A Compact Representation of Spatio-Temporal Data

Sigal Elnekave, Mark Last
Ben-Gurion University
elnekave@bgu.ac.il, mlast@bgu.ac.il

Oded Maimon
Tel-Aviv University
maimon@eng.tau.ac.il

Abstract

As technology advances we encounter more available data on moving objects, which can be mined to our benefit. In order to efficiently mine this large amount of data we propose an enhanced segmentation algorithm for representing a periodic spatio-temporal trajectory, as a compact set of minimal bounding boxes (MBBs). We also introduce a new, "data-amount-based" similarity measure between mobile trajectories which is compared empirically to an existing similarity measure by clustering spatio-temporal data and evaluating the quality of clusters and the execution times. Finally, we evaluate the values of segmentation thresholds used by the proposed segmentation algorithm through studying the tradeoff between running times and clustering validity as the segmentation resolution increases.

1. Introduction

With technological progress, more data is available on the location of moving objects at different times, either via GPS technologies, mobile computer logs, or wireless communication devices. This creates an appropriate basis for developing efficient new methods for mining moving objects.

Spatio-temporal data can be used for many different purposes. The discovery of patterns in spatio-temporal data, for example, can greatly influence such fields as animal migration analysis, weather forecasting, and mobile marketing. Clustering spatio-temporal data can also help in social network analysis, which is used in tasks like shared data allocation, targeted advertising, and personalization of contents and services.

In this work we build a compact representation of a trajectory by pre-processing a spatio-temporal data stream. Then we define a new data-amount-based similarity measure between trajectories for discovering similar trajectories according to proximity in time and space. This measure will allow the discovery of groups that have similar spatio-temporal behavior. Finally we evaluate the proposed algorithm by conducting experiments on a synthetic data stream. We cluster trajectories built by the proposed algorithm, using our suggested data-amount-based similarity measure compared to minimal distances similarity measure. We also examine different segmentation thresholds and analyze the tradeoff between running times and clustering validity obtained with different segmentation resolutions.

2. Related Work

Representing spatio-temporal data in a concise manner can be done by converting it into a trajectory form. In Hwang et al. [4] a trajectory is a function that maps time to locations. To represent object movement, a trajectory is decomposed into a set of linear functions, one for each disjoint time interval. The derivative of each linear function yields the direction and the speed in the associated time interval. A trajectory is a disjunction of all its linear pieces. For example, a trajectory of the user moving on a 2-D space may consist of the following two linear pieces: $[(x=t-3)\Lambda(y=t+3)\ \Lambda(0<t<2)]U[(x=6)\ \Lambda(y=-t)\Lambda(3<t<5)]$

In Pfoser [6], a linear interpolation is used. The sampled positions then become the endpoints of line segments of polylines and the movement of an object is represented by an entire polyline in 3D space. A trajectory T is a sequence $<(x_1,y_1,t_1),(x_2,y_2,t_2),\ldots,(x_k,y_k,t_k)>$. Objects are assumed to move straight between the observed points with a constant speed. The linear interpolation seems to yield a good tradeoff between flexibility and simplicity.

Anagnostopoulos et al. [1] summarize spatio-temporal data. They propose a distance-based segmentation criterion in an attempt to create minimal bounding rectangles (MBRs) that bound close data points into rectangular intervals in such a way that the original pairwise distances between all trajectories are preserved as much as possible. A variance-based hybrid variation is presented as a compromise between running time and approximation quality.

0-7695-3019-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDMW.2007.79

Several similarity measures are used in the literature. Anagnostopoulos et al. [1] define the distance between two trajectory segmentations at time *t* as the distance between the rectangles at time *t*, and the distance between two segmentations is the sum of the distances between them at every time instant. The distance between the trajectory MBRs is a lower bound of the original distance between the raw data, which is an essential property for guaranteeing correctness of results for most mining tasks.

In D'Auria, et al. [2] the similarity of trajectories along time is computed by analyzing the way the distance between the trajectories varies. More precisely, for each time instant they compare the positions of moving objects at that moment, thus aggregating the set of distance values. The distance between trajectories is computed as the average distance between moving objects.

In Li, et al.[5] the similarity of objects within a moving micro cluster is measured by distance on profiles of objects. Similar objects are expected to have similar initial locations and velocities.

