University of Texas Rio Grande Valley

ScholarWorks @ UTRGV

Theses and Dissertations - UTB/UTPA

5-2013

Document Collection Visualization and Clustering Using An Atom Metaphor for Display and Interaction

Khanh V. Nghi
University of Texas-Pan American

Follow this and additional works at: https://scholarworks.utrgv.edu/leg_etd

Part of the Computer Sciences Commons

Recommended Citation
Nghi, Khanh V., "Document Collection Visualization and Clustering Using An Atom Metaphor for Display and Interaction" (2013). *Theses and Dissertations - UTB/UTPA*. 727.
https://scholarworks.utrgv.edu/leg_etd/727

This Thesis is brought to you for free and open access by ScholarWorks @ UTRGV. It has been accepted for inclusion in Theses and Dissertations - UTB/UTPA by an authorized administrator of ScholarWorks @ UTRGV. For more information, please contact justin.white@utrgv.edu, william.flores01@utrgv.edu.

DOCUMENT COLLECTION VISUALIZATION AND CLUSTERING USING AN ATOM METAPHOR FOR DISPLAY AND INTERACTION

A Thesis

by

KHANH V. NGHI

Submitted to the Graduate School of the
University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2013

Major Subject: Computer Science

DOCUMENT COLLECTION VISUALIZATION AND CLUSTERING USING AN ATOM METAPHOR FOR DISPLAY AND INTERACTION

A Thesis
by
KHANH V. NGHI

COMMITTEE MEMBERS

Dr. Richard H. Fowler
Chair of Committee

Dr. Wendy A. Lawrence-Fowler
Committee Member

Dr. Zhixiang Chen
Committee Member

May 2013

Copyright 2013 Khanh V. Nghi

All Rights Reserved

ABSTRACT

Nghi, Khanh V., Document Collection Visualization and Clustering using an Atom Metaphor for Display and Interaction. Master of Science (MS), May, 2013, 76 pp., 1 table, 54 figures, references, 66 titles.

Visual Data Mining have proven to be of high value in exploratory data analysis and data mining because it provides an intuitive feedback on data analysis and support decision-making activities. Several visualization techniques have been developed for cluster discovery such as Grand Tour, HD-Eye, Star Coordinates, etc. They are very useful tool which are visualized in 2D or 3D; however, they have not simple for users who are not trained.

This thesis proposes a new approach to build a 3D clustering visualization system for document clustering by using k-mean algorithm. A cluster will be represented by a neutron (centroid) and electrons (documents) which will keep a distance with neutron by force.

Our approach employs quantified domain knowledge and explorative observation as prediction to map high dimensional data onto 3D space for revealing the relationship among documents. User can perform an intuitive visual assessment of the consistency of the cluster structure.

DEDICATION

The completion of my master degree would not have been possible without the love and support of my family. My Dad and Mom – Nghi Vinh Buu and Huynh Thi Tuyet Huong, my wife – Cao Thi Hong Cam, and two my daughters, wholeheartedly inspired, motivated and supported me by all means to accomplish this degree. Thank you for my love and patience.

ACKNOWLEDGMENTS

I will always be grateful to Dr. Richard H. Fowler, chair of my thesis committee, for all his mentoring and advice. From database funding, research design, and data processing, to manuscript editing, he encouraged me to complete this process through his infinite patience and guidance. My thanks go to my thesis committee members: Dr. Wendy A. Lawrence-Fowler, and Dr. Zhixiang Chen. Their advice, input, and comments on my thesis helped to ensure the quality of my intellectual work.

I would also like to thank my colleagues at the Tra Vinh University who give me motivations for my study. Also, I would like to acknowledge the Travinh-100 Project's Executive Board who supports full funding for my study abroad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Clustering is a very important technique for discovering human's knowledge. This technique helps us to separate objects from a dataset into groups –called clusters- based on the similarity of object in the group. There are a lot of cluster algorithms which is develop to deal with very large-scale and complex datasets has been widely used in a lot of areas such as data mining, image processing, pattern recognition, statistics, bio-information, and other field. However, we have not a general-purpose clustering algorithm so that we can apply it to all kind of application. Therefore, the evaluation of the quality is very necessary for cluster analysis. This can be supported by visualization. As we known, visual presentations are very useful in exposing trends of datasets, giving understanding overview/details or highlighting outliers. Clearly, cluster analysis use visualization to map the high-dimension data to a two or three-dimension space. As result, we have an intuitive image to easily understand relationship between object in the datasets.

Many techniques which worked on high dimensions data visualization were proposed. However, some of them deal with problem in very high dimensional and large datasets, especially in data mining. In other words, the impreciseness of visualization limits its utilization in quantitative verification and validation of cluster results. Therefore, motivation of this research is that developing a visualization technique with the features of purposeful cluster detection and precise contrast between clustering results.

We propose a visual 3D graph layout called an atomic structure model for document cluster. This model use SOM algorithm (3D or 2D) to estimate number of cluster which will be separated from a set of documents – PDF format. Then, Based on vector space model, in particular tf-idf vector, we use k-means/k-means++ or 1D-SOM to cluster the datasets into k groups. Then we visualize each cluster as an atom structure. In this model, neutrons are centroids of clusters and these documents are represented by electrons which rotate around its neutron. Electrons which belong to its atom are arranged on 5 levels by using Euclidian distance between electrons and its neutron – called energy level.

There are some advantages from our new approach:

First, the model shows quantitatively data distribution in 3D environment. It means that we can see how many documents in clusters, how far from the centroid, and so on. In this model, it also illustrates relationship among centroids of clusters by using force-directed algorithm and our algorithm based on their distances. Moreover, our approach can overcome the limit of space in compare with previous 2D approach by using arcball technique for interaction.

Second, our approach avoids arbitrary selection of k clusters by combination from SOM and K-mean. As a consequence, this model provides users a purposeful and effective visual method on cluster analysis.

This thesis is arranged as follow: Chapter 2 gives an introduction to clustering, clustering algorithms and visual cluster analysis. Chapter 3 presents related word on high dimension data and visual technique that have been used in cluster analysis; then we reviews results and evaluation in our approach. Finally, Chapter 4 summarizes the work in the thesis and discussion.

CHAPTER II

REVIEW OF LITERATURE

In this chapter, we review some clustering algorithm and visual cluster analysis as the background of this thesis. First, we introduce clustering algorithm with their advantages/disadvantages. Then we present some visualization technique which allow data miner to monitor, manipulate, evaluate and support decision-making activities on dataset. This is a combination of information visualization and cluster analysis technique.

Clustering Algorithms

Clustering aims at partitioning objects into groups according to a certain criteria. Indeed, the clustering problem is to partition a dataset into groups (clusters) so that the data elements within a cluster are more similar to each other than data elements in different clusters by given criteria. In general, clustering is considered as an unsupervised classification process [1]. A large number of clustering algorithms have been developed to achieve different kind of application [2]. However, each algorithm is only suitable for some application; it means that there are no general-purpose clustering algorithms, it need an evaluation mechanism to assess the quality of clustering results that produced by different clustering algorithms, so that the user may apply an appropriate algorithm for a specific application.

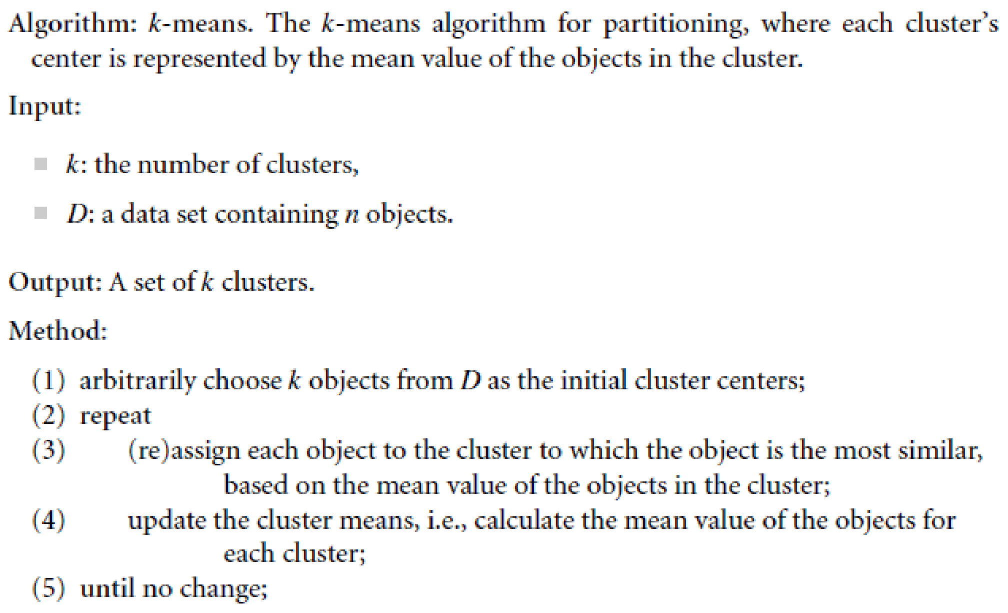
Clustering technique can be divided in two classes: hierarchical clustering techniques and partitioning clustering techniques [2]. However, there are some combinations of different clustering algorithms (both classes) as optimal approach for specific application.

Beside the two traditional hierarchical and partitioning classes, several clustering techniques that are categorized into independent classes such as density-based methods, Grid-based methods and Model based clustering methods [2]. A short review of these methods is described below.

Partitioning methods

Partitioning clustering algorithms, such as K-means [3], K-medoids PAM [4] assign objects into k clusters (we need to give k), and iteratively reallocate objects to improve the quality of clustering results. A useful tool for determining *k* is the silhouette [5].

Over 50 years from K-means proposed, it is probably the most well-known clustering algorithm. One of the important concepts in K-means is that of centroid. Each cluster has a centroid. We can consider it as the point that is most representative of the cluster. Equivalently, centroid is point that is the "center" of a cluster. However, K-means algorithm is very sensitive to estimate K number for the initial centroids; it means that the clustering result will be so different because it depends on the different centroids.


Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

```
(1) arbitrarily choose $k$ objects from $D$ as the initial cluster centers;
(2) repeat
(3)       (re)assign each object to the cluster to which the object is the most similar,
                based on the mean value of the objects in the cluster;
(4)       update the cluster means, i.e., calculate the mean value of the objects for
                each cluster;
(5) until no change;
```

Figure 1: The k-means partitioning algorithm [6].

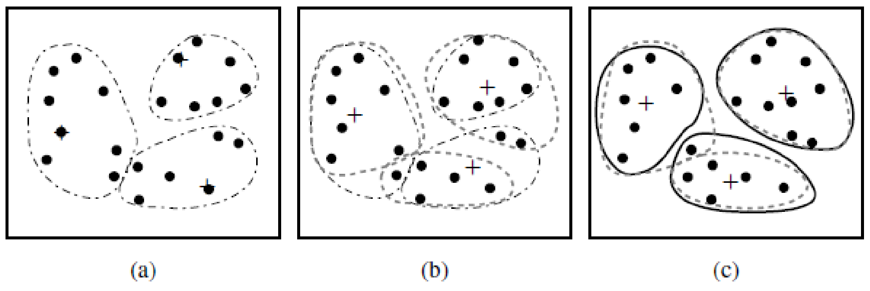
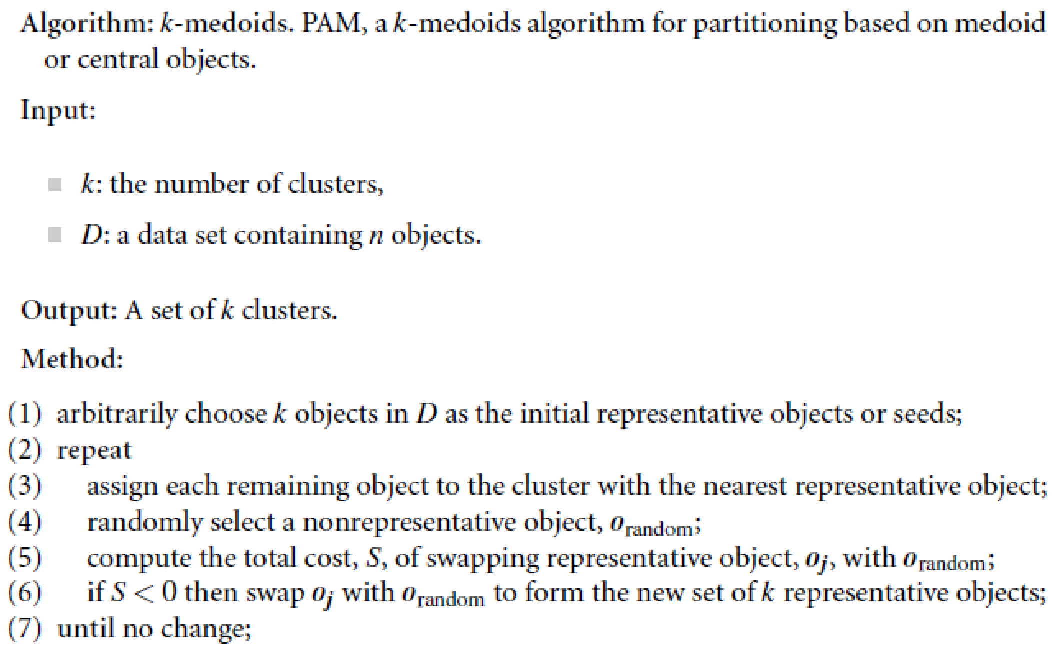


Figure 2: An example of K-means methods (The means of each cluster is marked by a "+") [6].

K-medoids - a variation of K-means- calculates the medoids of the objects in each cluster, rather using the mean value of data objects in a cluster as the center of the cluster. IN general, processes of K-medoids algorithm are quite similar K-means. However, K-medoids clustering algorithm is very sensitive to outliers whom could seriously influences clustering results. The most common realization of *k*-medoids clustering is the Partitioning Around Medoids (PAM) algorithm which was proposed by Kaufman and Rousseeuw [4]. PAM is more robust than k-means in terms of handling noise and outliers because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances. Time complexity is

O($k(n-k)^2$) for each iteration of swap (where k is the cluster number, n is the items of the data set); thus PAM only performs well on small-sized datasets.



Algorithm: k-medoids. PAM, a k-medoids algorithm for partitioning based on medoid or central objects.

Input:

- k: the number of clusters,
- D: a data set containing n objects.

Output: A set of k clusters.

Method:

```
(1) arbitrarily choose $k$ objects in $D$ as the initial representative objects or seeds;
(2) repeat
(3)     assign each remaining object to the cluster with the nearest representative object;
(4)     randomly select a nonrepresentative object, $o_{random}$;
(5)     compute the total cost, $S$, of swapping representative object, $o_j$, with $o_{random}$;
(6)     if $S < 0$ then swap $o_j$ with $o_{random}$ to form the new set of $k$ representative objects;
(7) until no change;
```

Figure 3: PAM, a k-medoids partitioning algorithm [6].

Hierarchical methods

Hierarchical clustering algorithms assign objects in tree-structured clusters, i.e., a cluster can have data points or representatives of low level clusters. According to the hierarchical decomposition is formed in a bottom up (merging) or top-down (splitting), hierarchical clustering algorithms can be classified into categories: agglomerative and divisive[6].

Agglomerative hierarchical clustering: start with each object in a separate cluster then merges these single clusters into larger cluster, ends up with a single cluster which contains all objects.

Divisive hierarchical clustering: it is reverse from agglomerative hierarchical clustering, start with all objects in a single cluster then subdivides the cluster into smaller clusters until one end up with clusters that contain individual units.

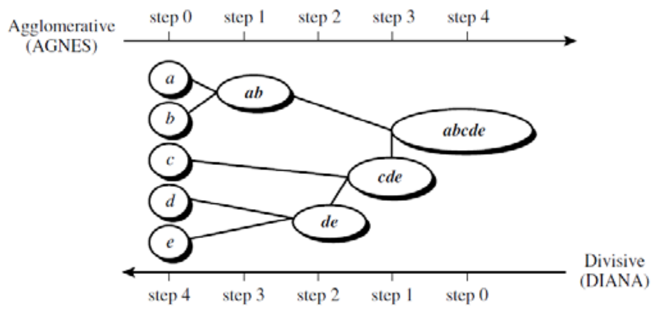


Figure 4: Hierarchical Clustering Process on data objects {a,b,c,d,e} [6]

In the hierarchical clustering method is deal with difficult problem for selection of merge or split points. Thus, the merging and splitting decisions are critical because once a group of objects is merged or split, the next step will work on new generated clusters. Practically, hierarchical clustering method is not suitable for very large datasets because its time complexity is totally $O(n^2)$.

BIRCH (Balanced and Iterative Reducing and Clustering using Hierarchies) [7, 8] is designed to overcome the two difficulties of agglomerative clustering method: scalability and inability to undo what was done in the previous step. BIRCH summarizes cluster representation (an entire dataset) into a CF-tree and then runs a hierarchical clustering algorithm on a multi-level compression technique, CF-tree, to get the clustering result. A CF tree is a height-balanced tree that contains clustering feature (CF = <n, LS, SS>; where n is the number of points in the cluster, LS is the linear sum of the n points, and SS is the square sum of data points) for a hierarchical clustering. Computation complexity of BIRCH is O(n) where n is the number of object in datasets. The linear scalability of algorithm is good at clustering with a single scan and its quality. It is an efficient clustering method on cluster with spherical in shape since it uses the notion of radius to control the boundary of a cluster. BIRCH is not always corresponding to what data miners may consider a natural cluster because each node in a CF tree can hold only a limited number of entries due to its size [6].

CURE which is hierarchical clustering algorithm is more robust to outliers, and identifies cluster having non-spherical shape and wide variances in size [9]. The worse-case time complexity of CURE is $O(n^2 \log n)$. Thus, the number of representative points increases dramatically in order to maintain the precision.

CHAMELEON [10] uses a k-nearest-neighbor graph to construct a sparse graph, where each vertex represents a data object, and if one object is among the k-most-similar objects of the other, there will exist an edge between two objects. This method may produce better results than CURE on complex cluster shapes for spatial datasets. But the high complexity of the algorithm prevents its application on higher dimensional datasets.

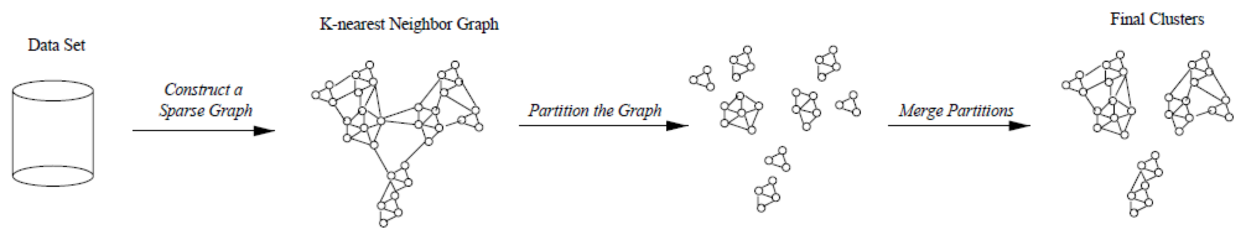


Figure 5: Overall framework CHAMELEON [10].

Density-based methods

Density-based clustering method has been developed to discover cluster with arbitrary shape. The method has played an important role in finding non-linear shapes structure based on the density. The main idea of density-based methods is that for each point of a cluster the neighborhood of a given unit distance contains at least a minimum number of points, i.e. the density in the neighborhood should reach some threshold [11]. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is most widely used density based algorithm. It uses the concept of density reachability and density connectivity.

- Density Reachability: - A point "p" is said to be density reachable from a point "q" if point "p" is within ε distance from point "q" and "q" has sufficient number of points in its neighbors which are within distance ε.
- Density Connectivity: - A point "p" and "q" are said to be density connected if there exist a point "r" which has sufficient number of points in its neighbors and both the points "p" and "q" are within the ε distance. This is chaining process. So, if "q" is neighbor of "r", "r" is neighbor of "s", "s" is neighbor of "t" which in turn is neighbor of "p" implies that "q" is neighbor of "p".

