A Non-Parametric Approach to Web Log Analysis*

Andrew Foss, Weinan Wang, Osmar R. Zaïane
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6E 2E8
{afoss,weinan,zaiane}@cs.ualberta.ca

Abstract: Clustering data generally involves some input parameters or heuristics that are usually unknown at the time they are needed. We discuss the general problem of parameters in clustering and present a new approach, TURN, based on boundary detection and apply it to the clustering of web log data. We also present the use of different filters on the web log data to focus the clustering results and discuss different coefficients for defining similarity in a non-Euclidean space.

1 Introduction

We are interested in the clustering of web users or web access sessions in the context of web applications such as distance education web-based systems and e-commerce sites. This pertains to what is usually refered to as web usage mining, data mining from large web access logs. Web logs represent a vast resource of information but one that can only be tapped through the techniques of data mining. An important technique of data mining is clustering which can reveal usage patterns for the benefit of site managers such as the educators whose logs we studied. The results of clustering can give insight to the users' behaviour in a web site and have significant applications in personalization, recommendation systems, adaptive sites, etc. Unfortunately, current clustering algorithms require the user to specify some parameters before the processing can be undertaken but without running the algorithm it is difficult to choose suitable values, especially for those unacquainted with data mining. We sought to find a new algorithm that does not require input parameters. In the particular application we are developing, an intelligent web-based learning environment for distance education, a major user is the educator trying to evaluate the learning process of on-line students based on access history from web logs. This specific user is not necessarily savvy in data mining techniques, and thus requesting parameters to oversee and control the mining process would be cumbersome and even objectionable.

Web usage mining is the application of data mining techniques on web access logs. Most of the current studies in this area are very new, but more and more work is being done. [14] is a recent survey paper which discusses some concept definitions and provides an up-to-date survey of Web Usage mining. [8] described the architecture of the WebMiner system, one of the first systems for Web Usage mining. WebLogMiner described in [15] uses a multidimentional datacube approach with on-line analytical mining to discover pertinent patterns in web logs but does not perform clustering per se. [5] is a paper defining user sessions and discussing clustering user sessions based on the pair-wise dissimilarities using a robust fuzzy clustering algorithm. There is also interesting work on personalization [9] and web adaptation and evaluation [6] relevant to web usage clustering. Overall there are few reports on real applications so far. An interesting clustering experiment on web log transactions using the known BIRCH algorithm was reported in [2], but the clustering is performed on generalized transactions using concept hierarchies of pages in a site.

In this paper we assert that the most natural way of determining whether data point A lies in cluster X or cluster Y is by determining the natural

*Research is supported in part by TeleLearning-NCE (Canadian Networks of Centres of Excellence).

boundary between X and Y. If that is found, then the data point can be easily assigned to the appropriate cluster. If X and Y are areas of high density of data points, then somewhere between them the data point density will decline to a minimum and start rising again. This is the natural cluster boundary and is an example of a turning point or minimum in the distribution. Since, locally, there can be many turning points, the main challenge is to rank the turning points. Examples of major and minor turning points discovered by our algorithm TURN are seen in Figure 1. Ranking the turning points allows us to discover the interesting levels of resolution and the clustering at each level. Web log mining presents an example of a non-Euclidean space to which we applied our approach. This developed approach is the main concern of this paper. While we have applied it to web log mining, our algorithm TURN can be used for clustering an arbitrary data space.

In the next section we give more motivation for our research and examine some initial concepts supporting our algorithm. We discuss in Section 3 web usage mining, in particular cleaning and filtering of the data and the methods tried for determining the similarity between sessions and users. Filtering is important because it provides user control over TURN's output. Section 4 introduces the algorithm for TURN, discusses its costs, and compares it with the ROCK algorithm, an efficient algorithm for clustering categorical data. A cluster quality measure is devised for comparison. Section 5 presents our results using TURN and ROCK for clustering, including the use of TURN to automate parameter discovery for ROCK.