In this work we improve the summarization of spatio-temporal data in a manner that will better fit real time data streams and will also provide tools for improving the efficiency of measuring similarities between trajectories.

3. Specific Methods

3.1. Representing trajectories

A spatio-temporal trajectory is a series of data-points traversed by a given moving object during a specific period of time (e.g., one day). Since we assume that a moving object behaves according to some periodic spatio-temporal pattern, we have to determine the duration of the spatio-temporal sequence (trajectory). Thus, in the experimental part of this paper, we assume that a moving object repeats its trajectories on a daily basis, meaning that each trajectory describes an object movement during one day. In a general case, each object should be examined for its periodic behavior in order to determine the duration of its single trajectory. The training data window is the period which is used to learn the object's periodic behavior based on its recorded trajectories (e.g., daily trajectories recorded during one month).

As a part of our suggested preprocessing technique we represent a trajectory as a list of minimal bounding boxes. A minimal bounding box (MBB) represents an interval bounded by limits of time and location. By using this structure we can summarize close data-points into one MBB, such that instead of recording these original data-points, we only need to record the following six elements:

$$\begin{aligned} i.x_{\min} &= \min(\forall m \in i, m.x_{\min}) \\ i.x_{\max} &= \max(\forall m \in i, m.x_{\max}) \\ i.y_{\min} &= \min(\forall m \in i, m.y_{\min}) \\ i.y_{\max} &= \max(\forall m \in i, m.y_{\max}) \\ i.t_{\min} &= \min(\forall m \in i, m.t_{\min}) \\ i.t_{\max} &= \max(\forall m \in i, m.t_{\max}) \end{aligned} \quad (1)$$

Where *i* represents a MBB, *m* represents a member in a box, *x* and *y* are spatial coordinates, and *t* is time.

We also add other properties to the standard MBB-based representation that improve our ability to perform operations on the summarized data, like measuring similarity between trajectories:

$$i.p = \text{aggregation}(\forall m \in i, m.\text{state}) \quad (2)$$

where *p* stands for the value of a property variable in a minimal bounding box *i*, *m* represents a member in a box, and *state* is the data-point's property that is being aggregated. In our algorithm, *p* represents the number of data points (*data#*) that are summarized by a given MBB:

$$i.data\# = \text{count}(\forall m \in i, 1) \quad (3)$$

A periodic trajectory of an object is identified by an object ID *O* and a date *D* and it can be stored as a list of MBBs: $\{O_1, D_1, [t_1,t_2,x_1,x_2,y_1,y_2,N_1], [t_3,t_4,x_3,x_4,y_3,y_4,N_2]\ldots, [t_{n-1},t_n,x_{n-1},x_n,y_{n-1},y_n,N_n]\}$ where *t* represents time, *x* and *y* represent coordinates (1 for minimal and 2 for maximal), and *N* represents the amount of data points belonging to each MBB. Figure 1 demonstrates an object's trajectory and its MBB-based representation for a given period.

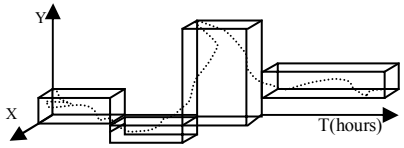

Figure 1. Object's trajectory

Incoming data-points update the MBBs in the order of their arrival times. Therefore, the minimal time bound of the first MBB is the time of the earliest data-point in the dataset and the maximal time bound of this MBB is stretched until the time or the space distance between the maximal and the minimal locations of this MBB reaches some pre-defined segmentation thresholds. When one of these thresholds is exceeded, a new minimal bounding box is created with the time of the subsequent data-point as its minimal time bound. The larger the threshold is, the more summarized trajectories we get, meaning that we increase the efficiency of the next mining stages (shorter running

times for less transactions) but also decrease their precision.