DBSCAN searches for clusters by checking the e-neighborhood of each point in the database. If the e-neighborhood of a point p contains more than MinPts (minimum number of points), a new cluster with p as a core object is created. DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters. The process terminates when no new point can be added to any cluster [6].

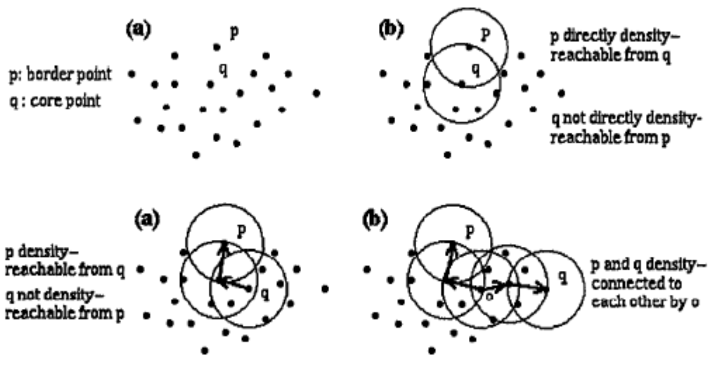

(a)
p
q
p: border point
q : core point
(b)
p
q
p directly density-reachable from q
q not directly density-reachable from p
(a)
p
q
p density-reachable from q
q not density-reachable from p
(b)
p
o
q
p and q density-connected to each other by o

Figure 6: Density reachability and density connectivity in density-based clustering [11]

The runtime of the algorithm is of the order O(n log n) - where n is the number of database objects - if region queries are efficiently supported by spatial index structures, i.e. at least in moderately dimensional spaces; otherwise, it is O(n^2). Besides DBSCAN algorithm is able to find arbitrarily size and arbitrarily shaped clusters, DBSCAN has some advantages such it does not require a-priori specification of number of clusters and easy to identify noise data while clustering. However, DBSCAN algorithm fails in case of varying density clusters or neck type of dataset. It is sensitive to selecting parameter values that will lead to discovery of acceptable clusters (such unit distance or radius and threshold density that are estimated by users).

OPTICS (Ordering Points to Identify the Clustering Structure) as proposed as extension of DBSCAN to produce a cluster ordering obtained from a wide of parameter setting [12].

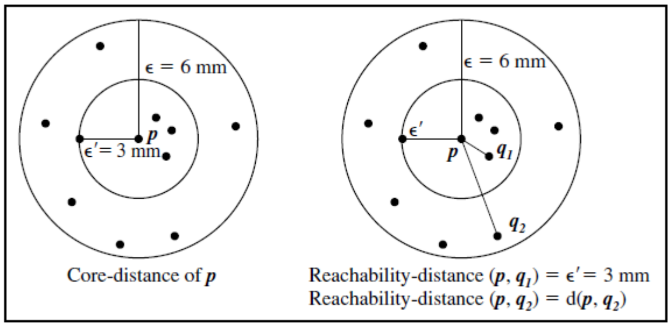

ϵ = 6 mm
ϵ'= 3 mm
p
Core-distance of p
ϵ = 6 mm
ϵ'
p
q_1
q_2
Reachability-distance (p, q_1) = ϵ' = 3 mm
Reachability-distance (p, q_2) = d(p, q_2)

Figure 7: OPTICS terminology [12]

OPTICS and DBSCAN have structural equivalence; thus runtime complexity is O(n log n) - where n is the number of database objects.

Grid-based methods

The grid-base clustering approach is based on the clustering-oriented query answering in multilevel grid data structures. The upper level stores the summary of the information of its next level, thus the grids make cells between the connected levels.

Some grid-based methods include STING (Statistical Information Grid Approach) [13] which a statistical information grid-based approach to spatial data mining. This method divides the spatial area into rectangular cells. These cells form a hierarchical structure in which each cell at a high level is partitioned to form a number of cells at the next lower level.

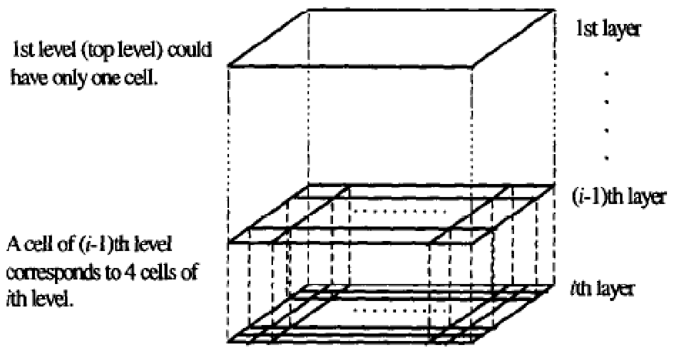

1st level (top level) could have only one cell.
A cell of (i-1)th level corresponds to 4 cells of ith level.
1st layer
(i-1)th layer
ith layer

Figure 8: A hierarchical structure for STING clustering method [13]

STING has some advantages such as: the grid-based computation is query-independent; the grid structure facilitates parallel processing.

WaveCluster [14], which clusters object using a wavelet transform method. It can handle efficiently large data sets, is insensitive to the order of input. WaveCluster was better BIRCH, DBSCAN in terms of efficiency and clustering quality [6].

CLIQUE [15] represents a grid and density-based approach clustering in high-dimensional data space. CLIQUE automatically finds subspaces of the highest dimensionality such that high-density clusters exist in those subspaces. Like WaveCluster, the approach is insensitive to the order of input objects.

The runtime complexity of the grid-based methods are O(N), thus they are efficient on clustering. However, to apply grid-based techniques, data miners must decide the size of grids which quite depends on the user's experience.

Model-based clustering methods

Model-based clustering methods are often based on the assumption that data are generated by a mixture of underlying probability distributions. These methods try to optimize the fit between the data and some mathematical model, for example Expectation-Maximization, Conceptual clustering, and neural network approach.

COBWEB [16] is a popular and simple method of conceptual clustering system that organizes data to maximize inference abilities. It does this by capturing attribute inter-correlations at classification tree nodes and generating inferences as a by-product of classification, then creates a hierarchical clustering in the form of a classification tree. However, the classification tree is not height-balanced of skewed input data which may affect to time and space complexity of this method.

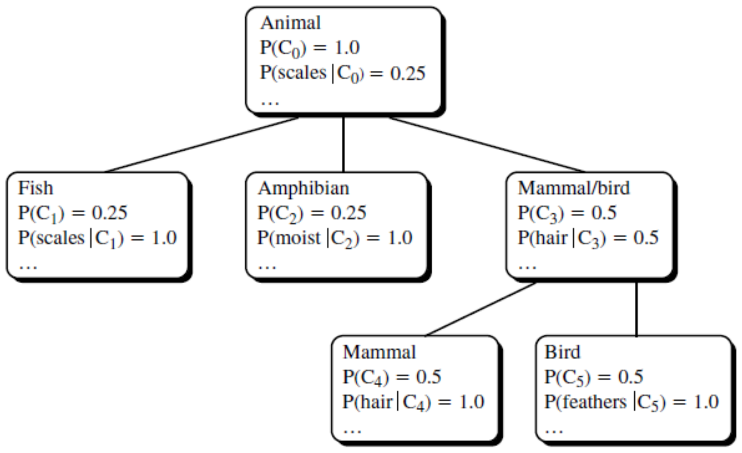

Animal
$P(C_0) = 1.0$
$P(\text{scales} | C_0) = 0.25$
…
Fish
$P(C_1) = 0.25$
$P(\text{scales} | C_1) = 1.0$
…
Amphibian
$P(C_2) = 0.25$
$P(\text{moist} | C_2) = 1.0$
…
Mammal/bird
$P(C_3) = 0.5$
$P(\text{hair} | C_3) = 0.5$
…
Mammal
$P(C_4) = 0.5$
$P(\text{hair} | C_4) = 1.0$
…
Bird
$P(C_5) = 0.5$
$P(\text{feathers} | C_5) = 1.0$
…

Figure 9: A classification tree, based on [16]

The neural network which is motivated by biological neural network approach to clustering tends to represent each cluster as an exemplar. An exemplar operates as “prototype” of the cluster. New object can be assigned to a cluster (exemplar) based on distance measure. Self-organizing feature maps (SOMs) are one of the most popular neural network methods for clustering analysis. We will discuss SOMs in next chapter.

The issues of cluster analysis

We reviewed some classes of clustering algorithms such partitioning method, hierarchical method, density-based method, grid-based method, and model-based clustering method. According to this survey, it is clear that there is no general purpose clustering algorithms. Besides many advantages, we can recognize that there are some drawbacks that influence the feasibility of cluster analysis in data mining. Indeed, most clustering algorithms deal with arbitrarily shaped data distribution of datasets. Moreover, very high dimensional and large datasets will influence on computational complexity of these methods.

An issue which most of the existing clustering algorithms tend to deal with the entire clustering process automatically is sets the parameters of algorithms; the clustering results depend on parameters in some algorithms.

To overcome some of above problem, combination of clustering algorithms and information visualization techniques have proven to be of high value in exploratory data analysis and data mining [17]. A detailed review of visualization techniques used in cluster analysis is presented in the next section.

Visual cluster analysis

As described in the above section, most of the existing clustering algorithms deal with some problems such as arbitrarily shaped cluster distributions and very large and multidimensional datasets. These drawback need to been overcome so that we can use apply efficiently clustering algorithms in data mining.

Thank to information visualization which is the study of interactive visual representations of abstract data to reinforce human cognition, we can combine visualization technique and clustering algorithm – called visual cluster analysis - to mitigate the above problems. One of the greatest benefits of visualization is the sheer quantity of information that can be rapidly interpreted if it is present well [24].

As Card et al [18] described, visualization is considered as one of the most intuitive methods for cluster detection and validation, especially performing well on the representation of irregularly shaped clusters. Based on powerful feature, visualization provides analysts with intuitive feedback on data distribution and support decision-making activities; it can be very powerful in revealing trends, highlighting outliers, showing clusters, and exposing gaps in data [17] in the cluster analysis process.