2 Preliminaries

A large number of problems involve the clustering and classification of data. Clustering is essentially a problem of boundary determination but most clustering algorithms don't attempt to do this directly. The human eye and optic processor actually only sees boundaries, working somewhat like a video games renderer that defines a polygon (the boundary) and then calls a fast fill routine. Further, given a pattern, a human asked to group the points would look for the spaces between groups. When referring to turning points, we refer only to these minima in the distribution. The difficulty arises because in any real-world distribution there are many turning points reflecting different levels of resolution. Small variations could be declared noise but one person's noise is another's important data. For instance, the Cosmic Microwave Background was first seen as noise but now we know that its very fine variations should reveal the distribution of matter at a very early stage of the universe's history.

Current approaches prefer to apply some heuristic such as a threshold beyond which a boundary is presumed to exist. Examples of this are k-means [4], CLARANS [10], BIRCH [16], CURE [11] , ROCK [3], etc. For all these methods, the user needs to specify the number of clusters and often a threshold value. The difficulties with this are obvious. For instance, the heuristic applied at the beginning is essentially a guess and is unlikely to be optimum and even if an optimum is found for one part of the distribution it may not represent other parts well.

One algorithm, WaveCluster [13], attempts to find cluster boundaries without the usual parameters. After digitising the data, which, in itself, involves a heuristic as this removes a certain level of granularity in the data, WaveCluster applies a filter (another potential imposed bias) and then uses wavelets to seek high frequency changes in the distribution which will indicate sharp changes or boundaries. This algorithm could well be adapted to scan different resolutions automatically but, as presented, the resolution viewed is a user input parameter. As far as we know WaveCluster is yet to be adapted for non-Euclidean distributions.

Current clustering algorithms including WaveCluster involve certain heuristics. Our approach is to detect turning points (minima) without applying any smoothing or noise filters and then apply the approach recursively. To accomplish this, we repeatedly differentiate the series and look for changes of sign. We refer to such a change of sign as a 'turn'. If one differentiates n times, then 2^n points are involved in a turn so the view becomes increasingly global removing fluctuations that involve smaller numbers of points. Because we use turning points to define clusters we call our approach TURN. It does not depend on any parameters even in the pre-processing stage.

Figure 1 illustrates how TURN picks out cluster

boundaries and assigns an amplitude to the change in a Euclidean distribution such as a time series. In this graph, the data is a time series with a couple of events where the measured attribute rose sharply and many minor fluctuations. The bottom graph shows peaks or spikes at the turning points found in the upper graph. The major events in the upper graph are picked out by the largest spikes, which then allow us to group the points in between as clusters. Within this we can see smaller peaks indicating another level of structure and even within the background there are small spikes which in some experimental situations, such as astronomy, could represent interesting events. It can be observed that some of the larger secondary spikes do not indicate sharp changes in the distribution but rather the beginning of a persistent upward or downward trend. The heights of the peaks are used to rank the turns and the algorithm is used recursively to detect clusters within this distribution.

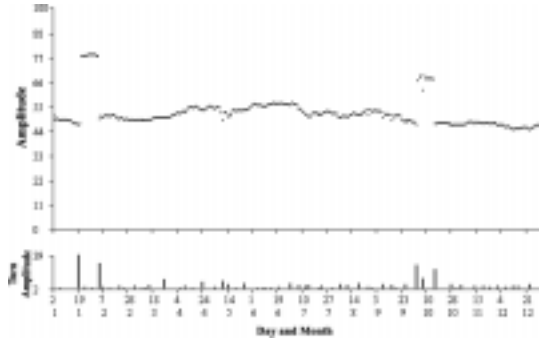

Figure 1: TURN detects cluster boundaries in a time series

The algorithm presented here is adapted to the problem of a non-Euclidean space as in the case of web log mining. Web log analysis presents a particular challenge because there is no straightforward way of defining a Euclidean distribution among the data points. Each data point is in fact a graph consisting of many URLs so there is no mean or medoid. This precludes the use of clustering algorithms which depend on these such as k-means [4], CLARANS [10], BIRCH [16], etc. However, one can use measures of similarity to define distance between data items (sessions or users). This also means that the distribution of points with respect to one point is a series starting from 100% similar and declining to 0% similar. There are no minima but there are turning points in the differentiated series at any level.

