Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining

Raghu Krishnapuram
IBM India Research Lab
Indian Institute of Technology, Hauz Khas, New Delhi 110016
kraghura@in.ibm.edu
On leave from Dept of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401
Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
joshi@cs.umbc.edu
Olfa Nasraoui
Department of Electrical Engineering
University of Memphis, Memphis, TN 38152
Liyu Yi
BoldTech Systems Inc.
Denver, CO 80202
lyi@mines.edu

Abstract

This paper presents new algorithms (Fuzzy c-Medoids or FCMdd and Robust Fuzzy c-Medoids or RFCMdd) for fuzzy clustering of relational data. The objective functions are based on selecting c representative objects (medoids) from the data set in such a way that the total fuzzy dissimilarity within each cluster is minimized. A comparison of FCMdd with the well-known Relational Fuzzy c-Means algorithm (RFCM) shows that FCMdd is more efficient. We present several applications of these algorithms to *Web mining*, including Web document clustering, snippet clustering, and Web access log analysis.

Keywords

Web mining, user access patterns, user profiles, fuzzy relational clustering, document clustering, snippet clustering.

I. Introduction

The evolution of the Internet into the Global Information Infrastructure has lead to an explosion in the amount of available information. The Web, then, is becoming the apocryphal *vox populi*. Realizing the vision of distributed knowledge access in this scenario and its future evolution will need tools to "personalize" the information space. If one were to look at the Web as a distributed, heterogeneous information base, Web personalization amounts to creating a system which responds to user queries based on information about him/her. As a trivial example, a biologist querying on cricket in all

likelihood wants something other than a sports enthusiast would. The explosion of the Web has brought the personalization problem front and center, not just because of the amount of time wasted in searching for information, but because of the massive traffic surge this has generated in the Internet backbone.

Personalization can either be done via information brokers (e.g. Web search engines), or in an *end-to-end* manner by making Web sites adaptive. The latter solution is further attractive since it also cuts down on network traffic. Initial work in this area has basically focused on creating broker entities, often called recommender systems. One of the earliest such systems was the Firefly system [47] which attempted to provide CDs that best match a user's professed interests based on "collaborative filtering". In collaborative filtering, each user is modeled by the items that he or she expresses interest in. However, the "characteristics" of the items are not taken into account. This is unlike the so-called "content-based" approaches (see articles in [13]). The advantage of this approach is that learning is unsupervised, i.e., no labeled data is required. More recently, systems such as PHOAKS [50] and our own W^3IQ[25], [24] have sought to use cooperative information retrieval techniques for personalization.

End–to–End personalization is predicated on adaptive Web sites[39], [38], which change the information returned in response to a user request based on the user. Very primitive forms of this can be seen in sites that ask the users to provide some basic information such as address, phone number, and keywords indicating interest, and then tailor their information content (and especially ads) based on zip code, area code, demographic profile, etc. However, in general the appearance of a particular page, including links on it, can also be changed when Web sites are adaptive. For example, Etzoni*et al.*[39] define operations such as promotions/demotions, highlighting and linking that could be done on static pages to create content tailored for a specific user dynamically. While interesting, their formalism seems to demand a robust clustering technique in order to work successfully. Moreover, factors they propose such as correlations between two pages are inherently fuzzy and best tackled using a fuzzy approach. Perhaps the earliest work along similar lines was the Webwatcher project at CMU[2], [22], which uses a type of reinforcement learning. It highlights hyperlinks in a page based on the declared interests and the path traversal pattern of a user as well as the path traversal patterns of previous users with similar interests. However, the underlying learning algorithms are not robust in terms of being able to handle bad data, which can pose a significant problem. As the authors themselves

observe, several users end up traversing paths that do not correspond to their stated interests, which can generate a significant number of bad data points and outliers. Also, the users are presented with a binary choice in terms of expressing their interest using keywords, and there is no notion of degree of interest.

In other more active schemes that use supervised learning[37], the users are asked to explicitly rate pages, and these ratings (encryption) are stored or their viewing time is monitored [37]. Supervised learning, should be avoided if possible, since most users are reluctant to label data and provide explicit feedback. A somewhat similar approach is taken in the Avanti project [14]. It uses an initial interview to gather user interests, as well as possibly classify them into known stereotype users. Another approach to observing path traversal and clustering based on that data is advanced by Shahabi et al. [7]. The basic approach there is to define a path similarity measure for a given Web site. Then, the logged data about user's paths is clustered using a simple K-means algorithm to aggregate users into groups. However, it is not clear how the similarity metric is devised, and whether it can produce meaningful clusters. There is also a recent body of work [3], [36] which seeks to transform the Web into a more structured, database-like entity. In particular, Han et al. [36] create a MOLAP-based warehouse from Web logs, and allow users to perform analytic queries. They also seek to discover time dependent patterns in the access logs [53]. Similar approaches, essentially based on the association rule ideas [1], have been proposed in [9], [10]. However, both these approaches assume that logs contain user ids, which is not true in the real world except in the rare case that the *ident* protocol is used and the clients are willing to release the user names. A related topic is that has been recently gaining momentum is the idea that we can learn much about users and customers by tracking and analyzing their *clickstreams*, which is of great importance in e-commerce.

An important component of personalization is Web Mining. Web Mining can be viewed as the extraction of structure from unlabeled, semi-structured data containing the characteristics of users or information. The logs kept by Web servers provide a classic example of such data. Web mining can be viewed as a special case of the more general problem of knowledge discovery in databases [1]. It can be said to have three operations of particular interest: clustering (e.g. finding natural groupings of users, pages etc.), associations (e.g. which URLs tend to be requested together), and sequential analysis (the order in which URLs tend to be accessed). The first two form the focus of our ongoing

work. However, Web mining in general, and site analysis in particular, may well turn out to be a flop unless proper attention is paid to formalizing the goals, analysis, and evaluation methods. In this paper, we describe some of the challenges of Web mining, and point out where a fuzzy approach might be useful. We present new low-complexity fuzzy relational clustering algorithms and their application to the analysis of Web access logs for extration of user profiles. These profiles can then be used to make a site adaptive.

The rest of the paper is organized as follows. In Section II we point out the role of fuzzy granularity in Web minging. In Section III, we present an overview of relational clustering algorithms. In Section IV we introduce the Fuzzy C Medoids (FCMdd) algorithm as well as a low-complexity version of it. In Section V, we describe a robust version of FCMdd that can handle outliers in the data. In Section VI , we discuss how Web objects can be represented and dissimilarities between them can be measured. In Section VIII , we present experimental results that involve the application of FCMdd and its variations for clustering Web documents and snippets, and for mining access logs. Finally, in Section IX we present the conclusions.

II. The Role of Fuzzy Granularity in Web Mining

The process of identifying structure in an unlabeled data set in terms of categories or components plays a central role in Web mining. Consider for example the problem of determining categories of users with "similar" interests, or the problem of grouping together a set of pages with similar content, and so on. Like Data mining, Web mining needs to deal with problems of scale (extremely large data sets). However, there are several new challenges that are raised by Web mining that make the straightforward use of data mining techniques not particularly useful. For one, the categories and associations in Web mining do not have crisp boundaries. They overlap considerably and are best described by fuzzy sets. In addition, bad exemplars (outliers) and incomplete data can easily occur in the data set, due to a wide variety of reasons inherent to Web browsing and logging. Thus, Web Mining requires modeling of an unknown number of overlapping sets in the presence of significant noise and outliers. For example, users' access patterns are not entirely fixed. While a user may mostly visit the CNN site to get financial information, she may also go there for sports- or politics-related news as well. So while there are trends there to be discovered regarding this users behaviour in CNN Web

server's logs, they are buried in data with significant noise components. Therefore, robust methods, that can deal with significant amounts of noise, are needed.

A major problem in personalization is the lack of data. To achieve effective personalization, we need to combine demographic data, psychographic data, as well as user access patterns and interests. This leads to a feature space of an extremely high dimensionality. However, this type of data is hard to collect for a large number of customers. In other words, in practice the number of data points we can collect can be small in relation to the number of features. Predictive models (including statistical tools) cannot produce reliable results unless the data represents a significant percentage of the total number of possible (traversal) patterns, which is astronomically large. Thus, methods for reducing the dimensionality of the problem are required. Also, Web objects (pages, URLs, etc.) are non numeric in nature, making "distance" measurements and "equality" judgements between them, a prerequisite to grouping of any kind, an issue in itself. Hence, algorithms that can also deal with non-numeric data are desirable. Data mining techniques have been typically developed for structured domains where these issues are not significant. Thefore, blindly dumping Web object data into a data mining tool is not expected to yield good results. Our approach to this problem is to simplify it by categorizing the user space as well as the document space by applying low-complexity robust relational fuzzy clustering techniques to generate reliable "fuzzy granules". By manipulating the granules rather than the original high-dimensional data, we can increase the reliability of the results.