Using visualization techniques to explore and understand high-dimensional data is an efficient way to combine human intelligence with the immense brute force computation power available nowadays [19].

There are four basic stages of the process of data visualization [24]:

- The collection and storage of data itself
- The preprocessing designed to transform the data into something we can understand
- The display hardware and the graphics algorithms that produce an image on the screen
- The human perceptual and cognitive system (the perceiver)

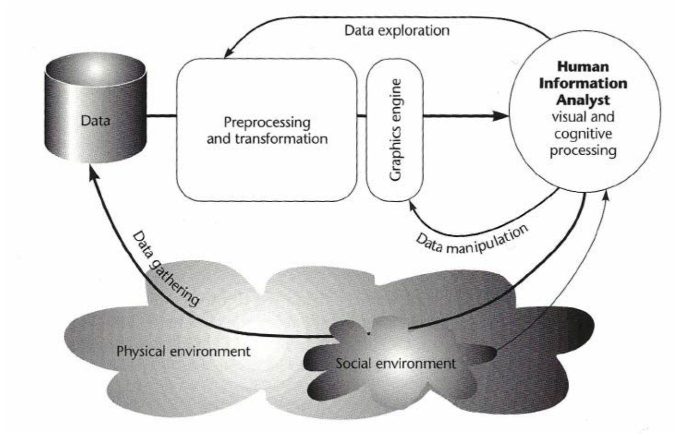


Figure 10: A schematic diagram of the visualization process [24]

A large number of visualization techniques have been developed to map multidimensional datasets to two or three dimensional space [20, 21, 12, 22, 23, 17, 19]. They have been developed to study the cluster structure of data, i.e., the existence of distinctive groups in the data and how these clusters are related to each other. In practice, most of them simply take information visualization as a layout problem; therefore, they are not suitable to visualize clusters of very large datasets.

Multidimensional Data Visualization

In this section, we'll take a closer look at the process of generating a visual representation from a certain number of data, using specific computer processes. Indeed, we will present some common techniques for visualizing multidimensional data structures. However, most of those visual approaches still have difficulty in dealing with high dimensional and very large datasets.

Geometric Techniques. Geometric techniques in information visualization consist of mapping the data of the attributes on a geometric space and projections of the data to produce useful and insightful visualizations. The typical systems used the geometric techniques are

Parallel Coordinates [26], Scatterplot-Matrices [27]. Here we introduce several of them as follows.

Parallel Coordinates is a famous multidimensional visualization technique which each the attributes corresponds to an axis and the axes are arranged to be parallel and equally spaced. A polygonal chain which represents each record of dataset connects the value of the attributes on its axes. This technique is very useful tool for explorative analysis of data. Indeed, the intersection of the line that join the value between the two axes at that point correspond to value of consider dimension, as present in below picture.

	Age	Weight	Sex
Vincenzo	32	75	M
Piero	24	63	M
Luisa	28	60	F
Giulia	18	58	F

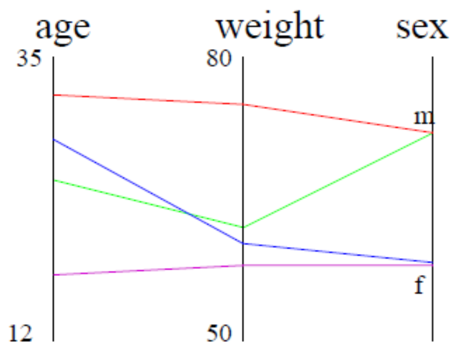


Figure 11: Dataset (left) and its representation (right) by Parallel coordinates [25]

Although this method is very powerful, it will deal with problems when datasets is very large (over 5,000 element). In fact, the representation really depends on the space available on the screen.

Scatterplot-Matrices are a Plot-based data visualization approaches. These methods represent pairs of attributes, through bi-dimensional scatterplots, and put the scatterplots side by side to share the same axes. In other word, they visualize data in rows and columns of cells which contain simpler graphical illustration. However, using these methods does not provide the best overview of the whole dataset. Moreover, the scatterplot matrices can deal with inconveniences with large number of dimensional datasets. Indeed, it needs an NxN scatterplot matrix for N attributes; thus the available space which represents the objects is very limited.

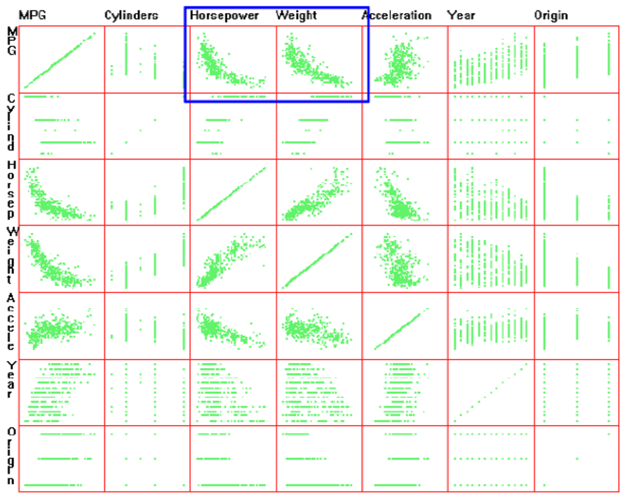


Figure 12: Scatterplot matrices present multivariate data very simply and intuitively. Image generated by Xmdvtool software [25].

As far as we concerned, most Geometric techniques – with represented by Parallel Coordinates and Scatterplot matrix can provide visual presentations of multidimensional attributes. However, they are sensitive to give the user a clear overall insight of data distribution when the dataset is huge because of limitation of space in screen.

Icon-based techniques. Icon-based technique is another family of older techniques for visualizing of multidimensional data which uses the geometric properties of a figure. An icon (called glyph) can have some features such as shape, color, size, orientation, etc. The idea of icon-based techniques is to map each multidimensional data item as an icon. We explain several popular techniques involving of Start Plots [28], Chernoff Faces [29], and Stick Figures [30].

Start Plots is method that represents a simple and relatively intuitive geometric figure by a star-shaped polygon whose vertices are defined by a collection of axes that all the same origin [25].

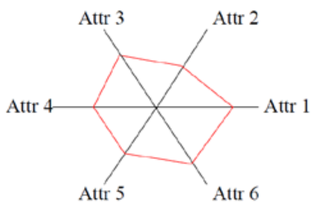


Figure 13: Star plot [25]

A star will represents for a record in dataset and the value of each attribute of the record are mapped to the length of each vertex. A geometric figure (shape which describes the record) is formed by joining the vertices. This technique can be useful for comparing objects in dataset from the polygonal shapes derived from each icon. However, Scalability is a drawback of this approach. Indeed, limitation of space on screen leads to be difficult to clearly make out the various icons because it is so dense. Moreover, with a very high dimensional data, it is so hard to get qualitative and quantitative comparisons among the icons in screen.

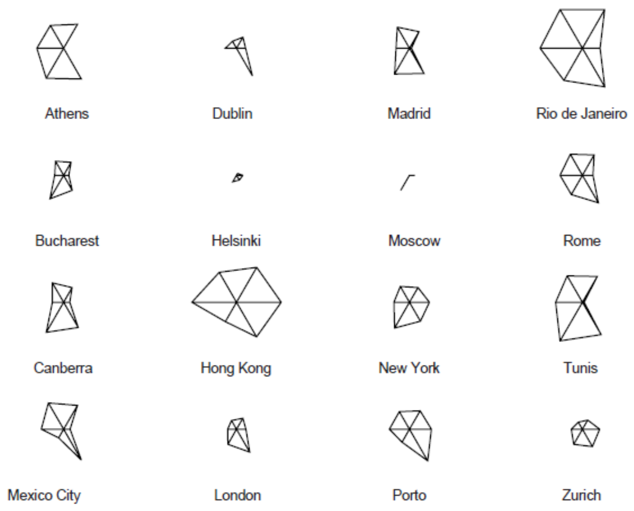


Figure 14: Annual climatic value in Celsius of some world cities with 6 attributes of dataset. Image generated by the S-PLUS tool [25]

Chernoff Faces uses the two dimensions of multidimensional data to locate a face position in the two display dimensions. This technique map all attributes of multidimensional data to the form, dimensions, and orientation of human facial feature, like the eyes, nose, mouth, ears, etc[25].

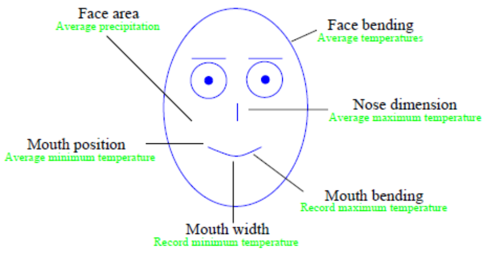


Figure 15: Chernoff face [25]

Although this representation is very interesting, Chernoff face is quite limited with a large number of data items visualized.

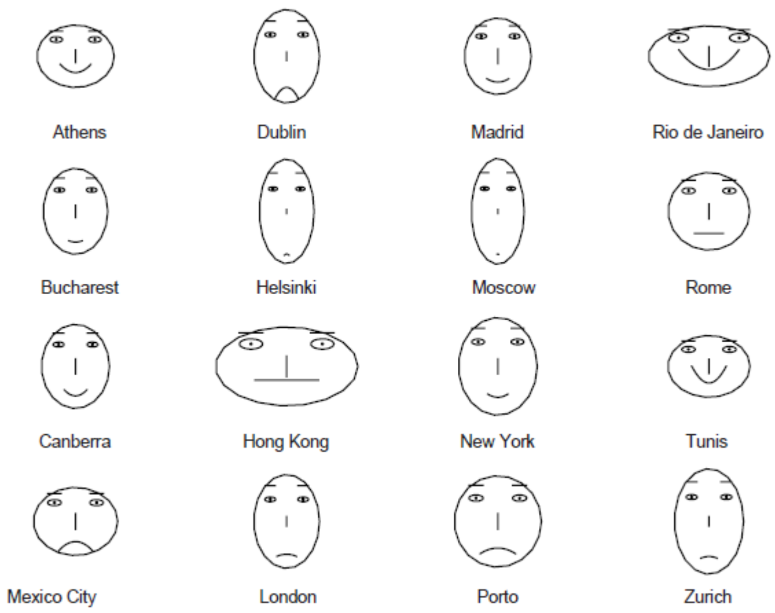


Figure 16: Annual climatic value in Celsius of some world cities (the same dataset with stat plot example. Image generated by the S-PLUS statistics tool [25]

Stick Figures is another famous icon-based technique which use stick figures for visualizing a larger amounts of data, therefore, an adequate number of data items can be presented for data mining purposes [30]. This technique maps two most important attributes (dimensions) to the two display dimensions, and the other attributes are mapped to angles and/or length of limbs of the stick figures. Different stick figure icons with variable dimensionality may be used to visualize the same dataset.