In the work reported here we took the rate of change of the acceleration or second differential of the series. At this level, flattening of the curve after a decline produces a spike or change in sign in the differentiated series. This can be seen in Figure 2. The third differential was chosen because a change of sign at this level rather clearly corresponded to a visual event or 'boundary' in the series. There are many spikes of different sizes corresponding to different degrees of sharpness of the turns. We found from observation that in this web log analysis case that it was sensible to take the first turn or 'spike', whatever its frequency. This approach entirely removed any need for parameterization in the boundary discovery but did not test our algorithm for resolution discovery, which remains a work in progress. We are also studying the use of differentiation at different levels.

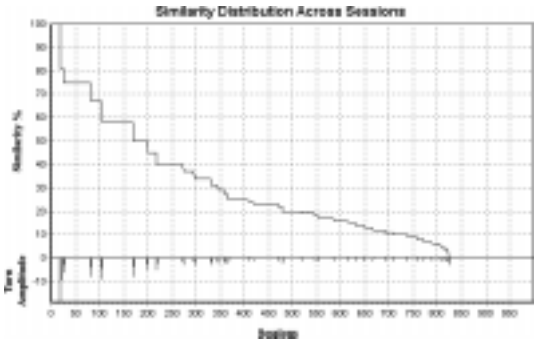

Figure 2: Distribution obtained with Jaccard coefficients and turns found by TURN (negative spikes)

Figure 2 illustrates TURN applied to a non-Euclidean distribution. The figure shows a typical plot of similarity between one session and 999 others, given 1000 sessions. When TURN finds a turn, it produces a spike with an amplitude related to the importance of the turn. Here these spikes are shown as negative values. Typically, there are some sessions that have zero distance from the reference session and then there is a sharp drop off followed by a series of smaller adjustments until it drops to zero similarity.

One recent algorithm which is well suited to non-Euclidean spaces is ROCK[3]. ROCK clusters a sample of the data using an arbitrary similarity

measure that returns a value between 0 and 1 and proceeds hierarchically by merging clusters with common neighbour exceeding a given threshold θ until reaching a fixed number k of desired clusters. We choose to compare our algorithm TURN with ROCK since ROCK handles categorical data. However, our algorithm does not sample the data but considers all data points.

3 Web Usage Mining

Web usage mining allows for the discovery of patterns in the behaviour of visitors to a web site allowing management to optimize the site for the benefit of visitors. Cluster discovery is an important part of finding patterns. The mining usually has three main steps: 1) data preprocessing, 2) data mining, and 3) pattern evaluation. In our work, the data preprocessing step contains two phases: data cleaning, which is done automatically using some heuristics, and data filtering, which is under user control. Data cleaning involves the removal of entries which contain an error flag, requests for images and other embedded files, applets and other script codes, requests generated by web agents such as web crawlers, etc. Some entries are also transformed into 'actions'. For example a CGI script call with given parameters could be replaced with the action pertaining to the script.

After cleaning the data we then seek to identify useful groupings of the transactions. We identified both users and sessions. Fu et al. [2] describe how these are identified. They employed Attribute Oriented Induction and the clustering algorithm BIRCH [16]. This scaled well over increasingly large data sets and produced meaningful clustering results. However, BIRCH involves the setting of a threshold to determine 'closeness' as well as its sensitivity to the order of data input. [2] does not discuss how they set the threshold or how the 'closeness' between sessions was computed. In this paper, we seek to avoid user defined thresholds as well as offering additional options to the user beyond but including Attribute Oriented Induction. Mobasher et al. [7] used clustering on a web log using the Cosine coefficient and a threshold of 0.5. No mention is made of the actual clustering algorithm used as the paper is principally on Association Rule mining. Our results with ROCK on the site we investigated suggest that 0.5 may be somewhat low.