To summarize, in order for a categorization technique to be useful in an application such as Web mining, it needs to satisfy five basic requirements: (i) The technique should be able to handle overlapping components. (ii) It needs to be robust. (iii) It needs to be able to handle relational data, since in many instances Web objects cannot be represented by numerical vectors. (iv) It needs to be scalable to extremely large high-dimensional data sets. (v) Finally, the technique should be able to determine the appropriate number of components automatically, since *a priori* knowledge about the number of components is rarely available. By robustness, we mean that the categorization process (and hence the performance of a system) should not be affected drastically due to outliers (bad observations), provided there is enough "good" data to support the assumed model. Robustness issues have been discussed in detail in [11]. Scalability is an issue that has often been ignored, but that is quite critical to these applications. For example, the Web server at CSEE department of UMBC experiences thirty

to forty thousand hits per day. Popular sites such as news (CNN, New York Times) or search engines get an order or two of magnitude greater number of hits. While the algorithm we propose helps with scalability (since it is of linear complexity), we feel that greater improvements are possible.

III. Relational Clustering

The term "relational data" refers to the situation where we have only numerical values representing the degrees to which pairs of objects in the data set are related. In contrast, "object data" refers to the the situation where the objects to be clustered are represented by vectors $\mathbf{x}_i \in \Re^p$. Algorithms that generate partitions of relational data are usually referred to as relational (or sometimes pair-wise) clustering algorithms. Relational clustering is more general in the sense that it is applicable to situations in which the objects to be clustered cannot be represented by numerical features. For example, we can use relational clustering algorithms to cluster URLs (Universal Resource Locators) if we can define a dissimilarity measure to quantify the degree of resemblance between pairs of URLs. The pair-wise dissimilarities are usually stored in the form of a matrix called the dissimilarity matrix.

There are several well-known relational clustering algorithms in the literature. One of the most popular is the SAHN (Sequential Agglomerative Hierarchical Non-overlapping) model [48] which is a bottom-up approach that generates crisp clusters by sequentially merging pairs of clusters that are closest to each other in each step. Depending on how "closeness" between clusters is defined, the SAHN model gives rise to single, complete and average linkage algorithms. A variation of this algorithm can be found in [17]. Another well-known relational clustering algorithm is PAM (Partitioning Around Medoids) due to Kaufman and Rousseeuw [27]. This algorithm is based on finding k representative objects (also known as *medoids* [26]) from the data set in such a way that the sum of the within cluster dissimilarities is minimized. A modified version of PAM called CLARA (Clustering LARge Applications) to handle large data sets was also proposed by Kaufman and Rousseeuw [27]. Ng and Han [35] propose another variation of CLARA called CLARANS. This algorithm tries to make the search for the k representative objects (medoids) more efficient by considering candidate sets of k medoids in the neighborhood of the current set of k medoids. However, CLARANS is not designed for relational data. Finally, it is also interesting to note that Fu [15] suggested a technique very similar to the k medoid technique in the context of clustering string patterns generated by grammars in syntactic

pattern recognition. Some of the more recent algorithms for relational clustering include [16], [40], [49], and [4].

SAHN, PAM, CLARA and CLARANS generate crisp clusters. When the clusters are not well defined (i.e., when they overlap) we may desire fuzzy clusters. Two of the early fuzzy relational clustering algorithms are the ones due to Ruspini [44] and Diday [12]. Other notable algorithms include Roubens' Fuzzy Non Metric Model or FNM [41], Windham's Association Prototype Model or AP [52], Hathaway & Bezdek's Relational Fuzzy c-Means or RFCM [20], and Kaufman & Rousseeuw's Fuzzy Analysis or FANNY [27]. FANNY is in fact very closely related to RFCM, and is essentially equivalent to RFCM when the fuzzifier, m, is equal to 2. In our experience, RFCM and FANNY are the most reliable. Some improvements on this algorithm can also be found in the literature. For example, the NERFCM model [19] extends RFCM to ease the restrictions that RFCM imposes on the dissimilarity matrix. More recently, Sen and Davé [46] have generalized this approach further, including an extension to handle data sets containing noise and outliers.

One problem with RFCM is that its computational complexity is $\mathcal{O}(n^2)$. Thus, it is unsuitable for Web mining applications where n is extremely large. In this paper, we present an objective function for fuzzy relational clustering based on the idea of identifying k medoids, and propose a heuristic algorithm to minimize it. We call this algorithm Fuzzy c-Medoids and abbreviate it as FCMdd rather than FCM. (FCM is usually associated with the Fuzzy C Means algorithm in the fuzzy clustering community). We show that even though the worst case complexity of FCMdd is $\mathcal{O}(n^2)$, it can be made linear in practice. We also propose a robust version of this alrorithm which has a complexity of $\mathcal{O}(n\log n)$. In the next section, we define the objective functions for FCMdd and present its low-complexity version. In Section V, we present objective functions for robust versions of FCMdd.

IV. The Fuzzy c Medoids Algorithm (FCMdd)

Let $X = \{\mathbf{x}_i | i = 1, 2, \ldots, n\}$ be a set of n objects. Each object may or may not be represented by a feature vector. Let $r(\mathbf{x}_i, \mathbf{x}_j)$ denote the dissimilarity between object $\mathbf{x}_i$ and object $\mathbf{x}_j$. Let $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}, \mathbf{v}_i \in X$ represent a subset of X with cardinality c, i.e., $\mathbf{V}$ is a c-subset of X. Let

X^c represent the set of all c-subsets $\mathbf{V}$ of X. The Fuzzy Medoids Algorithm (FCMdd) minimizes:

$$J_m(\mathbf{V}; X) = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^m \; r(\mathbf{x}_j, \mathbf{v}_i), \tag{1}$$

where the minimization is performed over all $\mathbf{V}$ in X^c. In (1), u_{ij} represents the fuzzy [6], or possibilistic [29], [18] membership of $\mathbf{x}_j$ in cluster i. The membership u_{ij} can be defined heuristically in many different ways. For example, we can use the FCM [6] membership model given by:

$$u_{ij} = \frac{\left(\frac{1}{r(\mathbf{x}_j, \mathbf{v}_i)}\right)^{1/(m-1)}}{\sum_{k=1}^{c} \left(\frac{1}{r(\mathbf{x}_j, \mathbf{v}_k)}\right)^{1/(m-1)}}, \tag{2}$$

where $m \in [1, \infty)$ is the "fuzzifier". Another possibility is [5]:

$$u_{ij}^m = \frac{\exp\{-\beta r(\mathbf{x}_j, \mathbf{v}_i)\}}{\sum_{k=1}^{c} \exp\{-\beta r(\mathbf{x}_j, \mathbf{v}_k)\}}. \tag{3}$$

Above equations generate a fuzzy partition of X in the sense that the sum of the memberships of an object $\mathbf{x}_j$ across classes is equal to 1. If we desire possibilistic memberships [29], we could use functions of the following type [30]:

$$u_{ij} = \left[1 + \frac{r(\mathbf{x}_j, \mathbf{v}_k)}{\eta_i}\right]^{-1} \tag{4}$$

or

$$u_{ij} = \exp\left(-\frac{r(\mathbf{x}_j, \mathbf{v}_i)}{\eta_i}\right). \tag{5}$$

The parameters β in (3) and η (4) are "scale" parameters that reflect the size of the clusters. In (4) the scale is different for each cluster. This parameter needs to be specified or estimated from the data. This parameter plays an important role in determining cluster boundaries and ignoring outliers [11].

Since u_{ij} is a function of the dissimilarities $r(\mathbf{x}_j, \mathbf{v}_k)$, it can be eliminated from (1). This is the reason J_m is shown as a function of $\mathbf{V}$ alone. When (1) is minimized, the $\mathbf{V}$ corresponding to the solution generates a fuzzy or possibilistic partition via an equation such as (2). However, (1) cannot be minimized via the alternating optimization technique, because the necessary conditions cannot be derived by differentiating it with respect to the medoids. (Note that the solution space is discrete.) Thus, strictly speaking, an exhaustive search over X^c needs to be used. However, following Fu's [15] heuristic algorithm for a crisp version of (1), we describe the following fuzzy algorithm (FCMdd) that minimizes (1). In this version, we assume that the memberships are given by (2).