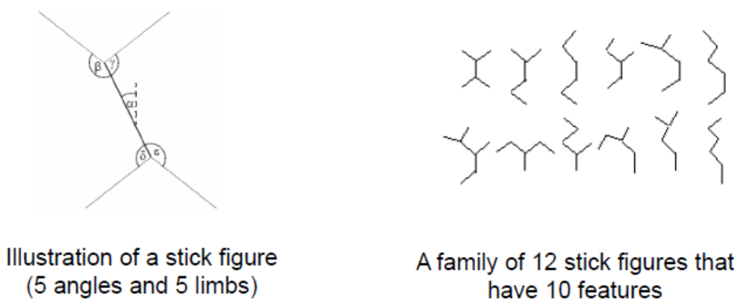


Illustration of a stick figure
(5 angles and 5 limbs)
A family of 12 stick figures that
have 10 features

Figure 17: Stick figure technique

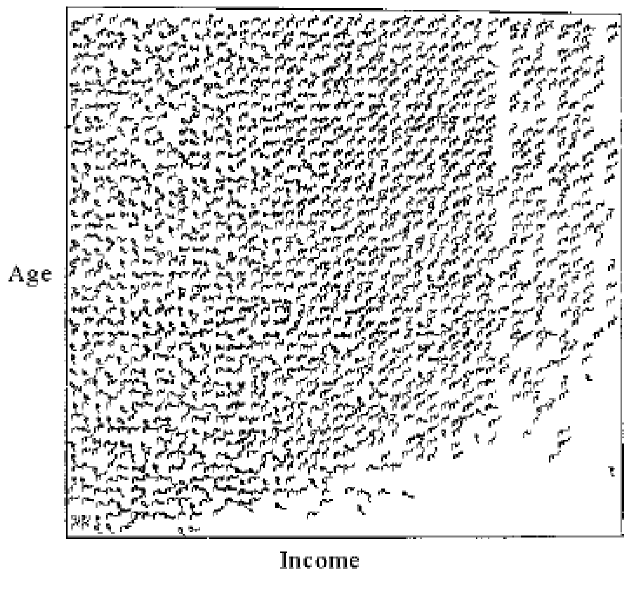


Figure 18: Stick figures of 1980 US census data [30]

Above figure shows the census data of 1980 United States visualized by the stick figure visualization technique, and the census data have five dimensions. As we can see from the figure, income and age are mapped to display dimensions. Occupation, education levels, marital status, and sex are mapped to stick figure features.

However, it can be observed from the figure, the user is really difficult to understand and interpret the graph of stick figures in case of the data records are relatively dense with respect to the display. It means that the user has to have a good training in advance.

Icon-based techniques can display multidimensional properties of medium datasets with a few thousand data records, as icons tend to use a screen space of several pixels. With the amount of data increasing, interpretation is not straightforward and users are required to train so that they can understand most properties of data intuitively; this is because the user cannot focus on the details of each icon when the data scale is very large.

Pixel-oriented Techniques. This techniques map each attribute value of data to a single colored pixel, yielding the display of the most possible information at a time [31, 32]. Each variable is represented as a subwindow in the display which is filled with colored pixels. A k-variables record is represented by k colored pixels (each pixel in one subwindow associated with a variable). In practical, Pixel-oriented techniques use various color mapping techniques to map each attribute to a colored pixel.

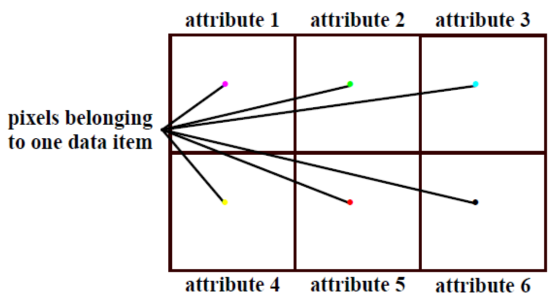


Figure 19: Arrange of Windows for displaying 6 attributes [32]

We can classify this technique into two types. The first is Query-independent techniques which visualize the entire dataset. Screen-filling Curve Techniques (the Peano-Hilbert curve is the best optimization) and Recursive pattern technique are two common methods in this group. The second is Query-dependent techniques which visualize a subset of data that are relevant to the context of a specific user query. This group involves Spiral technique and Circle segment. More reference in [32].

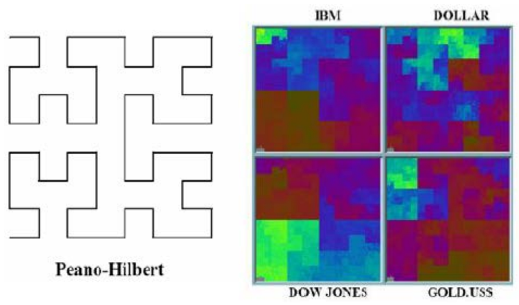


Figure 20: The pixel-based visualization of a financial dataset using Peano-Hilbert arrangement.

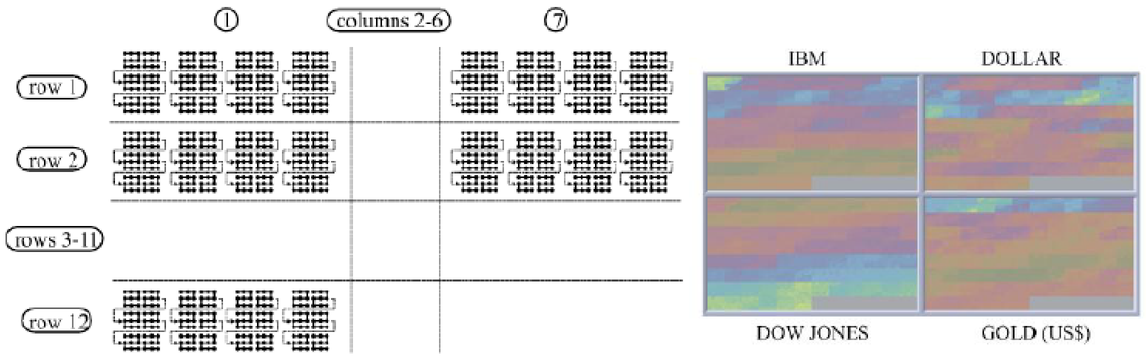


Figure 21: Schematic Representation (left) of the Highly-Structured Arrangement (right) [32]

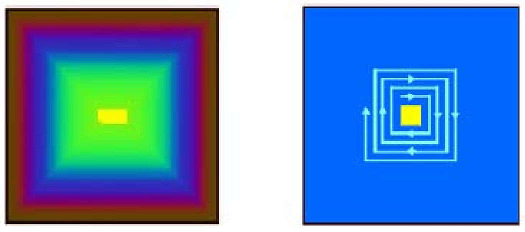


Figure 22: Window that shows the overall distance (left) and Spiral arrangement of pixels (right)

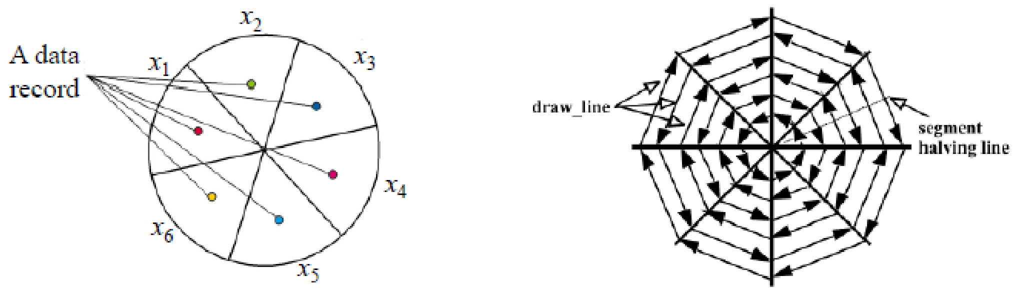


Figure 23: Circle segment representation of a dataset with 6 variables (left) and Circle segment pixel arrangement for a dataset with 8 variables [33]

Pixel-oriented techniques are powerful to handle large and very large datasets on high-resolution displays, and they give an efficient perception of small regions of interest. Moreover, each data record is uniquely mapped to a pixel so data record overlapping and visual cluttering do not occur.

However, the close data items are colored similarly, but distributed in time series order by the pixel-oriented techniques, which cannot visualize the insight of clusters very well. Thus, this technique limited in revealing quantitative relationships between variables because color is not effective in visualizing quantitative values.

Besides Geometric Techniques, Icon Technique, Pixel-based technique, Hierarchical Techniques which subdivide the k-D data space and present subspaces in a hierarchical fashion (including Dimensional stacking [35], Treemap [36], Cone Trees [37]) is one of well-known technique for visualizing multidimensional dataset [34]. In addition, many other techniques have been proposed in multidimensional data visualization [38]. However, most of them suffer from the weakness on visualizing large amount data items and higher dimensional data. Some techniques are limited by providing clearly clustered perception in visual form to the data miner.

Below section will take a brief survey some visualization techniques to assist data miners for finding cluster patterns in dataset.

Visual Cluster Analysis

In the last decade, a large number of novel information visualization techniques have been developed, allowing visualizations of multidimensional data sets without inherent two- or three-dimensional semantics. The techniques can be classified based on three criteria (see below figure) [41]: The data to be visualized, the visualization technique, and the interaction technique used.

