director of Key Laboratory of Reliable distributed computing and service of MOE, China. Her research activities are focused on information security, intelligent information processing, disaster recovery technology standards and Emergency communication standards.

Lixiang Li received the M.S. degree in circuit and system in 2003 from Yanshan university, qinhuangdao, China, and a Ph.D. degree in signal and information processing in 2006 from Beijing University of Posts and Telecommunications, Beijing, China. She is currently an associate professor at the School of Computer Science and Technology, Beijing University of Posts and Telecommunications, China. Her research interests are swarm intelligence, reconfigurable dynamic computing and information security.

Yixian Yang is the Managing Director of information security center, Beijing University of Posts and Telecommunications, Beijing, China. He received the M.S. degree in applied mathematics in 1986 and the Ph.D. degree in electronics and communication systems in 1988 from Beijing University of Posts and Telecommunications, Beijing, China. His research interests are network security, information security and coding theory.