The Fuzzy c-Medoids Algorithm (FCMdd)

Fix the number of clusters c; Set $iter = 0$;
Pick initial medoids $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$ from X^c;
Repeat
  Compute memberships u_{ij} for $i = 1, 2, \ldots, c$ and
    $j = 1, 2, \ldots, n$, by using (2); (A)
  Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
  Compute the new medoids $\mathbf{v}_i$ for $i = 1, 2, \ldots, c$:
    $q = \underset{1 \le k \le n}{\operatorname{argmin}} \sum_{j=1}^{n} u_{ij}^m \, r(\mathbf{x}_k, \mathbf{x}_j)$; $\mathbf{v}_i = \mathbf{x}_q$; (B)
  $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V}$ or $iter = MAX\_ITER\right)$.

The crisp version of FCMdd above, which we call the Hard c Medoids (HcMdd) algorithm, can be obtained by replacing step (A) with:

$$q = \underset{1 \le k \le c}{\operatorname{argmin}} \, r(\mathbf{x}_j, \mathbf{v}_k); \quad u_{ij} = \begin{cases} 1 & \text{if } i = q \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The above algorithm falls in the category of Alternating Cluster Estimation [43] paradigm, and is not *guaranteed* to find the global minimum. It is advisable to try many random initializations to increase the reliability of the results. We have experimented with three different ways of initializing the medoids. The first way is to pick all the medoid candidates randomly. We call this method Initialization I. The second way is to pick the first candidate as the object that is most central to the data set, and then pick each successive one by one in such a way that each one is most dissimilar to all the medoids that have already been picked. This makes the initial medoids evenly distributed. We refer to this procedure as Initialization II.

Initialization II for FCMdd

Fix the number of medoids $c > 1$;
Compute the first medoid:
  $q = \underset{1 \le j \le n}{\operatorname{argmin}} \sum_{i=1}^{n} r(\mathbf{x}_j, \mathbf{x}_i)$; $\mathbf{v}_1 = \mathbf{x}_q$;
Set $V = \{\mathbf{v}_1\}$, $iter = 1$;
Repeat
  $iter = iter + 1$;
  $q = \underset{1 \le i \le n; \mathbf{x}_i \notin V}{\operatorname{argmax}} \; \underset{1 \le k \le |V|}{\min} \, r(\mathbf{v}_k, \mathbf{x}_i)$; $\mathbf{v}_{iter} = \mathbf{x}_q$;
  $V = V \cup \{\mathbf{v}_{iter}\}$;
Until $(iter = c)$.

For a given data set, the initialization produced by Initialization II is always fixed. Sometimes a bit of randomness might be desirable. In the third initialization strategy, we add randomness by picking the first medoid candidate randomly. The rest of the medoids are selected the same way as in Initialization II. We call this method Initialization III. The computational complexity of Initialization I, II and III are $\mathcal{O}(c)$, $\mathcal{O}(nc^2)$ and $\mathcal{O}(nc^2)$ respectively. We found that both Initialization II and III work well in practice.

The fuzzifier m in FCMdd determines the degree of fuzziness of the resulting clusters. Since the medoid always has a membership of 1 in the cluster, raising its membership to the power m has no effect. Thus, when m is high, the mobility of the medoids from iteration to iteration may be lost, because all memberships become very small except the one corresponding to the current medoid. For this reason, we recommend a value between 1 and 1.5 for m.

It can be seen from step (B) of FCMdd that the complexity of the algorithm is $\mathcal{O}(n^2)$, where n is the number of input objects. However, this is too expensive for most Web mining applications. To overcome this problem, we can modify step (B) of FCMdd so that it examines only a subset of objects while updating the medoid for cluster i. The subset we choose is the set of p objects in X that correspond to the top p highest membership values in cluster i. We denote this subset by $X_{(p)i}$. The subsets $X_{(p)i}$, $i = 1, 2, \ldots, c$, can be identified during the membership updating step, i.e., in step (A) of the algorithm. This increases the complexity of step (A) to $\mathcal{O}(ncp)$. However, the complexity of step (B) is reduced to $\mathcal{O}(ncp)$. Therefore, the overall complexity is linear in the number of objects. The value of p should be proportional to the dimensionality d of the data, e.g. $2d$. However, when the data is relational, d has no meaning. In such cases, p could be chosen to be much smaller than the average number (n/c) of points in a cluster. The modified FCMdd algorithm is summarized below.

Linearized Fuzzy c-Medoids Algorithm (LFCMdd)

Fix the number of clusters c; Set iter = 0;
Pick the initial set of medoids $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$ from X^c;
Repeat
  Compute memberships u_{ij} for $i = 1, 2, \ldots, c$, and $j = 1, 2, \ldots, n$,
    by using (2) and identify $X_{(p)i}$, $i = 1, 2, \ldots, c$. (A)
  Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
  Compute the new medoids $\mathbf{v}_i$ for $i = 1, 2, \ldots, c$:
    $q = \underset{\mathbf{x}_k \in X_{(p)i}}{\operatorname{argmin}} \sum_{j=1}^{n} u_{ij}^m \; r(\mathbf{x}_k, \mathbf{x}_j)$ $\mathbf{v}_i = \mathbf{x}_q$; (B)
  $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V}$ or $iter = MAX\_ITER\right)$.

V. Robust Versions of FCMdd

It is well-known that algorithms that minimize a Least-Squares type objective function are not robust [42], [11]. In other words, a single outlier object could lead to a very unintuitive clustering result. To overcome this problem, we design an objective function for a robust version of FCMdd based on the Least Trimmed Squares idea [42], [28], we use the membership function in (2). Substituting the expression for u_{ij} in (2) into (1), we obtain:

$$J_m(\mathbf{V}; \mathbf{X}) = \sum_{j=1}^{n} \left(\sum_{i=1}^{c} (r(\mathbf{x}_j, \mathbf{v}_i))^{1/(1-m)} \right)^{1-m} = \sum_{j=1}^{n} h_j, \tag{7}$$

where

$$h_j = \left(\sum_{i=1}^{c} (r(\mathbf{x}_j, \mathbf{v}_i))^{1/(1-m)} \right)^{1-m} \tag{8}$$

is $1/c$ times the harmonic mean of the dissimilarities $\{r(\mathbf{x}_j, \mathbf{v}_i)) : i = 1, 2, \ldots, c\}$ when $m = 2$. The objective function for the Robust Fuzzy c Medoids (RFCMdd) algorithm is obtained by modifying (7) as follows:

$$J_m^T(\mathbf{V}; \mathbf{X}) = \sum_{k=1}^{s} h_{k:n}. \tag{9}$$

In (9), $h_{k:n}$ represents the k-th item when $h_j, j = 1, 2, \ldots, n$, are arranged in ascending order, and $s < n$. The value of s is chosen depending on how many objects we would like to disregard in the clustering process. This allows the clustering algorithm to ignore outlier objects while minimizing the objective function. For example, when $s = n/2$, 50% of the objects are not considered in the clustering process, and the objective function is minimized when we pick c medoids in such a way that the sum

of the harmonic-mean dissimilarities of 50% of the objects is as small as possible. We can design the following heuristic algorithm to minimize (9).

The Robust Fuzzy c Medoids Algorithm (RFCMdd)

Fix the number of clusters c, and the fuzzifier m;
Pick the initial medoids $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$ from X^c;
$iter = 0$;
Repeat
  Compute harmonic dissimilarities h_j for $j = 1, 2, \ldots, n$, using (8);
  Sort h_j, $j = 1, 2, \ldots, n$ to create $h_{j:n}$;
  Keep the s objects $\mathbf{x}_{1:n}, \ldots, \mathbf{x}_{s:n}$, corresponding to the first s $h_{j:n}$;
  Compute memberships $u_{ij:n}$ for $i = 1, 2, \ldots, c$ and $j : n = 1, 2, \ldots, s$,
    by using (2), and identify $X_{(p)i}$, $i = 1, 2, \ldots, c$;
  Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
  Compute the new medoids $\mathbf{v}_i$ for $i = 1, 2, \ldots, c$:
    $q = \underset{\mathbf{x}_{k:n} \in X_{(p)i}}{\operatorname{argmin}} \sum_{j=1}^{s} u_{ij:n}^m \, r(\mathbf{x}_{k:n}, \mathbf{x}_{j:n})$; $\mathbf{v}_i = \mathbf{x}_q$;
  $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V}$ or $iter = MAX\_ITER\right)$.

As mentioned above, the choice of the retention ratio, s/n, should reflect the percentage of noise in the data. If the noise proportion is higher than 50%, we cannot guarantee that we will obtain the correct estimates for the parameters such as cluster center, variance, etc [11], [42]. A common approach in robust statistics is to assume that the noise proportion is 50% and then apply a correction to the estimate *after* the parameters have been (robustly) estimated [42]. For a known distribution (such as a Gaussian), the correction is given by a simple formula. In our application, since the data is relational, such corrections can be difficult to apply, except in a heuristic manner. Another option is to estimate the retention ratio when it is not known in advance. When estimation is not possible, our recommendation is that we estimate or pick a lowerbound for retention ratio and use it. This will at least ensure that the estimates are not affected by outliers. It is also possible to optimize the objective function with respect to the retension ratio. See for example [33], [34]. We have made appropriate changes to the manuscript.