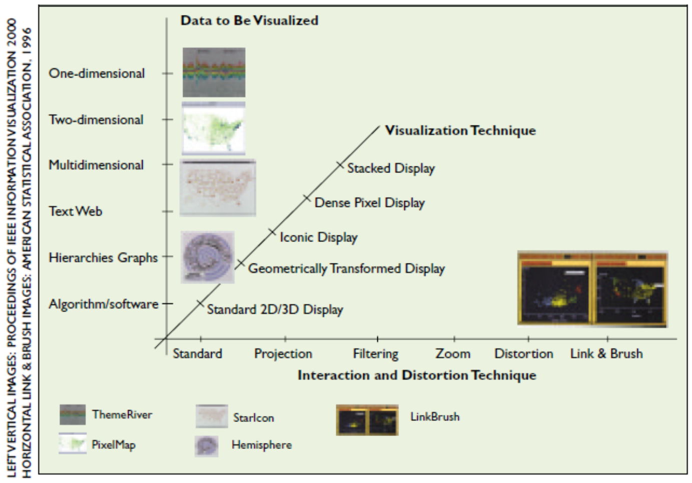


Figure 24: Classification of Information Visualization Techniques [41]

Data sets may be one-dimensional, two-dimensional, and multidimensional or may have more complex data types such as text/hypertext or hierarchies/graphs. Sometimes, a distinction is made between dense (or grid) dimensions and the dimensions that may have arbitrary values. Depending on the number of dimensions with arbitrary values the data is sometimes also called univariate, bivariate, or multivariate.

The visualization technique used may be classified as [40]: Standard 2D/3D displays (bar charts and x-y plots), geometrically transformed displays (Scatterplot matrices, Star coordinates and parallel coordinates), Icon-based displays (Chernoff Faces and Star Glyphs), Dense pixel displays (the recursive pattern and circle segments), Hierarchical techniques (treemaps and dimensional stacking). The classes correspond to basic visualization principles that may be combined in order to implement a specific visualization system.

Interaction techniques allow users to directly navigate and modify the visualizations, as well as select subsets of the data for further operations. Examples include: Dynamic Projection, Interactive Filtering, Interactive Zooming, Interactive Distortion, Interactive Linking and Brushing

A specific system may be designed to support different data types and that it may use a combination of visualization and interaction techniques. There are a large number of visualization techniques that can be used for visualizing data. In addition to standard 2D/3D-techniques such as x-y (x-y-z) plots, bar charts, line graphs, and maps, there are a number of more sophisticated classes of visualization techniques. Several representative visualization techniques that are especially important in cluster analysis are discussed below such as Grand Tour [42], OPTICS [12], HD-EYE [20], H-BLOB [22], Fastmap [43], Star Coordinate [44], SOM-based techniques [44, 45, 46].

Data sets may be one-dimensional, two-dimensional, and multidimensional or may have more complex data types such as text/hypertext or hierarchies/graphs. Sometimes, a distinction is made between dense (or grid) dimensions and the dimensions that may have arbitrary values. Depending on the number of dimensions with arbitrary values the data is sometimes also called univariate, bivariate, or multivariate.

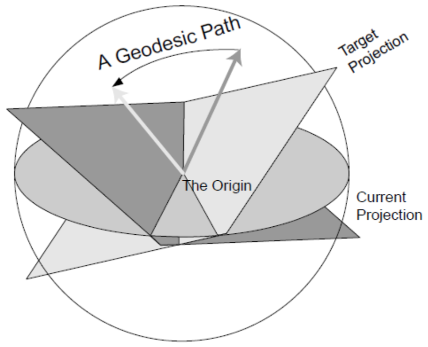
The visualization technique used may be classified as [40]: Standard 2D/3D displays (bar charts and x-y plots), geometrically transformed displays (Scatterplot matrices, Star coordinates and parallel coordinates), Icon-based displays (Chernoff Faces and Star Glyphs), Dense pixel displays (the recursive pattern and circle segments), Hierarchical techniques (treemaps and dimensional stacking). The classes correspond to basic visualization principles that may be combined in order to implement a specific visualization system.

Interaction techniques allow users to directly navigate and modify the visualizations, as well as select subsets of the data for further operations. Examples include: Dynamic Projection, Interactive Filtering, Interactive Zooming, Interactive Distortion, Interactive Linking and Brushing

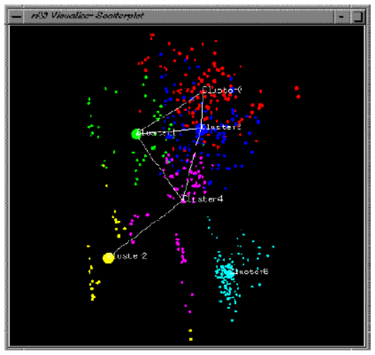
A specific system may be designed to support different data types and that it may use a combination of visualization and interaction techniques. There are a large number of visualization techniques that can be used for visualizing data. In addition to standard 2D/3D-techniques such as x-y (x-y-z) plots, bar charts, line graphs, and maps, there are a number of more sophisticated classes of visualization techniques. Several representative visualization techniques that are especially important in cluster analysis are discussed below such as Grand Tour [42], OPTICS [12], HD-EYE [20], H-BLOB [22], Fastmap [43], Star Coordinate [44], SOM-based techniques [44, 45, 46].

Grand Tour. The grand tour, one of the most popular methods for multidimensional data exploration, is based on orthogonally projecting multidimensional data to a sequence of lower dimensional subspaces (2D space or 3D space) and then moving continuously from one to another in order to obtain different perspectives of data.

While this approach works well with low- to medium-dimensional data sets, it is difficult to apply to large high-dimensional data sets, especially if the clusters are not clearly separated and the data set also contains noise (data that does not belong to any cluster). In this case, more sophisticated visualization techniques are needed to guide the clustering process, select the right clustering model, and adjust the parameter values appropriately. Yang proposed a Grand Tour based system named n23Tool (use in a CAVE virtual reality environment) which scales well to large datasets [42]. The computational complexity of tour is linear to the number of variables. The number of variables matters only in calculating dot products which has a linear complexity to the dimensionality of arguments. Making projections and scatterplots have a computational complexity linear to the number of data points.



Moving 2D projections along a geodesic path in a 3D space.



A cluster similarity graph adds yet another information dimension.

Figure 25: The Grand Tour technique [42]

However, these techniques are not intuitive to users because Grand Tour systems have several times projections which requires users more assistance to understanding.

OPTICS. Ordering Points To Identify the Clustering Structure is a system that uses visualization techniques to help in high-dimensional clustering. The idea of OPTICS is to create a one-dimensional ordering of the database representing its density based clustering structure and visualizes them in “Gaussian bumps” [12].

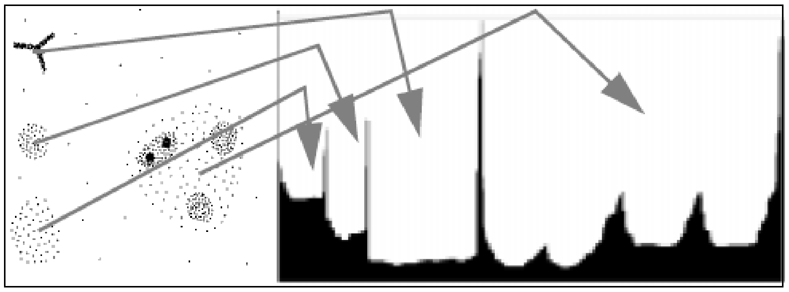


Figure 26: Data set (left) and Reachability Plot mapped in Gaussian bumps [12]

Above Figure shows a two-dimensional example data set together with its reachability distance plot. Intuitively, points within a cluster are close in the generated one-dimensional ordering and their reachability distance shown in the figure is similar.

The reachability plot provides a visualization of the inherent clustering structure and is therefore valuable for understanding the clustering and guiding the clustering process. It works well in finding the basic arbitrarily shaped clusters, as presented in below Figure. It is an intuitive method to assist the user to observe cluster structures.

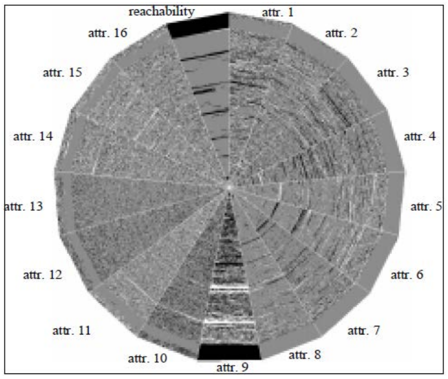


Figure 27: Clustering structure of 30,000 16-d objects visualized by OPTICS [12]

However it lacks the ability in helping the user understand inter-cluster relationships. Its non-linear time complexity makes it neither suitable to deal with very large data sets, nor suitable to provide the contrast between clustering results OPTICS also visualizes clusters in 1D visualization manner [12].

HD-Eye. Another interesting approach is the HD-Eye system [20]. The HD-Eye system considers the clustering problem as a partitioning problem and supports a tight integration of advanced clustering algorithms and state-of-the-art visualization techniques, allowing the user to directly interact in the crucial steps of the clustering process. This approach combines the strengths of an advanced automatic clustering algorithm with new visualization techniques that effectively support the clustering process by representing the important information visually. The visualization techniques use a combination of pixel-oriented density plots and iconic representations of the data and allow users to directly specify cluster separators in the visualizations.

Novel visualization techniques are employed to help the user identify the most interesting projections and subsets as well as the best separators for partitioning the data. Below Figure shows an example screen shot of the HD-Eye system with its basic visual components for cluster separation. The separator tree represents the clustering model produced so far in the clustering process.