In [3], we have presented an enhanced algorithm for representing an object trajectory as a set of MBBs from a spatio-temporal dataset *D* covering object movement during a pre-defined period (e.g., 24 hours). This algorithm is described below:

```
Input: a spatio-temporal dataset (D), a threshold of x
and y distances and of time duration of a MBB.
Output: new objects' trajectory (T)
Building an object's trajectory:
item← D[1]
T.addMBB(item)    --First item updates first MBB
For each item in D  --Except for first item
  while(|item.X-T.lastMBB.maxX|<XdistThreshold
    and |item.Y-T.lastMBB.maxY|<YdistThreshold
    and | item.T-T.lastMBB.maxT|<durationThreshold)
    T.lastMBB.addPoint(item)--Insert into current MBB
T.addMBB(item) --Create MBB when out of thresholds
```

The algorithm processes each data point in the data stream and inserts a data point into an existing MBB as long as its bounds are within the threshold defined as algorithm parameters; otherwise it creates a new MBB. The "lastMBB" function returns the MBB with the maximal (latest) time bounds in the trajectory, the "addMBB" function initializes a new MBB in the trajectory with bounds and properties updated by the first incoming data-point (in the first arrival, the minimum and the maximum are equal to the data-point values), and the "addPoint" function updates MBB properties (bounds and data amount).

3.2. Defining the thresholds parameters

We will stretch the MBBs in each dimension by a threshold that is set in advance. The larger the threshold is, the more summarized trajectories we get, meaning that we increase efficiency of the next mining stages (shorter running times for less transactions) but also decrease its validity. We will therefore analyze the tradeoff between efficiency and validity for different segmentation thresholds. We set the threshold in each dimension according to the standard deviation and the range of the data values in that dimension. We do so in order to maintain the standard deviation inside the MBBs, and also to maintain the relative size of a MBB in proportion to the data range. As the data standard deviation decreases we need to limit it with stricter bounds. The thresholds for each dimension are set to:

$$bound = stdev(D) \cdot (\max(D) - \min(D)) \cdot b \quad (4)$$

where *D* are the data points of the corresponding dimension, *stdev* retrieves the standard deviation of the data-points, *max* returns the maximal value in that dimension, *min* retrieves the minimal value, and *b* is a user-defined parameter that we try to optimize in the experimental section. The data parameters *stdev*, *max*, and *min* are calculated from the entire dataset *D* before the start of the segmentation procedure. We will evaluate *b* values between 0.05 and 10 that cover different segmentation options within the possible data range.

3.3. Defining a similarity measure

We define similarity between two trajectories as the sum of the similarities between the trajectories' MBBs, divided by the amounts of MBBs in each of the compared trajectories, where the two compared trajectories are described as shown in Figure 2.

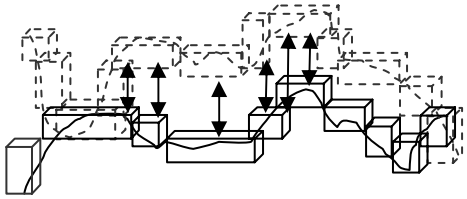

Figure 2. Similarity between two trajectories

In this paper, we will empirically compare two similarity measures between two MBBs. The first similarity measure is called "minimal distances" and it was suggested in [1]. It defines the distance between two segmentations at time *t* as the distance between the rectangles at time *t*, the first segmentation is of trajectory T_i and the other is of trajectory T_j. Formally:

$$d(s(T_i), s(T_j), t) = \min_{\substack{x_i \in P(s(T_i),t) \\ xj \in P(s(Tj),t)}} d(x_i, x_j) \quad (5)$$

The distance between two segmentations is then calculated as the sum of the distances between them at every time instant:

$$d(s(T_i), s(T_j)) = \sum_{t=0}^{m-1} d(s(T_i), s(T_j), t) \quad (6)$$

The distance between the trajectory MBBs is a lower bound of the original distance between the raw data, which is an essential property for guaranteeing correctness of results for most mining tasks. Therefore, if we treat each MBB as a segment we can use the following formula, where t_m is the time when the two MBBs start to overlap and t_n is the time when its overlapping ends:

$$D(MBB(T_i),MBB(T_j))= \min D(MBB(T_i),MBB(T_j)) \cdot |t_m - t_n| \quad (7)$$

The minimal distance and the times t_m and t_n are described in Figure 3.