Interestingly, the worst-case complexity of RFCMdd algorithm still remains $\mathcal{O}(n\log n)$. This is a good result, considering that robust algorithms are very expensive. It is quite trivial to design a robust version based on the Least Median of Squares idea as well [32]. In this case we simply replace

the summation in (9) by the median. In other words, we use

$$J_m^M(\mathbf{V};\mathbf{X}) = \underset{1\le k\le n}{\text{median}}\ h_{k:n}. \tag{10}$$

VI. Representing Web Objects and Measuring Dissimilarity

A. Web Documents and Snippets

We represent Web documents and snippets by feature vectors. The procedure we used to generate the feature vectors is as follows. We first use a "stop-word elimination and stemming" algorithm obtained from Louisiana State University [8] to filter out insignificant words and remove certain types of word-endings, e.g., "ing" and "ed". We then select 500 keywords out of the collection by using the Inverted Document Frequency (IDF) method [23], [45]. The IDF method takes into account both the frequency of a given keyword and the information about its distribution in the whole document collection. The IDF value for a given keyword is computed as follows:

$$f_{IDF} = \frac{f_w}{f_{w\_max}} \log\left(\frac{P}{P_w}\right). \tag{11}$$

In (11) f_w is the frequency of occurrence of the keyword in the document collection, f_{w_max} is the maximum frequency of occurrence of any keyword in the document collection, P_w is the number of documents that include this keyword and P is the number of documents in the whole collection. The IDF value of a keyword measures its "importance". We sort the keywords by their IDF values and select the 500 keywords with the largest IDF values. These keywords are denoted by $K_1, K_2, \ldots, K_{500}$. We then represent the i-th document by a feature vector $\mathbf{k}_i = [k_{i1}, \ldots, k_{id}]$, where k_{ik} is the normalized frequency of the the k-th keyword K_k in the i-th document. The normalization is done with respect to the total number of keywords in this document. Thus, we generate a 500-dimensional feature vector for each document. To reduce the dimensionality, we apply principal component analysis (PCA) [51] and select the eigenvectors corresponding to the top 10 eigenvalues as the new features. We project each 500-dimensional vector $\mathbf{k}_i$ representing a document onto the 10 eigenvectors to form a 10-dimensional feature vector $\mathbf{k}_i'$. We perform the clustering on the dissimilarity matrix generated from these 10-dimensional vectors. However, in the case of snippets, since the text is very short, the dimensionality reduction step did not give good results. Therefore we used the original d-dimensional vecotors directly in computing dissimilarities.

In addition to the traditional Minkowski norms, we can also compute the dissimilarity between two snippets $\mathbf{k}_1$ and $\mathbf{k}_2$ using the Jaccard Index [21] dissimilarity measure given by:

$$d_J(\mathbf{k}_1, \mathbf{k}_2) = \frac{\sum_{i=1}^{d} min(k_{1i}, k_{2i})}{\sum_{i=1}^{d} max(k_{1i}, k_{2i})}, \tag{12}$$

or the more traditional cosine measure given by:

$$d_c(\mathbf{k}_1, \mathbf{k}_2) = \frac{\mathbf{k}_1^T \mathbf{k}_2}{\| \mathbf{k}_1 \| \| \mathbf{k}_2 \|}. \tag{13}$$

B. Web Sessions

In this section, we define the notion of a "user session" as being a temporally compact sequence of Web accesses by a user. We also define a new dissimilarity measure between two Web sessions that captures the organization of a Web site. The goal of this particular Web mining application is to categorize these sessions so that we can extract typical session profiles.

Each access log entry consists of: (i) user's IP address, (ii) access time, (iii) request method ("GET", "POST", $\cdots$, etc), (iv) URL of the page accessed, (v) data transmission protocol (typically HTTP/1.0), (vi) return code, and (vii) number of bytes transmitted. First, we filter out log entries that are not germane for our task. These include entries that: (i) result in any error (indicated by the error code), (ii) use a request method other than "GET", or (iii) record accesses to image files (.gif, .jpeg, , $\cdots$, etc.), which are embedded in other pages. Next, analogous to [10], the individual log entries are grouped into user sessions. Since Web servers do not typically log user names (unless *ident* is used), we define a user session as accesses from the same IP address such that the duration of elapsed time between two consecutive accesses in the session is within a prespecified threshold. In our experiments, we used 45 min as the threshold. Each URL in the site is assigned a unique number $j \in \{1, 2, \ldots, N_U\}$, where N_U is the total number of valid URLs. The i^{th} user session is encoded as an N_U-dimensional binary vector $\mathbf{s}^{(i)}$ with the property

$$s_j^{(i)} = \begin{cases} 1 & \text{if user accessed } j^{th} \text{ URL during } i^{th} \text{ session} \\ 0 & \text{otherwise} \end{cases}$$

Our scheme will map one user's multiple sessions to multiple user sessions. This notion of multiple user sessions enable us to better capture the situation when the same user displays a few (different) access patterns on this site.

C. Defining Dissimilarity between Two Sessions

A simple measure of similarity between sessions $\mathbf{s}^{(k)}$ and $\mathbf{s}^{(l)}$ is given by:

$$S_{1,kl} = \frac{\sum_{i=1}^{Nu} s_i^{(k)} s_i^{(l)}}{\sqrt{\sum_{i=1}^{Nu} s_i^{(k)}} \sqrt{\sum_{i=1}^{Nu} s_i^{(l)}}} \tag{14}$$

Note that the above measure is essentially the cosine measure, since the s_i's are binary. The problem with this similarity measure is that it completely ignores the hierarchical organization of the Web site. For example, the session pair {/courses/cecs345} and {/courses/cecs343}, as well as the session pair {/courses/cecs345} and {/research/grants} will receive a zero similarity score according to S_1. This leads us to define an alternative similarity measure that takes into account the syntactic representation of two URLs as follows.

$$S_u(i,j) = \min\left(1, \frac{|(p_i \cap p_j|}{\max(1, \max(|p_i|, |p_j|) - 1)}\right), \tag{15}$$

where p_i denotes the path traversed from the root node to the node corresponding to the i^{th} URL, and $|p_i|$ indicates the length of this path or the number of edges included in the path. Now the similarity on the session level, which incorporates the syntactic URL similarities, is defined by correlating all the URL attributes and their similarities in two sessions as follows:

$$S_{2,kl} = \frac{\sum_{i=1}^{N_U} \sum_{j=1}^{N_U} s_i^{(k)} s_j^{(l)} S_u(i,j)}{\sum_{i=1}^{N_U} s_i^{(k)} \sum_{j=1}^{N_U} s_j^{(l)}} \tag{16}$$

Unlike S_1, this similarity uses soft URL level similarities. When the two sessions are identical, $S_{2,kl}$ simplifies to $S_{2,kk} = \frac{1}{\sum_{i=1}^{N_U} s_i^{(k)}}$, which can be considerably small depending on the number of URLs accessed. Besides identical sessions, this similarity will generally be underestimated for session pairs who share some identical URLs while the rest of the unshared URLs have low syntactic similarity. In general for such sessions $S_{1,kl}$ provides a higher and more accurate session similarity. Therefore, we define a new similarity between two sessions that takes advantage of the desirable properties of S_1 and S_2 as follows:

$$S_{kl} = \max(S_{1,kl}, S_{2,kl}) \tag{17}$$

For relational clustering, this similarity is mapped to the dissimilarity measure:

$$r_s(k,l) = (1 - S_{kl})^2 . \tag{18}$$

As will be seen in Section VIII, our experiments indicate that this dissimilarity measure is reasonable.

VII. Interpretation of Session Clustering Results

Since both FCMdd and RFCM require that we specify the number of clusters, and this is not possible when clustering user sessions, we used the following procedure: We cluster the sessions with a large (overspecified) number of clusters. (In our experiments, this number was 50.) We then build the Minimum Spanning Tree (MST) [48], [31] of the relational graph generated by FCMdd. Each node in the relational graph represents a cluster generated by FCMdd. Each edge has a weight associated with it, which is equal to the dissimilarity between the two clusters corresponding to the two nodes that the edge connects. (The dissimilarity measure we use is described in the next paragraph.) We then sort the edges of the MST by the dissimilarity values, and find the largest "jump" between two consecutive dissimilarities to determine a dissimilarity threshold T_D. By cutting the tree at dissimilarity threshold T_D, we automatically determine the number of final clusters.