Transactions (cleaned web log entries) are grouped into sessions and sessions are grouped, if desired, by user in order to investigate patterns in the usage of the site in individual user sessions or over time by individual users. Prior to seeking any rules, it is desirable to find if there are any clusters or groupings within this data. We might find, for instance that many of the visitors to a particular university course web page also accessed some help page. This might identify a need which the administration could respond to.

Data cleaning is followed by user-controlled filtering. Many filters could be defined. Some obvious ones, are 'generalize to level', 'remove duplicate pages', 'remove duplicates to levels', and 'remove short duration pages'. 'Generalize to level' would mean that all URLs would be generalized to a certain level of the site tree, essentially Attribute Oriented Induction [2]. For example, on a university web site computer science courses might appear on the third level of the university domain - Root/Department/Course. By generalizing to the third level and removing duplicates, one would be able to cluster all sessions or users primarily viewing one course.

Removing short duration pages is a very logical choice to eliminate click-throughs and is generally more effective than Maximal Forward Reference (MFR) [1] for this. Applying MFR was investigated in our study and has very little effect on sites with a strong horizontal movement component. Most web sites today are like this as they contain links on each page to most other pages.

The interface of our application offers the user various ways of filtering the transactions. Here we define a transaction as a cleaned URL in the log file. All filters apply to a single session or user depending on which filters the user selects to cluster. The user can also choose the similarity coefficient to apply during clustering. Various coefficients have been proposed in the literature such as Jaccard [3], Dice [12], Cosine [12] and Overlap [12]. Jaccard and Dice have been found to be functionally equivalent so we did not consider Dice. The Overlap coefficient tends to yield a much larger number of identical items, which could lead to very large clusters.

For instance, a session consisting of A/B/C will be judged identical to a session consisting of A/B/C, A/B/C/D, and any number of other URLs. We feel this is likely to give results that a user might judge as extraneous. We compared Jaccard with Cosine and found little difference in the results so only Jaccard was used for the work presented in this paper.

4 TURN clustering algorithm

To decide which cluster to assign a data point to, we need to locate the natural boundary between the clusters. If shown a picture with patches of black on it, we would naturally define them as clusters by identifying white or grey areas between the darker ones. This amounts to searching for the minima in the distribution of 'blackness' - turning points. TURN is an attempt to do this and in the case of web log analysis, it seeks to find the sessions/users close to each other by grouping sessions that fall within the range of the 1st turning point. The distribution is differentiated n times looking for a change of sign which we take as a 'turn'. In this application where a similarity measure is used and the distances are sorted to find the closest items, changes of sign only occur for $n \geq 3$. The sign changes for $n = 3$ identify boundary type events as can be seen in Figure 2 and only the first turn was used as beyond that cluster quality declined sharply. This is a result of the algorithm which, like ROCK, joins neighbours of neighbours and as the distance between an item and its neighbours increases the cluster spreads out and it becomes increasingly difficult to see why the items have been grouped. This method gives a more global solution and is a valuable feature of both ROCK and TURN but can also be a problem if the distance between a point and its neighbours is too great.

More generally in other applications, $n \geq 3$ and turns are ranked to discover meaningful levels of granularity within the data, which may ultimately involve a heuristic to terminate the search. In this present application, beside the choices just listed, the approach is entirely free of parameters. User control is provided by the user by the choice of filters and similarity coefficients.