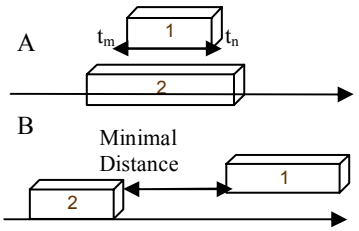

Figure 3: A. Times of overlapping between two MBBs; B. minimal distance between two MBBs

We define the minimal distance between two MBBs are:

$$\min D(MBB(T_i), MBB(T_j)) = X \min D(MBB(T_i), MBB(T_j)) + Y \min D(MBB(T_i), MBB(T_j)) \quad (8)$$

where the minimal distance between two MBBs in X and Y dimensions as follows:

$$X \min D(MBB(T_i), MBB(T_j)) = \max(0, (\max(MBB(T_i).X_{\min}, MBB(T_j).X_{\min}) - \min(MBB(T_i).X_{\max}, MBB(T_j).X_{\max}))) \quad (9)$$

$$Y \min D(MBB(T_i), MBB(T_j)) = \max(0, (\max(MBB(T_i).Y_{\min}, MBB(T_j).Y_{\min}) - \min(MBB(T_i).Y_{\max}, MBB(T_j).Y_{\max}))) \quad (10)$$

Using the enhanced representation of trajectories (see section 3.1 above) we can improve the similarity measure between trajectories as follows. We multiply the minimal-distances measure (6) by the distance between the amounts of data points of the two compared MBBs (data#D). Since each MBB summarizes some data points, the more data points are included in both of the compared MBBs, the stronger support we have for their similarity. Our "data-amount-based" distance is calculated as:

$$d(MBB\ (T_i), MBB\ (T_j)) = \min\ D(MBB\ (T_i), MBB\ (T_j)) \cdot |t_m - t_n| \cdot \text{data\#}\ D(MBB\ (T_i), MBB\ (T_j)) \quad (11)$$

Where the distance between the amounts of data points that are summarized within two MBBS is:

$$\text{data\#}\ D(MBB(T_i), MBB(T_j)) = |MBB(T_i).data\# - MBB(T_j).data\#| \quad (12)$$

3.4. Representing a cluster's centroid

The centroid of a trajectory cluster should represent all items that belong to that cluster in some summarized manner. We represent a cluster centroid as a summarized trajectory (as described in Formulas 1 and 2), or in other words as a vector of MBBs. Each MBB is an interval that holds information about the upper and lower bounds in each one of the d-dimensions (in our case 2-D) of the location, lower and upper time bounds, and the amount of data-points that are summarized within the MBB.

During the clustering phase several similar trajectories are inserted into each cluster. First the centroid of the cluster is initialized to the first inserted trajectory, and after all the trajectories are clustered, the cluster centroids need to be updated.

Instead of the traditional centroid structure of a numeric values vector and its common cluster updating method that calculates a cluster's centroid as a vector of averages of the items in the cluster, we represent clusters as bound intervals, which requires using a bounding technique for updating clusters, since averaging interval bounds will lead to invalid intervals, with bounds that are not the interval's real limits.

We first sort all the MBBs that belong to items in a cluster, where MBBs with larger intervals will appear first (sorted first by the time dimension) in order for smaller MBBs to be inserted into it. Then each MBB is added to the trajectory that represents the cluster's centroid in the following manner: (1) if the inserted MBB is contained within another MBB it will only update the amount of summarized data points in the existing MBB, (2) if the inserted MBB only exceeds an existing MBB with an allowed distance(less than the pre-determined thresholds) it updates the amount of summarized data points in the existing MBB and it also updates the exceeded bounds of the existing MBB. (3) Otherwise, the inserted MBB is added as a new MBB.

3.5. Finding movement patterns

A trajectories cluster contains similar trajectories. Trajectories in the same cluster contain as much similar MBBs as possible (close in space and time). The centroid of a trajectories cluster represents a group of similar trajectories, meaning that this cluster's centroid can represent a movement pattern.

Since in order to run generic clustering algorithms on the trajectories data, the algorithm needs to handle an input that consists of bound intervals (trajectories) instead of numeric values vectors, we developed in [3] a spatio-temporal version of the K-Means algorithm for clustering trajectories using the similarity measures defined earlier. This version handles interval-bounded data represented by a variable amount of attributes. It uses the data-amount-based similarity measure and a new centroid structure and updating method that were defined earlier.