The distance between two fuzzy clusters X_i and X_k generated by FCMdd cannot be meaningfully measured by computing the dissimilarity $r(\mathbf{v}_i, \mathbf{v}_k)$ between their medoids $\mathbf{v}_i$ and $\mathbf{v}_k$. We need to somehow take in account the affinity between the clusters as measured by the strength and extent of the common boundary between them. Since the membership u_{ij} of an object $\mathbf{x}_j$ in cluster X_i depends on the object's position with respect to the medoid of X_i relative to its position with respect to all other medoids, it indirectly encodes the affinity of $\mathbf{x}_j$ with respect to X_i. Let $U_i = (u_{i1}, \ldots, u_{ij}, \ldots, u_{in})$ represent the fuzzy cluster X_i associated with medoid $\mathbf{v}_i$, where u_{ij} is the degree to which object $\mathbf{x}_j$ belongs to X_i. The overlap between two fuzzy clusters U_i and U_k can be measured by the Jaccard index [21]:

$$\frac{|U_i \cap U_k|}{|U_i \cup U_k|}, \tag{19}$$

where

$$\begin{aligned} U_i \cap U_k &= (\min(u_{i1}, u_{k1}), \ldots, \min(u_{in}, u_{kn})), \\ U_i \cup U_k &= (\max(u_{i1}, u_{k1}), \ldots, \max(u_{in}, u_{kn})), \\ &\text{and } |U_j| = \sum_{l=1}^{n} u_{jl}. \end{aligned} \tag{20}$$

We can modify the set dissimilarity $r(\mathbf{v}_i, \mathbf{v}_k)$ between X_i and X_k by taking into account the overlap

as follows:

$$r'(\mathbf{v}_i, \mathbf{v}_k) = r(\mathbf{v}_i, \mathbf{v}_k) \left(1 - \frac{|U_i \cap U_k|}{|U_i \cup U_k|}\right). \tag{21}$$

We use $r'(\mathbf{v}_i, \mathbf{v}_k)$ in constructing the MST. Since we apply MST to the clustered data and not the origianal data, it does not increase the computational complexity of the algorithm.

We interpret results of applying FCMdd or FCTMdd on the user session relational data using the following quantitative measures: First, the user sessions are crisply assigned to the closest clusters based on the memberships. This creates c clusters $\mathcal{X}_i = \left\{\mathbf{s}^{(k)} \in \mathcal{S} \mid r_{ik} \leq r_{jk}\ \forall j \neq i\right\}$, for $1 \leq i \leq c$. The sessions in cluster $\mathcal{X}_i$ are then summarized in a typical session "profile" vector $\mathbf{P}_i = (P_{i1}, \ldots, P_{i_{N_U}})^t$. The components of $\mathbf{P}_i$ are URL weights that represent the "probability of access" of each URL during the sessions of $\mathcal{X}_i$ as follows:

$$P_{ij} = p\left(\mathbf{s}_j^{(k)} = 1 | \mathbf{s}_j^{(k)} \in \mathcal{X}_i\right) = \frac{\left|\mathcal{X}_{i_j}\right|}{|\mathcal{X}_i|}, \tag{22}$$

where $\mathcal{X}_{i_j} = \left\{\mathbf{s}^{(k)} \in \mathcal{X}_i \mid s_j^{(k)} > 0\right\}$. The URL weights P_{ij} measure the "significance" or "strength" of a given URL to the i^{th} profile. Besides summarizing profiles, the components of the profile vector can be used to recognize an invalid profile which has no strong or frequently accessed pattern. For such a profile, all the URL weights will be low.

Several classical cluster validity measures can be used to assess the goodness of the partition. The intra-cluster or within-cluster distance represents an average of the distances between all pairs of sessions within the the i^{th} cluster, and is given by $\overline{D}_{Wi} = \frac{\sum_{\mathbf{s}^{(k)} \in \mathcal{X}_i} \sum_{\mathbf{s}^{(l)} \in \mathcal{X}_i, l \neq k} r_{kl}}{|\mathcal{X}_i|(|\mathcal{X}_i|-1)}$. This is inversely related to the compactness or goodness of a cluster. A good guideline to use when evaluating clusters based on the intra-cluster distances is to compare these values to the total average pairwise distance of all sessions. The latter corresponds to the intra-cluster distance if all the user sessions were assigned to one cluster (i.e., no category information is used). Also, it is important to recall that all distances are in $[0, 1]$. The inter-cluster or between-cluster distance represents an average of the distances between sessions from the i^{th} cluster and sessions from the j^{th} cluster, and is given by $\overline{D}_{Bij} = \frac{\sum_{\mathbf{s}^{(k)} \in \mathcal{X}_i} \sum_{\mathbf{s}^{(l)} \in \mathcal{X}_j, l \neq k} r_{kl}}{|\mathcal{X}_i||\mathcal{X}_j|}$. For a good partition, the inter-cluster distances should be high because they measure the separation between clusters.

VIII. Experimental Results

In this section, we illustrate the use of FCMdd and RFCMdd for several Web mining applications. We also provide comparisons with the well-known RFCM algorithm. In Section VIII-A, we give examples of Web document clustering. In Section VIII-B, we present the results of snippet clustering. In Section VIII-C, we discuss a Web mining application that mines user profiles from access logs.

A. Document Clustering

The first data set is a collection of 1042 abstracts obtained from the Cambridge Scientific Abstract Web site. The abstracts correspond to 10 topics (distance education, filament, health care, intermetallic, laminate, nuclear, aeronautics, plastic, trade, furnace, and recycling). There are about 100 abstracts per topic, but since the abstracts were not carefully chosen, some are outliers. In addition, we deliberately added 20 outliers. The second data set is a collection of 59 HTML documents compiled by 6 students at the Colorado School of Mines. Each student was asked to collect about 10 Web pages related to 6 topics (fish, astronomy, ray tracing, chatroom conversation, neurobiology and sports). For both data sets, the documents were represented by a 10-D vector as explained in Section VI-A. In the case of RFCM, the dissimilarity matrix was generated using the square of the Euclidean distance between the vectors. In the case of FCMdd, in addition to Euclidean distance, several other dissimilarity measures were also tried. In all cases, the nuber of clusters was specified to be equal to the number of topics.

Table I shows the results obtained for the first two data sets. The column with the heading FCMdd-cos corresponds to the case when FCMdd was run with the cosine dissimilarity measure (i.e., 1−cosine of angle between vectors), and the columns with the headings FCMdd-E, FCMdd-E^2, and FCMdd-L_1 correspond to the case when FCMdd was run with the Euclidean distance, squared Euclidean distance, and L_1-norm respectively. The first row results are the average of 20 different runs of the algorithms on the Cambridge Scientific Abstract data set. In each run, 120 abstracts were randomly selected from the collection, and only these 120 abstracts were used to generate the keywords and the 10 eigenvectors. The 500-dimensional feature vectors were constructed for all the remaining abstracts, and these feature vectors were then projected onto the 10 eigenvectors to generate the object data for a particular run of the algorithms. This process was repeated 20 times. The second row of the table corresponds to

the Web-page data set.

The results show that FCMdd compares favorably with RFCM in terms of classification rates, and at the same time is an order of magnitude faster. The CPU times (in seconds) were recorded on a Pentium II, 400 MHz processor, and do not include the time to compute the dissimilarity matrix.

B. Snippet Clustering

We also have results on a collection of snippets corresponding to Web documents that were retrieved by a search engine [54]. However, we will present only one example due to space constraints. The data set we used was a set of snippets corresponding to 200 Web documents retrieved by a search engine in response to the query "salsa". To represent the snippets, we used the "stop-word elimination and stemming" algorithm followed by the IDF method to generate $d = 500$ keywords (See Section VI-A). Each snippet was then represented by a d-dimensional vector, where the $i-$th component represents the (normalized) frequency of occurrence of the $i-$th keyword in the snippet. To compute the dissimilarities between snippets, we used the Jaccard dissimilarity Index in (12). We also tried the cosine measure, but the results were somewhat worse. FCMdd was applied to this data with $c = 7$, and the snippets were crisply assigned to the clusters after the algorithm converged. There were three main clusters with 151, 19, and 19 snippets. The remaining clusters had 1 to 6 snippets in them and therefore we did not consider them as significant. We computed average keyword frequencies for each individual cluster. The top 10 most frequent keywords (along with their frequencies in parentheses) are shown in Table II. These keywords give us a "profile" of the cluster. As can be seen, the first cluster is mostly about hot sauces, the second one is about salsa music, and the third one is about salsa dancing. Since FCMdd is not robust, the first cluster also contains many irrelevant (outlier) snippets. Table IV shows randomly sampled snippets from each cluster. It can be seen that the last two snippets in cluster 1 are outliers.