The version of the algorithm used here is:

1 Select an unclassified item A as a seed;

2 Compute all distances between A and all other non-classified items;

3 Sort this result;

4 Search through the sorted result until the first 'turn';

5 Classify all items up to the first 'Turn' into the same cluster as A;

6 Take each item classified as the seed and iterate recursively the above until no new items are added.

The sorting of the difference data introduces an $O(nlog(n))$ cost. This could be reduced substantially by introducing a heuristic that the first turn must be within 50% similarity and thus any data with similarity $< 50\%$ can be discarded. In resource bounded reality, heuristics are always attractive. Other than that, TURN's complexity is O(kn) where $1 \leq k \leq n$ and k is a function of the number of clusters and their sizes. It goes to $O(n)$ when only one cluster is found and $O(n^2)$ when every point is deemed an outlier.

TURN's memory requirements are $O(n)$. Since applying certain sensible filters can reduce the number of transactions by 75% or more, very large logs can still be held in main memory. Even if the web log is too large to be held in memory, part can be held on disk. As TURN classifies sessions/users these no longer need to be held in memory so at some point in processing, no further disk accesses are required.

ROCK [3] is a robust agglomerative heirarchical clustering algorithm, which is particularly suitable for clustering categorical attributes. Since session information is also a kind of categorical attribute, ROCK is suited to clustering web log data. ROCK's cluster similarity is based on the number of shared neighbours and it follows an iterative process of merging clusters pairs on the basis of a measure of best merging 'goodness'. This iteration will terminate when the number of remaining clusters has reached the specified number, or when no further clustering can take place.

ROCK, like TURN can find clusters of arbitrary shape and can be applied in a non-Euclidean space, unlike many clustering algorithms. Its computation complexity is $O(n^3)$ and the memory space

complexity of the algorithm is $O(n^2)$. Because of this ROCK clusters on a sample. This and the choice of a threshold for choosing neighbours make ROCK rather unstable. Changes in the parameters can substantially affect the clustering result. It is difficult for the user to predict what parameters are appropriate especially for a non-Euclidean space which is much more difficult to conceive.

For this reason we were looking for a method for finding parameters automatically and initially developed TURN to find parameters for ROCK. While the method we employed for doing this is rather simple, the results as shown in Table 4 indicate how TURN has adjusted to the changing of filters and produced quite consistent results. Even the best parameters for ROCK, give less consistent results across filter choices.

The method of parameter discovery was to let TURN count the number of turns in the similarity distribution for a sample of the data set and compute the mean of these. As many of these turns may not be important, this gives us a larger figure than ROCK is likely to find but as ROCK stops before reaching the user defined cluster number when it can not find an improvement by further clustering, this was an effective if not very rigorous approach. We also took the mean of the thresholds found for the first turn and used that as the threshold for ROCK. Optimising this choice might well improve the results. We compared these results to those for TURN and ROCK given different threshold values.

We developed an interface to give the user immediate access to the functionality with views on the site and on the consequences or results of any action. The interface lets the user open a cleaned web log and look at any session or user and compare it with an other and see the similarity computed between them with any coefficient of similarity selected. It is also possible to see the similarity or distance between any one session or user of the web site and all others and the points marked as 'turns' by TURN. This can be viewed either in a text window (Figure 3) or graphically (Figure 2).

The ability to easily study sessions from the site allows the user to quickly determine sensible settings for the filters, level of generalisation (Attribute Oriented Induction) and other options.