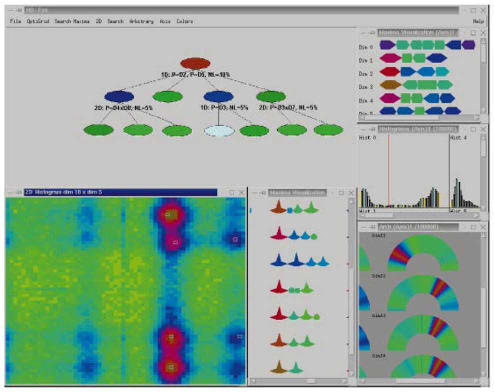


Figure 28: Examples of a projection and separator tree. Clockwise from the top: a projection and separator tree, iconic representation for 1D projections, curve-based density plots, color-based density plots, iconic representation for 2D projections, and a color-based 2D density plot.

HD-Eye is also addressed to use 3D techniques to visualize data in mountain-like structures, and use the intersected planes of the 3D graphs for presenting the trails of the graphs on the planes in different level in 2D forms [HKW99]. But the kernel density-based 3D graphs formation in HD-Eye limits it to be employed in the interactive cluster detection of large datasets.

Hierarchical BLOB. Based on the BLOB clustering method, Sprenger et al [22] presented a technique for visualizing hierarchical clusters in the nested blobs which groups and visualizes cluster hierarchies at multiple levels-of-detail, extended by the capability to handle hierarchical settings. This method combines grouping and visualization in a two stage process constructing a hierarchical setting. In the first stage a cluster tree is computed making use of an edge contraction operator. Exploiting the inherent hierarchical structure of this tree, a second stage visualizes the clusters by computing a hierarchy of implicit surfaces.

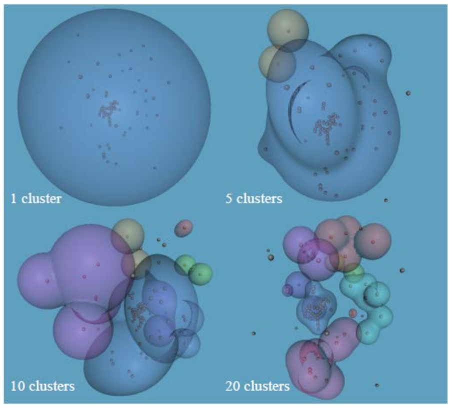


Figure 29: Cluster hierarchies are shown with 20, 10, 5 and 1 cluster [22].

The most significant feature of their technique is that, H-BLOB not only provides the overview manner of whole dataset in blobs, but also gives the detailed visual representation of lower leveled clusters. Actually, it is a very intuitive and easily understood visual presentation.

However the high visual complexity of the two stages of blob graphs' formation makes them unsuitable to be applied in cluster visualization of very large sized datasets.

Star Coordinates and VISTA. Kadogan proposed a method called star coordinate [43] which provided valuable insight on several real data sets for cluster discovery and multifactor analysis tasks.

In star coordinates, each dimension is represented as a vector radiating from the center of a unit circle in a two-dimensional plane. Initially, all axes have the same length and are uniformly placed on the circle. Data points are scaled to the length of the axes, with the minimum being mapped to the origin and the maximum to the other end of the axes on the unit circle. The idea of Star Coordinates technique is intuitive, which extends the perspective of traditional orthogonal 2D X-Y and 3D X-Y-Z coordinate's technique to a higher dimensional space.

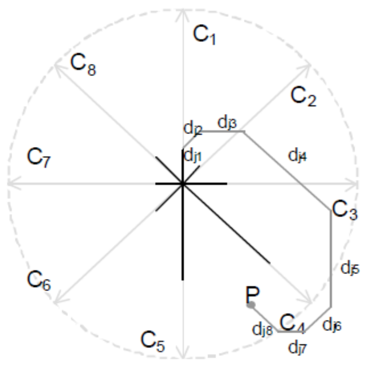


Figure 30: Calculation of data point location for an 8-dimensional dataset.

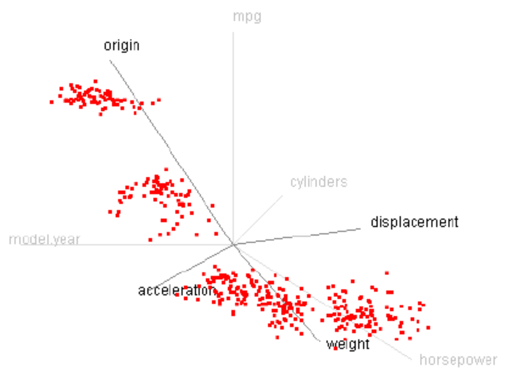


Figure 31: Cluster analysis on approximately 400 cars manufactured world-wide – based on Star Coordinate technique [43]

Based on Star Coordinate technique, Ke-Bing Zhang et al proposed HOV^3 as an Approach for Visual Cluster Analysis [48] to assist data miners in cluster analysis of high-dimensional datasets by visualization. The HOV^3 visualization technique employs hypothesis oriented measures to project data and allows users to iteratively adjust the measures for optimizing the result of clusters. The technique can improve the effectiveness of the cluster analysis by visualization and provide a better, intuitive understanding of the results.

Star Coordinates provides users the ability to apply various transformations dynamically, integrate and separate dimensions of interest, analyze correlations of multiple dimensions, view clusters, trends, and outliers in the distribution of data. However, the existing Star Coordinates based visualization techniques employed in cluster analysis tend to be used as information rendering tools, but do not perform well on verifying the validation of clustering results.

SOM−Based Data Visualization Methods. The self-organizing map (SOM) is a neural network algorithm based on unsupervised learning. It implements an ordered dimensionality reducing mapping of the data training data [44]. The map follows the probability density function of data and is robust to missing data. Indeed, it is really a useful method for visualization of complex multidimensional data [46]. We will review SOM technique in next section.

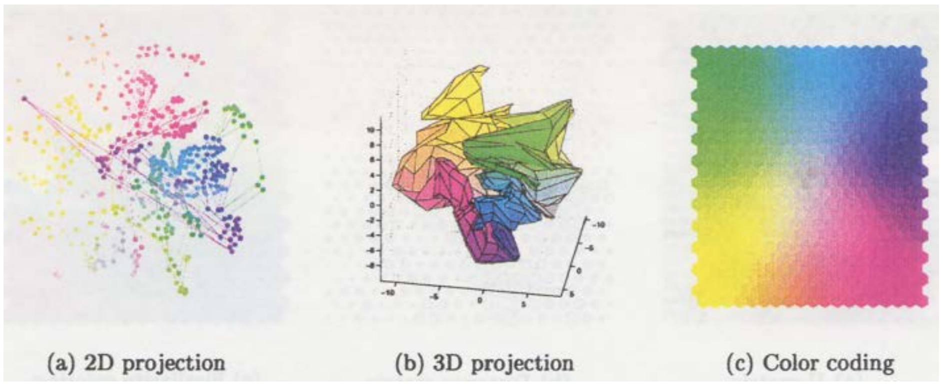


Figure 32: The overall shape of data cloud can be visualized by SOM [46]

Based on SOM technique, Kaski el. al employs to project multidimensional data sets to 2D space for matching visual models [45]. Technically, in their method, a sample data is mapped into a bar graph, and then the graph is compared with all existing vector models in bar graphs to find the most matched one, where the bar graphs in the rectangular region are existing models, X_k is the sample bar graph – below figure.

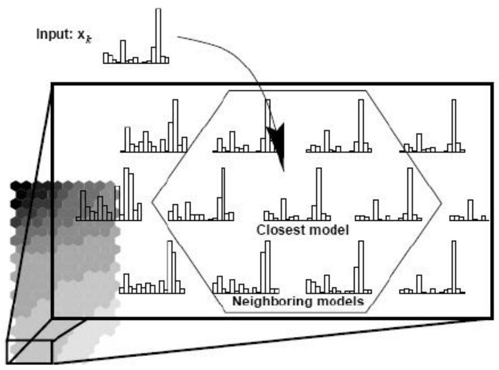


Figure 33: Model matching with SOM by [45]

Another example is WEBSOM project which based on the Self-Organizing Map. These methods are insightful for information retrieval. We can see more detail in [47].

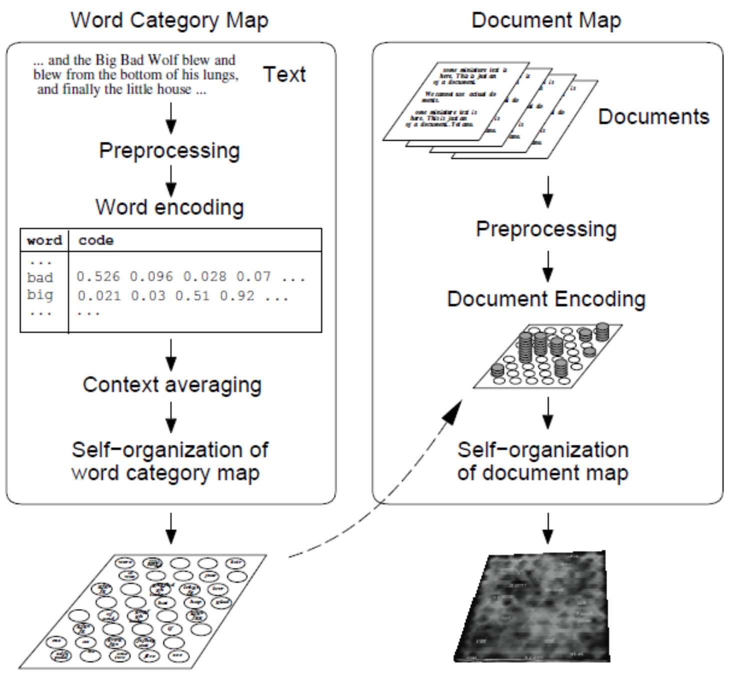


Figure 34: The basic architecture of WEBSOM method [47]

However, the traversal matching process is time-consuming. On the other hand, the SOM technique is based on a single projection strategy. It is not powerful enough to discover all the interesting features from the original data. Another drawback of this technique is that, with an increasing number of dimensions, the bar graphs would be wider. As a result, the user cannot easily observe the matched cluster model in intuition.

Self-Organizing Data Visualization Using Decentralized Multi-agent Systems. Self-organizing data visualization is fundamentally different from most other agent based approaches known in the fields of visualization and data mining, which tend to focus on using agent intelligence for data analysis, visualization data flow, or software development optimization.