We applied RFCMdd to the same data with 25% trimming, and found that most of the irrelevant snippets were identified correctly as outliers by the algorithm. For example, RFCMdd eliminates the last two snippets shown in Table IV from cluster 1. Cluster 1 now contains only 65 snippets, and the other two clusters are unchanged. Table III summarizes the results of RFCMdd. If we compare the profile vectors produced by RFCMdd with those of FCMdd, we can see that the profile of cluster 1 is

significantly strengthened by RFCMdd. The remaining clusters are virtually unchanged by RFCMdd, since their already have fairly strong profiles.

C. Mining User Profiles from Access Logs

To validate the capacity of our algorithms to extract user access patterns, we ran several experiments comparing FCMdd with RFCM. Both the naive and the linearized implementations of FCMdd were used. The experiments were done on a number of Web server logs obtained from servers at the University of Maryland - Baltimore County (UMBC). We report here a subset of our experimental results, on log sizes that represent a quarter days activity to five days of activity on the UMBC CSEE Web server. Instead of reporting the number of hits in each log, we report here the number of sessions generated from a log since the clustering algorithm operates on the sessions.

Table V compares the performance of LFCMdd, regular FCMdd, and RFCM on the access log data. As can be seen quite clearly, even though the theoretical complexities of RFCM and FCMdd are the same, i.e., $\mathcal{O}(n^2)$, FCMdd performs significantly better. In the case of FCMdd, we overspecify the number of clusters to be 50. We then apply the MST algorithm as described in section VII. For LFCMdd, a value of 15 for p was found to give good results while still significantly reducing computation time compared to FCMdd. The same number of clusters (50) was used for RFCM as well.

From Table V, we see that for the smaller logs, FCMdd is three to four times faster. As the log size increases, this difference increases as well, becoming 20 times for the 3 day log. In fact, we do not report any results for 4 day and 5 day logs with RFCM, since it was taking an inordinately large amount of time. We can also see that the linearized version of FCMdd is significantly faster than its regular implementation. It is about 4-5 times faster for smaller logs, and becomes 10-12 times faster as the log size grows. As larger logs are sought to be analyzed, this difference will be significant. These data establish fairly clearly that traditional approaches like RFCM are too computationally intensive to deal with the large data sizes that are prevalent in the data mining task. They also show the superior performance of the linearized implementation.

Tables VI and VII show the validity measures (see Section VII) computed on the clusters generated by the algorithms. They clearly indicate that the intercluster distance for most clusters is much

smaller than the intracluster distance, although there are a few exceptions, such as cluster 0. This cluster represented an aggregation of all course related URLs.

Given that in the linearized implementation, we are using only a fraction of the possible medoid candidates at each update step, one might expect that the clusters formed may not be as good the regular implementation. At best, one would hope that we would get clusters that were no worse than the original. Somewhat surprisingly, we found that the LFCMdd also generated better clusters than the regular implementation for many of our logs. This can be seen both subjectively by looking at the URLs in a clusters, and more objectively by the validity measures. Tables VI and VII show that the average intra cluster distance was 0.343 for the linearized version as opposed to 0.435 for the regular implementation. We speculate that this is because of the "local" nature of the search used by the linearized version. The linearized version confines its search for the medoid of a cluster to the neighborhood of the current medoid. This means that given a good initialization, this method can find good "localized" clusters. The regular implementation does a more global search and sometimes could get stuck in strange minima.

We provide brief descriptions of the clusters generated by our algorithm from the Web server logs at *www.cs.umbc.edu* for the month of April, 2000. Table VIII lists the clusters found. Recall that the clusters were generated by overspecifying the number of clusters to be 50 and then determining the actual number c_{actual} using the procedure described in Section VII. When the cardinality of a cluster is too low (in our case when less than 1% of the sessions belonged to a cluster), we discard the cluster as not having enough support. For illustration purposes, we show here the top level URLs found, and in the table show the sum of the strengths of the underlying URLs. For instance, if a cluster has the URL ∼plusquel/ with strength 0.8 and ∼plusquel/cmsc310 with strength 0.7, we will report it here as ∼plusquel with strength 1.5.

The clusters seemed fairly consistent across logs worth several days. In other words, similar (though not identical) clusters were formed as we mined different sized logs from the same time period. In our analysis, we chose to disregard clusters that had a very small cardinality (number of sessions in them). In traditional data mining jargon, one would say that there was not enough support for such clusters. We also ignored those clusters that did not have a strong URL profile as explained in Section VII.

- Clusters 0 (/agentslist/) and 20 (/agents/) contain user sessions that accessed the pages maintained

by the Agents group at UMBC. Agentslist is an archived mailing list about agents, and agents page is a well-known resource for agent information in the DAI and software agent community.

- Cluster 2 (/help/oracle/) represents user sessions that accessed the help pages of oracle8. These are likely to be from students enrolled in the undergraduate and graduate database courses, both of which use oracle.
- Cluster 6 (/www/) represents user sessions that accessed an older version of the CSEE Web pages.
- Cluster 8 (/stephens/) represents user sessions that accessed the home pages of a faculty member to access information about his course in cryptology.
- Cluster 9 (/471/) represents users who want to access the CMSC 471 course pages. It contains hits to pages containing current information, lectures and notes.
- Cluster 10 (/%7Esli2) and Clusters 16 (/∼sli2/, /∼sli2/cube , /∼sli2/plot) correspond to users interested in Java based computer games in a user's page.
- Clusters 11 (/cgi-bin/) represents access to the queries provided on the server using CGI scripts.
- Clusters 15 (/∼plusquel/) represents users accessing the pages of a faculty member, especially course CMSC310 and a topics course on VLSI that the faculty member teaches.
- Cluster 19 (/∼ugrad) represents users who accessed Web pages that provides brochures and admission information for undergraduates.
- Clusters 21 (/courses/) contains user sessions that access information about the courses offered by the CS department. Given the enrollment differences, a larger support is found for /courses/undergrad/ as opposed to /courses/graduate/, as expected.
- Clusters 1 (/∼thurston), 3 (/∼kalpakis), 4 (/∼mshadl1), 7 (/∼tbogar1), 13 (/∼qlu2), 17 (/∼vick) and 18 (/∼mikeg) correspond to user sessions that accessed home pages of individual users.
- Clusters 5, 12, 14,and 22 have cardinalities that are too small to be included in the study.

IX. Conclusions

In this paper, we have presented new relational fuzzy clustering algorithms (FCMdd and RFCMdd) based on the idea of medoids. The worst-case complexity of the algorithms is $\mathcal{O}(n^2)$, which happens while updating the medoids in each iteration. This complexity compares very favorably with other fuzzy algorithms for relational clustering, such as RFCM. However, with minimal changes, the com-

plexity of FCMdd will be $\mathcal{O}(ncp)$, and that of RFCMdd will be $\mathcal{O}(n\log n)$. Moreover, in our experience, these algorithms converge very quickly, in 5 or 6 iterations. Our results show that these algorithms are very useful in Web mining applications such as categorization of Web documents, snippets, and user sessions.

Note that in some applications, the frequency of accesses to a Web page within the same session may be important. In that case, the definition of $\mathbf{s}_j^{(k)}$ should be modified to $\mathbf{s}_j^{(k)}$ = number of times the j^{th} URL is accessed in the k^{th} session. In ongoing experiments, we are looking into a multi-resolution profiling approach where clustering is applied recursively on the profiles found in previous runs.

Acknowledgements

Partial support of this work by the National Science Foundation Grants IIS 9800899 and IIS 9801711 is gratefully acknowledged. Tapan Kamdar helped implement the code and run some experiments. Additional help was provided by Noam Zeilberger and Kejian Hu. The authors would also like to thank the anonymous referees whose comments helped improve the paper.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[2] R. Armstrong, T. Joachims D. Freitag, and T. Mitchell. Webwatcher: A learning apprentice for the World Wide Web. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, pages 6–13, Stanford, CA, March 1995.

[3] G. Arocena and A. Mendelz. Weboql: Restructuring documents, databases, and web. In *Proc. IEEE Intl. Conf. Data Engineering '98*, pages 24–33. IEEE Press, 1998.

[4] P. Bajcsy and N. Ahuja. Location- and density-based hierarchical clustering using similarity analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1011–1015, 1998.

[5] G. Beni and X. Liu. A least biased fuzzy clustering method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16:954–960, September 1994.

[6] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[7] J. Abidi C. Shahabi, A.M. Zarkesh and V. Shah. Knowledge discovery from users web-page navigation. In *Proceedings of the Seventh IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE)*, pages 20–29, Birmingham, UK, 1997.

[8] J. Chen, A. Mikulcic, and D. H. Kraft. An integrated approach to information retrieval with fuzzy clustering and fuzzy inferencing. In O. Pons, M. Ampara Vila, and J. Kacprzyk, editors, *Knowledge Management in Fuzzy Databases*, volume 163. Physica Verlag, Heidelberg, Germany, 2000.