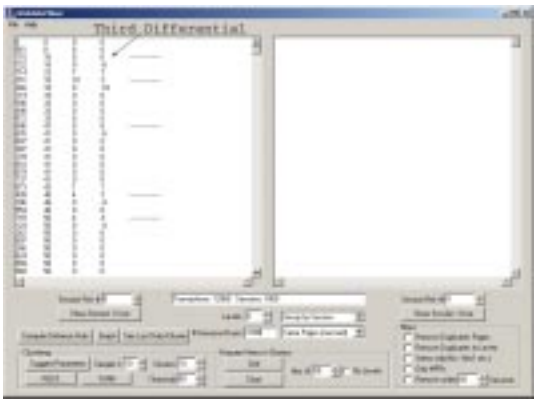

Figure 3: Visualization of turning points on the third differential

5 Results

We used a web log from a on-line university containing transactions of registered learners accessing a variety of courses and on-line activities. After cleaning of the log data, the user of our interface can choose between various similarity coefficients and various filters before clustering. There are a very large number of possible combinations. Of the similarity coefficients, we found the Jaccard coefficient to be the most useful. The only coefficient that produced significantly different results was the Overlap coefficient and, as we have explained above, this causes very different sessions (intuitively) to be classified as similar. Thus the results presented here are for the Jaccard coefficient which simply computes the intersection of two sets of URLs over their union. Which parts of the URLs and which URLs we select from within any session are determined by the filters.

We studied three filter combinations: 1) Remove short duration pages ($<$ 60 seconds); 2) Filter 1 plus 'remove duplicate pages'; and 3) Filter 2 plus 'remove duplicates after generalizing to the third level'. Generalizing to any level is the process of Attribute Oriented Induction discussed above. It lets us group sessions by the visits to a certain nodal level of the site tree. For instance to the course level or chapter level on an academic site. This reduced the number of transactions on our test web log by 75.6%, 86.4%, and 92.2% respectively.

In order to assess the clustering of the two algo-

rithms TURN and ROCK, we computed a cluster quality measure. This is simply the mean similarity μ between all data items in the cluster. That is,

$$\mu = \frac{2}{n(n-1)} \sum_{i,j,i<j}^{n} sim_{ij}$$

where sim is the similarity between sessions i and j computed as a percentage using the selected coefficient, and n is the number of sessions in the cluster. Outliers are excluded (cluster size < 3) to get a meaningful metric of cluster quality. For example, if the metric differences between them are all zero, this gives a mean similarity of 100%. For TURN, we also computed the value of k where the complexity is $O(kn)$ plus the cost of sorting.

ROCK has no defined means of dealing with outliers. We considered all clusters with less than three members as outliers for both algorithms. The tables are computed on a 1000 sessions with a 10% sampling rate for ROCK. Both algorithms scale up though ROCK's sampling rate has to be reduced as the data size increases.

Filters	Cluster Quality (%)	No. of Clusters	No. of Outliers	Complexity
None	86.29	44	574	496
1	83.64	43	582	493
2	93.12	46	550	358
3	99.39	36	122	351

Table 1: Clustering Results for TURN.

Filters	Cluster Quality (%)	No. of Clusters	No. of Outliers
None	48.50	8	489
1	61.09	26	599
2	70.39	32	127
3	79.98	17	127

Table 2: Clustering Results for ROCK using a 40% Threshold.

Filters	Cluster Quality (%)	No. of Clusters	No. of Outliers
None	64.87	12	855
1	90.38	18	768
2	95.79	46	320
3	100	28	180

Table 3: Clustering Results for ROCK using a 70% Threshold.

Since ROCK requires parameters, we selected two reasonable sets representing a looser and tighter cluster definition. As can be seen in Table 3, ROCK's cluster quality was higher with the tighter cluster definition but the results were much poorer when no filters were applied. We also used TURN to suggest parameters to ROCK (Table 4) and while ROCK can be fine tuned to get better results with the right filters and parameters, TURN's parameter choices were more consistent across filters.

Further it is clear that the most consistent results come from the use of TURN as a clustering algorithm in its own right. TURN found more clusters and left fewer outliers than ROCK which could be expected from ROCK's clustering being based only on a sample. Small clusters could easily be missed by a sampling process. TURN also had a somewhat higher cluster quality overall and particularly so when no filters were applied.

Filters	Cluster Quality (%)	No. of Clusters	No. of Outliers	Threshold
None	82.83	31	638	76
1	87.19	22	699	52
2	64.62	33	519	45
3	80.72	18	73	48

Table 4: Clustering Results for ROCK using TURNs Threshold Parameters.

While ROCK runs faster than TURN due to only clustering a sample, TURN avoids the potential errors due to sampling and has substantially lower complexity and memory requirements.