Each trajectory is represented by an *id* and a set of MBBs (as described below). This algorithm receives as input a set of trajectories of the form: $\{T_1, [t_1,t_2,x_1,x_2,y_1,y_2,N_1], [t_3,t_4,x_3,x_4,y_3,y_4,N_2]..\ ,[t_{n-1},t_n,x_{n-1},x_n,y_{n-1},y_n,N_n]\}$ $\{T_2,[t_1,t_2,x_1,x_2,y_1,y_2,N_1], [t_3,t_4,x_3,x_4,y_3,y_4,N_2]..,\ [t_{n-1},t_n,x_{n-1},x_n,y_{n-1},y_n,N_n]\}..$ and it outputs new object clusters of the form: $\{[c_1,(T_1,T_2,T_9),$ trajectory centroid of c_1 containing three trajectories T_1,T_2,and $T_9]$, $[c_1,(T_3,T_7,T_8),$ trajectory centroid of $c_2]\ldots)\}$.

3.6. Using incremental clustering approach

We adapt the incremental approach in order to benefit from the difference between the clustering for the first training data window (e.g. trajectories during the first month of data collection), when no previous data is available, and clustering for the subsequent windows, where using previous clustering centroids can help performing a more efficient incremental clustering process, since less updates are needed assuming that the movement behavior of the same object stays relatively stable. With the non-incremental approach, the clustering algorithm is applied repeatedly to each new data window (e.g., on a monthly basis) without utilizing clustering results of earlier data windows (the centroids are re-calculated for each new window).

An algorithm for incrementally discovering periodic movement patterns during a given data window includes the initialization of cluster centroids according to the previous cluster results of the same objects, where early clustering results exist. In our previous work [3] we performed experiments that showed that the incremental approach decreases clustering running times and increases cluster's validity.

4. Experimental Results

4.1. Generation of spatio-temporal data

In the absence of real spatio-temporal datasets, mainly due to privacy issues, we have generated synthetic data for our experiments.

This synthetic data was used for the empirical evaluation of the proposed algorithm for incremental representation of trajectories with the new "data-amount-based" similarity measure. We ran 20 simulations each representing a moving object. The first 10 runs simulated daily movements (trajectories) of a mobile object during 25 days, and the other 10 runs simulated daily movements (trajectories) of a mobile object during 45 days. These trajectories belonged to five movement patterns. Trajectories that belong to the same movement pattern reached at least three identical locations at identical times. The location of each object was sampled at least 35 times during each day.

4.2. Detailed Results

We preprocessed the data using our suggested algorithm for building trajectories. In order to evaluate the proposed similarity measure, we clustered the trajectories once using the "minimal distances" similarity measure [1] and once using our proposed "data-amounts-based" similarity measure.

For each simulation K-Means iterations amount was set to 20, and the we examined the following amounts of clusters k: 2, 3, 4, 5, 6, 7. For each value of k, we examined 6 different options for the segmentation thresholds (using the following values of b substituted in equation (4): 1, 1.5, 2, 2.5, 3, 3.5). We compared the clustering running times. We also compared clusering Dunn index, which measures the worst-case compactness and separation of a clustering, with higher values being better [7]

$$D_I = \frac{D\min}{D\max} \quad (13)$$

D_{min} is the minimum distance between any two objects in different clusters (separation) and D_{max} is the maximum distance between any two items in the same cluster (homogeneity).

After the experiments, we evaluated the results using a oneway analysis of variance (ANOVA), for the independent variable: similarity measure type (minimal-distance or data-amounts-based). Our dependent variables (tested separately) are Dunn index, and running time. The ANOVA results show that our suggested "data-amounts-based" similarity measure significantly outperforms "minimal distances" similarity measure when analyzed according to Dunn index, as can be seen in Figure 4.

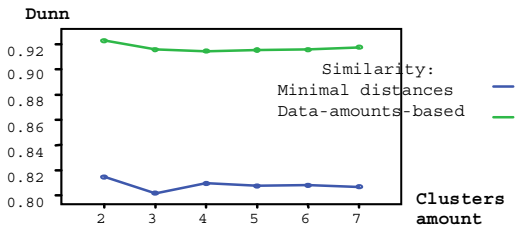

Figure 4. Dunn index vs. different cluster amounts

We summarized the trajectories with different segmentation thresholds calculated by Formula 4, and ran our clustering algorithm on the summarized trajectories. As one can see in Figure 5, when b increases, run durations decrease, but so is the validity according to the Dunn index. When b=4 we get about 90% of the validity for 3% of the running time. While b=2 gives a nice increase of accuracy to 98% for about 8% of running time, b=1 gives a less significant increase of accuracy to 99.9% for about 17% of run time. b=0.2 gives 100% accuracy for about 44% of running time.