[9] M.S. Chen, J.-S. Park, and P. S. Yu. Efficient data mining for path traversal patterns. *IEEE Trans. Knowledge and Data Engineering*, 10(2):209–221, April 1998.

[10] R. Cooley, B. Mobasher, and J. Srivastav. Web Mining: Information and pattern discovery on the World Wide Web. In *Proc. IEEE Intl. Conf. Tools with AI*, pages 558–567, Newport Beach, CA, 1997.

[11] R. N. Davé and R. Krishnapuram. Robust clustering methods: A unified view. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, 1997.

[12] E. Diday. La methode des nuees dynamiques. *Rev. Stat. Appliquee*, XIX(2):19–34, 1975.

[13] D. Riecken: Guest Editor. Special issue on personalization. *Communications of the ACM*, 43(9), Sept. 2000.

[14] J. Fink, A. Kobsa, and J. Schreck. Personalized hypermedia information provision through adaptive and adaptable system features. *http://zeus.gmd.de/hci/projects/avanti/publications/ISandN97/ISandN97.html*, 1997.

[15] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Academic Press, San Diego, CA, 1982.

[16] K. C. Gowda and E. Diday. Symbolic clustering using a new similarity measure. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:368–377, 1992.

[17] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient algorithm for large databases. In *Proceedings of SIGMOD '98*, pages 73–84, Seattle, June 1998.

[18] R. J. Hathaway and J. C. Bezdek. Switching regression models and fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 1(3):195–204, 1993.

[19] R. J. Hathaway and J. C. Bezdek. NERF c-means: Non-Euclidean relational fuzzy clustering. *Pattern Recognition*, 27:429–437, 1994.
[20] R.J. Hathaway, J.W. Devenport, and J.C. Bezdek. Relational dual of the c-means clustering algorithms. *Pattern Recognition*, 22(2):205–212, 1989.
[21] P. Jarccard. *Nouvelles recherches sur la distribution floral.* Bull. Soc. Vard. Sci. Nat., Paris, 1908.
[22] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proc. 15th International Conf. On Artificial Intelligence*, pages 770–775, Aug. 1997.
[23] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–20, 1972.
[24] A. Joshi, C. Punyapu, and P. Karnam. Personalization and asynchronicity to support mobile web access. In *Proc. Workshop on Web Information and Data Management, 7^{th} Intl. Conf. on Information and Knowledge Management*, pages 17–20, November 1998.
[25] A. Joshi, S. Weerawarana, and E. Houstis. On disconnected browsing of distributed information. In *Proceedings of IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE)*, pages 101–108, Birmingham, UK, 1997.
[26] L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. In Y. Dodge, editor, *Statistical Data Analysis Based on the L_1 Norm*, pages 405–416. North Holland/Elsevier, Amsterdam, 1987.
[27] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data, An Itroduction to Cluster Analysis.* John Wiley & Sons, Brussels, Belgium, 1990.
[28] J. Kim, R. Krishnapuram, and R. N. Davé. Application of the least trimmed squares technique to prototype-based clustering. *Pattern Recognition Letters*, 17:633–641, 1996.
[29] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.
[30] R. Krishnapuram and J. M. Keller. The possibilistic c-means algorithm: Insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.
[31] S. Miyamoto. *Fuzzy Sets in Inforamtion Retrieval and Cluster Analysis.* Kluwer Academic, Boston, 1990.
[32] O. Nasraoui and R. Krishnapuram. A genetic algorithm for robust clustering based on a fuzzy least median of squares criterion. In *Proceedings of NAFIPS'97*, pages 217–221, Syracuse, NY, Sept. 1997.
[33] O. Nasraoui and R. Krishnapuram. Mining web access logs using a relational clustering algorithm based on a robust estimator. In *Proc. of the Eighth International World Wide Web Conference*, pages 40–41, Toronto, 1999.
[34] O. Nasraoui, R. Krishnapuram, and A. Joshi. Relational clustering based on a new robust estimator with application to web mining. In *Proceedings of the North American Fuzzy Information Society Workshop International World Wide Web Conference*, pages 705–709, New York City, 1999.
[35] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference*, pages 144–155, Santiago, Chile, Sept. 1994.
[36] O.Zaiane and J. Han. Webml: Querying the world-wide web for resources and knowledge. In *Proc. Workshop on Web Information and Data Management, 7^{th} Intl. Conf. on Information and Knowledge Management*, pages 9–12, 1998.
[37] M. Pazzani, L. Nguyen, and S. Mantik. Learning from hotlists and coldlists: Towards a WWW information filtering and seeking agent. In *Proceedings of Tools with Artificial Intelligence '95*, pages 492–495, Washington, DC, 1995.
[38] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Proc. AAAI 98*, pages 727–732, 1998.
[39] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, 118:245–275, 2000.
[40] G. D. Ramkumar and A. Swami. Clustering data without distance functions. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21:9–14, 1998.
[41] M. Roubens. Pattern classification problems and fuzzy sets. *Fuzzy Sets and Systems*, 1:239–253, 1978.
[42] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley and Sons, New York, 1987.
[43] T. A. Runkler and J. C. Bezdek. ACE: A tool for clustering and rule extraction. *IEEE Transactions on Fuzzy Systems*, 1999.
[44] E. H. Ruspini. Numerical methods for fuzzy clustering. *Information Science*, 2:319–350, 1970.
[45] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.
[46] S. Sen and R. N. Davé. Clustering of relational data containing noise and outliers. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 1411–1416, Anchorage, May 1998.
[47] U. Shardanand and P. Maes. Social information filetering: Algorithms for automating 'word of mouth'. In *Proc. CHI'95 Conference on Human Factors in Computing Systems*, New York, 1995. ACM Press.
[48] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy - The Principles and Practice of Numerical Classification.* W. H. Freeman, San Francisco, 1973.
[49] Y. El Sonbaty and M. A. Ismail. Fuzzy clustering for symbolic data. *IEEE Transactions on Fuzzy Systems*, 6:195–204, 1998.
[50] L. Terveen, W. Hill, and B. Amento. PHOAKS - a system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.
[51] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, San Diego, CA, 1999.
[52] M. P. Windham. Numerical classification of proximity data with assignment measures. *Journal of Classification*, 2:157–172, 1985.
[53] O.R. Zaiane, M. Xin, and J. Han. Discovering web access patterns and trends by applying olap and data mining technology on web logs. In *Proc. Advances in Digital Libraries Conf. (ADL'98)*, pages 19–29, 1998.
[54] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of SIGIR'98*, pages 46–54, Melbourne, Australia, Aug. 1998.

TABLE I

RESULTS ON ABSTRACT AND WEB-PAGE DATA SETS

Data Set	RFCM-E^2		FCMdd-cos		FCMdd-E		FCMdd-E^2		FCMdd-L_1	
	rate (%)	CPU (s)	rate (%)	CPU (s)	rate (%)	CPU (s)	rate (%)	CPU (s)	rate (%)	CPU (s)
Abstr	83.60	126.88	83.10	5.03	74.55	4.20	83.97	5.20	75.52	3.77
Web Pg	83.05	0.34	84.75	0.01	84.75	¡0.01	84.75	0.01	88.14	0.01

TABLE II

FCMDD RESULTS ON SNIPPET DATA SET

cluster	kw1	kw2	kw3	kw4	kw5	kw6	kw7	kw8	kw9	kw10
1	hot (.35)	sauc (.23)	thi (.21)	chile (.19)	recip (.15)	food (.13)	gourmet (.13)	garlic (.11)	spice (.10)	mexican (.09)
2	music (1.0)	danc (.63)	dj (.47)	latin (.42)	onli (.26)	todai (.21)	lesson (.16)	listen (.16)	live (.16)	floor (.16)
3	danc (1.4)	lesson (.31)	look (.32)	jeanni (.21)	gonzalez (.16)	jobi (.16)	london (.16)	lui (.16)	oxford (.16)	page (.16)

TABLE III

RFCMDD RESULTS ON SNIPPET DATA SET

cluster	kw1	kw2	kw3	kw4	kw5	kw6	kw7	kw8	kw9	kw10
1	hot (.82)	sauc (.52)	thi (.48)	chile (.43)	gourmet (.29)	garlic (.25)	food (.23)	recip (.23)	pepper (.19)	mexican (.17)
2	music (1.0)	danc (.63)	dj (.47)	latin (.42)	onli (.26)	todai (.21)	lesson (.16)	listen (.16)	live (.16)	floor (.16)
3	danc (1.4)	lesson (.31)	look (.32)	jeanni (.21)	gonzalez (.16)	jobi (.16)	london (.16)	lui (.16)	oxford (.16)	page (.16)

TABLE IV

SAMPLE SNIPPETS IN 3 CLUSTERS FOUND BY FCMDD

cluster #1

1) Vermont maple syrup, gift baskets, hot sauce, salsa, new England specialty products, gourmet foods, corporate, sweets]. Vermont maple syrup, gift baskets, hot sauce, salsa, new England specialty products, gourmet foods, corporate,

2) HOT SAUCE. #299 Devil's Dozen! $38.00 Click here to see a few of our hot sauces! #201 Batten Island Gourmet, Mild $2.00 closeout - only 1 left #211 Mountain Man Fire Roasted Habanero 3.95$MountainManisourbestsellinghabsauce!$#212$PurpleHaze$3.95

3) Made with searing red savina chiles, hot habanero chiles and thai chiles makes this salsa hotter then you know where. The Paradise Pineapple Salsa was awarded first place in the 1996 Fiery Foods Festival-fruit salsa division.