6 Conclusion

In the work reported here we looked at the possibility of non-parametric clustering in a non-Euclidean space as well as issues particularly pertinent to web log data analysis by session and user. While experts may succeed after some trials in optimising clustering results through judicious choice of parameters, the need for the user to specify parameters remains the major difficulty in clustering. This difficulty is particularly acute in web log analysis where the non-Euclidean space makes it difficult to visualise even for experts and users will rarely have data mining expertise. Thus, we have presented the algorithm TURN as the first entirely non-parametric

approach to clustering in data mining. We also investigated the use of various filters and different coefficients of similarity to improve clustering. We compared TURN to another important recently developed algorithm ROCK. ROCK is one of the few clustering algorithms that can be applied in a non-Euclidean space.

TURN allowed for efficient clustering without any request to the user for parameters, the principle drawback of most clustering algorithms. While wavelets [13] have been applied to give non-parametric clustering in certain applications, this has not been applied to metric spaces such as web log analysis as far as we are aware. While WaveCluster does not require normal parameters, the user has to specify a resolution which is a key input parameter, besides the choice of a noise filter.

In our case, while the user has no direct parametric control of the clustering produced by TURN, it can be controlled by the choice of filters all of which have a rationale and can easily be understood by a naïve user. Our interface - WebSiteMiner - also allows the user to compare sessions or users before and after applying the different filters as well as the similarity measures so as to get a feel for the effect they produce.

The interface we developed allows the user to visualize the clustering results in terms of a distribution of frequent items in the clusters both in a grid format and graphically.

We also plan to present TURN as applied to Euclidean spaces and investigate the automatic discovery of boundaries at different levels of granularity or resolution.

References

[1] Ming-Syan Chen, Jong Soo Park, and Philip S. Yu. Efficient data mining for path traversal patterns. *IEEE Transaction on Knowledge and Data Engineering*, 10(2):209–221, March/April 1998.

[2] Yongjian Fu, Kanwalpreet Sandhu, and Ming-Yi Shih. Clustering of web users based on access patterns. In *Workshop on Web Usage Analysis and User Profiling (WEBKDD99)*, August 1999.

[3] Studipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In *15th Int'l Conf. on Data Eng.*, 1999.

[4] J.MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Statist. Prob.*, 1967.

[5] A. Joshi and K. Joshi. On mining web access logs. Technical report, CSEE Department, UMBC, 1999. http://www.csee.umbc.edu/~ajoshi/web-mine/tr1.ps.gz.

[6] Spiliopoulou M. and Pohle C. Data mining for measuring and improving the success of web sites. *Data Mining and Knowledge Discovery, Special Issue on Electronic Commerce*, 2000.

[7] B. Mobasher, R. Cooly, and J. Srivastava. Automatic personalization based on web usage mining. Technical Report TR99-010, Department of Computer Science, Depaul University, 1999.

[8] B. Mobasher, N. Jain, E. Han, and J. Srivastava. Web mining: pattern discovery from world wide web transactions. Technical report, Department of Computer Science, University of Minnesota, 1996.

[9] M.D. Mulvenna, S. S. Anand, and A. G. Büchner. Personalization on the net using web mining. *Communications of the ACM*, 43(8), August 2000.

[10] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 144–155, Santiago, Chile, September 1994.

[11] K. Shim S. Guha, R. Rastogi. CURE: An efficient clustering algorithm for large databases. In *SIGMOD'98*, Seattle, Washington, 1998.

[12] G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1982.

[13] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: a multi-resolution clustering approach for very large spatial databases. In *24th VLDB Conference*, New York, USA, 1998.

[14] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations*, Jan 2000.

[15] Osmar R. Zaïane, Man Xin, and Jiawei Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Proc. Advances in Digital Libraries ADL'98*, pages 19–29, Santa Barbara, CA, USA, April 1998.

[16] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Conf. Management of Data*, pages 103–114, Montrea, Canada, June 1996.