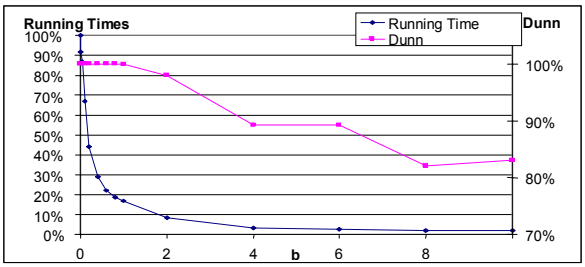

Figure 5. Running time and Dunn index vs. *b* values with summarized trajectories

We then summarized the trajectories with different thresholds, ran our clustering algorithm on the summarized trajectories, while the centroids were also summarized using the same thresholds as a part of their updating procedure. As one can see in Figure 6, when *b* increases, run durations decrease much lower when summarizing centroids, but so is the validity according to the Dunn index. Thus, a lower segmentation resolution is needed (lower *b*) in order to get more validity for less computation time. While *b*=0.8 gives a nice increase of accuracy to 96% for 2% of running time when running the clustering without segmentation of trajectories, *b*=0.2 gives a less significant increase of accuracy to 99.9% for 4% of run time. *b*=0.1 gives 100% accuracy for about 5.7% of running time.

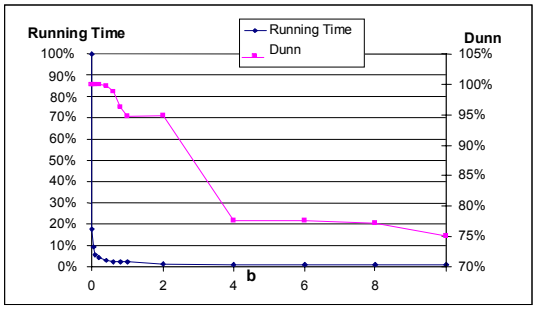

Figure 6. Running time and Dunn index vs. b values with summarized trajectories and centroids

5. Conclusions

In this paper, we improved our suggested method for clustering summarized trajectories into movement patterns. We presented a new way for summarizing spatio-temporal data, including a new similarity measure between summarized trajectories that outperforms existing similarity measures according to clusters validity index.

Analyzing the validity-efficiency tradeoff for different segmentation resolutions demonstrates that it is best to choose *b*=2 when we are indifferent between efficiency and validity, since running times stay low and accuracy is relatively high. It also shows that adding segmentation of centroids during their updating stage, significantly decreases running times (6% running time for 100% validity when *b*=0.1).

Further work is needed for finding the best segmentation resolution for applications where efficiency and validity have different importance weights. Experimentation with massive streams of complex spatio-temporal data is another important research task.

6. Bibliography

[1] A. Anagnostopoulos, M. Vlachos , M. Hadjieleftheriou, E Keogh., P.s. Yu, "Global Distance-Based Segmentation of Trajectories". *KDD'06*, Philadelphia, Pennsylvania, USA, August 20–23, 2006.

[2] M. D'Auria, M. Nanni D., Pedreschi "Time-focused density-based clustering of trajectories of moving objects" To appear in JIIS Special Issue on "*Mining Spatio-Temporal Data"*, 2006

[3] S. Elnekave, M. Last, O. Maimon "Incremental Clustering of Mobile Objects*",STDM07,* IEEE , 2007

[4] S.Y. Hwang, Y.H. Liu, J.K. Chiu, F.P. Lim (2005) "Mining Mobile Group Patterns: A Trajectory-based Approach". *Lecture Notes in Artificial Intelligence, PAKDD* 2005, v. 3518, , p 713-718

[5] Y. Li, J. Han, J. Yang, "Clustering Moving Objects". *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, p 617-622.

[6] D. Pfoser, "Indexing the Trajectories of Moving Objects". *IEEE Data Engineering Bulletin, 2002*

[7] A. Schenker, H. Bunke, M. Last, A. Kandel, "Graph-Theoretic Techniques for Web content Mining", World Scientific, *Series in Machine Perception and Artificial Intelligence*, 2005, Vol. 62