4) Kaari Tiistai 17.00-18.00 SALSA Alkeet Miguel 18.00-19.00 PARISALSA Alkeet Miguel 19.00-20.00 OPISKELE ESPANJAA Salsaamalla Alkeet (tunti pidet espanjaksi) Miguel 20.00-21.00 RUEDA de CASINO & PARISALSA Jatko Miguel Keskiviikko 17.00-18.00 FLAMENCO Alkeet (sevillanas) Kaari 18.00-19.00 FLAMENCO Jatko (tangos) Kaari 19.00-20.00.

5) By combining a multi-faceted corporate marketing background with the latest in computer animation and presentation techniques, ROUTE 66 PRODUCTIONS, LLC has formed a division specializing in sophisticated digital media applications, called DIGITAL SALSA.

cluster #2

1) Edinburgh Latin/Hispanic Music and Dance Service: Tel-Aviv Yigal Korolevski has kindly send me a summary of what's going on in Tel-Aviv. Wednesday - 21:00-22:00 (beginners), 22:00-24:00 Brazilian party.

2) A collection of the best music played at Tropicana famous dancing floor. Compositions by Ignacio Pieiro, Ernesto Lecuona, Enrique Jorrn, the father of cha cha ch, Celina Gonzlez, Miguel Matamoros. They received the award "Disco de Oro" for their

3) DJ Jesus R. R. DJ Jesus plays all the hottest Latins tunes: salsa, cumbia, merengue, banda, quebradita, ranchera and romantica. Available to DJ latin techno and salsa hits for any occasion. El Tigre" DJ Manny - LATIN MUSIC PRODUCTIONS. Phone: 283-4213. 3213

cluster #3

1) The instructor, Gustavo Sr. has been teaching authentic Latin dances in the Seattle area for many years, and travels often to his native Panama, to constantly update his knowledge of the true international dance hall scene. The instructor, Gustavo Sr. has

2) What I really love, is watching great salsa dancing - I wish I could dance well but need a lot of lessons and practice (so does my novia who is Guatemalteca). What I really love, is watching great salsa dancing - I wish I could dance well but need a lot of lessons and practice (so does my novia who is Guatemalteca).

3) After a few lessons you'll have stepped into a whole new scene - you'll quickly make new friends, and before long you'll be dancing the nights away at one of Viva Salsa's events or at one of the many salsa clubs in Oxfordshire, the home counties, London, New

TABLE V

TIME REQUIRED BY THE CLUSTERING ALGORITHMS, SHOWN AS H:M:S.MS

Log Size	Sessions	LFCMdd	FCMdd	RFCM
Quarter day	1605	2:34	9:42	46:01.97
Half day	2668	10:09.11	42:41	2:06:31.34
1 day	2913	10:52.30	55:27.25	2:52:59.37
2 day	5956	14.45.74	2:34:04.32	8:39:45.24
3 day	8777	30:31.11	5:33:40.05	107:17:45.4
4 day	11074	49:22.25	9:27:02.58	–
5 day	12586	1:09:50	12:24:04	–

Cluster	Intra-Cluster Distance	Inter-Cluster Distance
0	0.986	0.999
1	0.137	0.994
2	0.262	0.995
4	0.025	0.999
5	0.086	1.000
6	0.365	0.998
7	0.072	0.999
8	0.080	0.999
9	0.702	0.998
10	0.481	0.993
11	0.184	0.998
12	0.090	0.996
13	0.454	0.996
14	0.542	0.995
15	0.405	0.998
16	0.284	0.992
17	0.333	0.997
18	0.400	0.993
19	0.400	0.995
20	0.204	0.998
21	0.209	0.999
22	0.858	0.984
Average	0.343	0.996

TABLE VI

INTRA– AND INTER–CLUSTER DISTANCE FOR LFCMDD GENERATED FOR A QUARTER DAY LOG

Cluster	Intra-Cluster Distance	Inte-Cluster Distance
0	0.984	0.997
1	0.025	0.999
2	0.518	0.994
3	0.448	0.996
4	0.371	0.998
5	0.687	0.997
6	0.429	0.998
7	0.090	0.993
9	0.674	0.988
10	0.342	0.995
11	0.405	0.997
12	0.487	0.995
13	0.311	0.999
14	0.128	0.999
16	0.514	0.989
17	0.212	0.997
20	0.635	0.980
22	0.463	0.995
23	0.438	0.991
24	0.556	0.991
Average	0.435	0.995

TABLE VII

Intra– and Inter–Cluster Distance for FCMdd generated for a Quarter Day Log

TABLE VIII
CSEE Logs Analysis using Linear FCMdd Algorithm

Cluster	Cardinal	URLs	URLs	Deg
0 - /agentslist/	628	3864	{/agentslist/*}	1.511
			{/agentslist/archive/*}	1.449
1 - /~thurston/	11	8	{/~thurston/*}	2.999
			{/~thurston/swarch.htm/*}	1.273
2 - /help/oracle8/	41	241	{/help/*}	6.976
			{/help/oracle8/*}	6.585
			{/help/oracle8/server803*}	4.488
			{/help/oracle8/server803/A54654_01/*}	1.098
3 - /~kalpakis/	46	137	{/~kalpakis/*}	3.543
			{/~kalpakis/Courses/*}	1.761
			{/~kalpakis/Courses/441/*}	1.326
4 - /~mshadl1/	20	12	{/~mshadl1/*}	3.2
			{/~mshadl1/profiler.html/*}	1.7
5	*1*	*2*	*/~cmcnau2/**	
6 -/www/	32	104	{/www/*}	3.156
			{/www/graduate/*}	2.531
			{/www/graduate/rpg/*}	1.156
7 - /~tbogar1/	18	42	{/~tbogar1/*}	4.722
			{/~tbogar1/dlance/*}	1.222
8 - /~stephens/	27	150	{/~stephens/*}	8.444
			{/~stephens/crypto/*}	7.074
9 -/471/	60	893	{/471/*}	14.167
			{/471/current/*}	5.45
			{/471/current/lectures/*}	4.333
			{/471/lectures/*}	5.25
			{/471/lectures/uninformed-search/*}	1.05
			{/471/notes/*}	3.333
			{/471/notes/7/*}	1.05
10 -/%7Esli2/	16	24	{/%7Esli2/*}	2
			{/%7Esli2/cube/*}	1.938
11 -/cgi-bin/	76	358	{/cgi-bin/*}	4.171
			{/cgi-bin/raw?url=http:/*}	3.776
			{/agents/*}	2.697
12	*3*	*5*	*/~kjoshi/**	
13 -/~qlu2/	49	12	{/~qlu2/*}	1.837
14	*6*	*42*	*/~khu1/* ,/~lxu21/*, /courses/**	
15 -/~plusquel/	12	85	{/~plusquel/*}	8.0
			{/~plusquel/310/*}	3.417
			{/~plusquel/310/nasm/*}	1.083
			{/~plusquel/310/syllabus/*}	1.166
			{/~plusquel/310/slides/*}	1.166
			{/~plusquel/vlsi/*}	1.333
			{/~squire/*}	2.5
16 -/~sli2/	236	203	{/~sli2/*}	13.86
			{/~sli2/cube/*}	9.932
			{/~sli2/plot*}	2.085
17 -/~vick/	9	20	{/~vick/*}	2.778
18 -/~mikeg/	81	90	{/~mikeg/*}	3.012
19 -/~ugrad/	25	75	{/~ugrad/*}	4.84
			{/~ugrad/brochure/*}	3.92
			{/~ugrad/brochure/cmpe/*}	1.76
			{/~ugrad/brochure/cmsc/*}	1.52
			{/courses/*}	1.52
20 -/agents/	466	1544	{/agents/*}	3.406
21 -/courses/	1048	3443	{/courses/*}	7.889
			{/courses/undergraduate/*}	6.776
			{/courses/undergraduate/201/*}	2.512
22	*2*	*127*	*/%7Echang/**	