The self-organizing data visualization concept is based on mapping data items directly onto agents [49, 50, 51]. Each single data item (e.g. data object, data tuple, or database row, retrieved from a database or dataset) is mapped onto a unique data visualization agent. The aim of this particular approach is to let agents "behave" according to their individual data values according or to any differences with the data values of their neighbors.

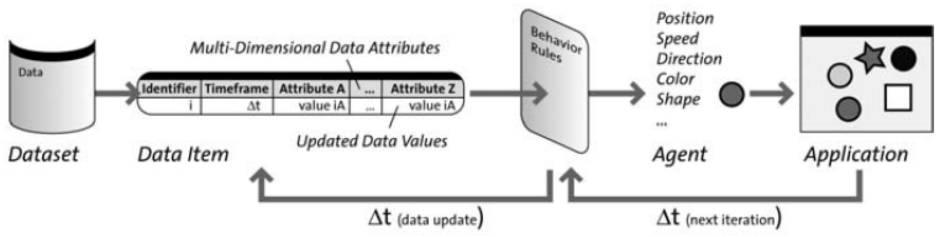


Figure 35: A data visualization agent and its behavior as determined by data-driven behavior rules. [49]

The information flocking method simulates interpretable motion typologies, with the concept of swarming. Individual agents are enhanced with limited vision and communication capabilities, so they can identify other agents in their neighborhood, read their data items, and move toward or away from them without the need for external forces.

Swarming is based on the mathematical simulation of flocking birds or schooling fish. More specifically, swarming is believed to reflect the underlying internal relationships and energy considerations.

Reynolds [50] used observations of real flocks to derive the three primary behavior rules that each member of a flock needs to follow. Each of these rules is applied in parallel between all pairs of boids when the distance between them is smaller than predefined threshold values. Instead of simply averaging all behavior rules, a prioritized acceleration allocation strategy has to be used. This approach allocates priority according to the importance of the rule and normalizes the resulting values when a rule receives less than what was requested. This weighted average is used to compute the final velocity vector. The three behavior rules that govern each boid include (see below Figure):

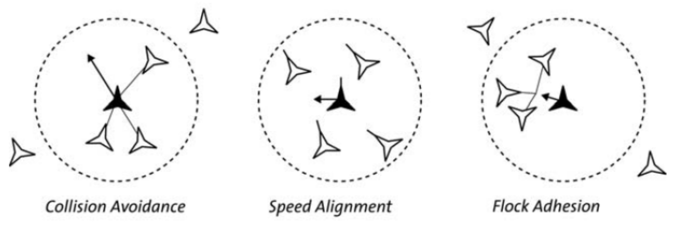


Figure 36: Standard boid behavior rules (after Reynolds 1987).

- Collision Avoidance. Boids need to move away from other boids nearby, in order to avoid crashing into one another.
- Velocity Matching. Each boid should move with about the same speed as the agents in its neighborhood. As a result, multiple boids, as a group, seem to pursue a common goal.
- Flock Centering. Boids should attempt to stay close to other boids nearby. Boids in the center will feel no pull outward, whereas boids on the periphery will be deflected inward. As a result, each boid stays relatively near the center of the flock.

In addition to the three traditional swarming rules mentioned earlier, each agent is governed by additional two behavior rules:

- Data Similarity. Each agent attempts to stay close to those agents with similar data values. This attraction rule results in the spatial clustering of agents that have similar data items. The similarity between agents is determined by testing whether the difference in data values of a single data attribute is lower than a predefined data similarity threshold.
- Data Dissimilarity. Each agent attempts to move away from those agents with dissimilar data values. This repulsion rule significantly accelerates the clustering

effect of the first rule. Agents clustering too slowly would eventually cause the visualization to be confusing and nonrepresentative. Accordingly, the continuous progression of the timeline requires all boids to cluster, uncluster, and recluster efficiently.

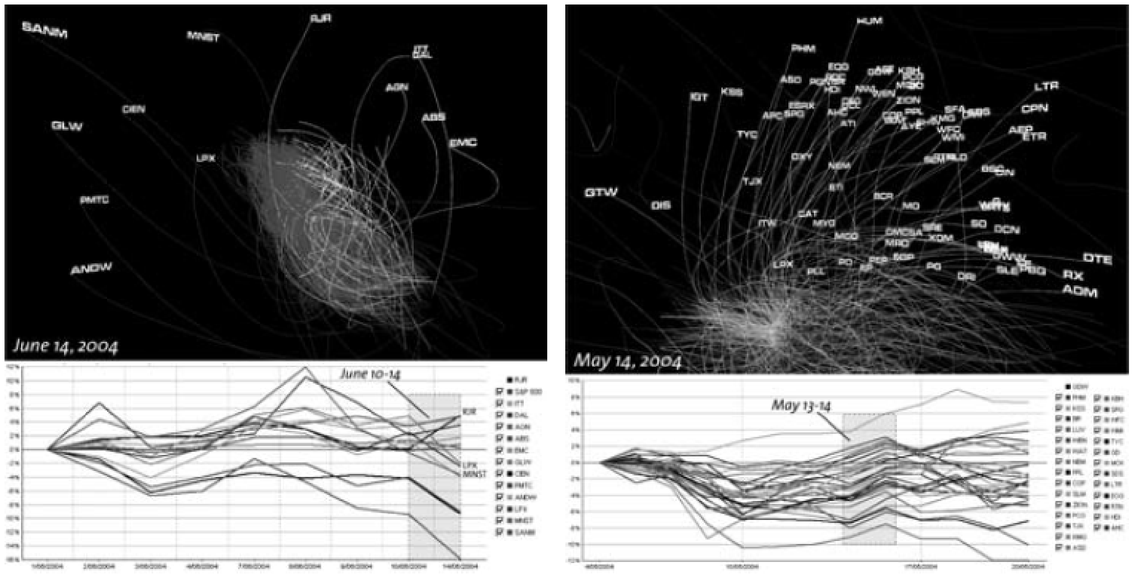


Figure 37: Short-term zoning patterns: Individual boids being expulsed by the main flock (left), and subflocks with similar data value changes appearing from the main flock (right) [49]

The information flocking method is able to represent short-term (e.g., clustering) as well as long-term (e.g., zoning) data tendencies in datasets with complex and potentially randomly changing data values. It is capable of displaying hundreds of time varying data items simultaneously, even when the data changes in real time, as it performs the data value comparisons within the visualization canvas itself. Displaying or even visually analyzing such a number of stock market companies simultaneously via line graphs would be impracticable. Whereas this approach is certainly not ideal for real-world stock exchange applications, it is proposed as a prototype toward more usable visualizations that are able to perform data analysis as well as data representation simultaneously and in real time. It also illustrates how motion typology and spatial clustering can be used to convey complex data tendencies.

Major Challenges

By the above analysis, we can summarize that the visualization techniques to be used in cluster analysis should be able to handle several important aspects of visual perception:

- Visualizing large and multidimensional datasets;
- Providing a clear overview and detailed insight of cluster structure;
- Having linear time complexity on data mapping from higher dimensional space to lower dimensional space;
- Supporting interactive cluster visual representation dynamically;
- Involving knowledge of domain experts into the cluster exploration;
- Giving data miners purposeful and precise guidance of cluster investigation and cluster validation rather than simply random cluster exploration;

As discussed above, most existing cluster visualization techniques work well on visualizing multidimensional data sets. However, as the size and dimensionality of data sets increase, these techniques do not perform well on very large data visualization, they can hardly deal with visual representation of higher dimensional data, they cannot provide an intuitive overview of cluster structure, etc. In short, they satisfy not all of the above requirements.

CHAPTER III

METHODOLOGY AND RESULT

A question arises: which visualization technique can provide a genuine representation of cluster structure of data? In practice, a few visualization techniques can achieve the above requirements. As Seo and Shneiderman pointed out, “A large number of clustering algorithms have been developed, but only a small number of cluster visualization tools are available to facilitate researchers’ understanding of the clustering results”. How to preserve the identity of “problem domain” and “representation domain” by visualization is the critical challenge of cluster visualization.

Visualization is typically employed as an observational mechanism to assist users with intuitive comparisons and better understanding of the studied data. Instead of precisely contrasting clustering results, most of the existing visualization techniques employed in cluster analysis focus on providing the user with an easy and intuitive understanding of the cluster structure, or explore clusters randomly.

In general, it is not easy to visualize multidimensional data sets on 2D space and give a “genuine” visual interpretation. This is because mapping multidimensional data onto 2D space inevitably introduces overlapping and bias. For mitigating the problem, Star Coordinates based techniques provide some visual adjustment mechanisms. However, the stochastic adjustment of Star Coordinates and VISTA limits their usability in cluster analysis.

We propose a visual 3D graph layout called an atomic structure model for document cluster. This model use SOM algorithm (3D or 2D) to estimate number of cluster which will be separated from a set of documents – PDF format. Then, Based on vector space model, in particular tf-idf vector, we use k-means/k-means++ or 1D-SOM to cluster the datasets into k groups. Then we visualize each cluster as an atom structure.

Pre-Processing

Read Pdf file

Using library PDFBox-0.7.3.dll to read all content (text only) from a Pdf file by function ReadFullPDF and stores them into a member of class PDFBOX which is String Text.

The first task is pre-processing the lexicon. This typically involves three parts: removal of stop words, stemming, and term weighting. Any one, all, or none of these can be applied to the lexicon. The application and usefulness of these methods is an open research question in text data mining and should be of interest to the statistical community [52].

Remove Stop-words

Stop-words are words that from non-linguistic view do not carry information. Stop words can be a pre-specified list of words or they can be dependent on the context or corpus. They have mainly functional role. We usually remove them to help the methods to perform better.

Natural language dependent –examples some stop words in English: A, ABOUT, ABOVE, ACROSS, AFTER, AGAIN, AGAINST, ALL, ALMOST, ALONE, ALONG, ALREADY, ALSO, and so on.

Stemming

Stemming is often applied in the area of information retrieval, where the goal is to enhance system performance and to reduce the number of unique words. Stemming is the process of removing suffixes and prefixes, leaving the root or stem of the word.

Different forms of the same word are usually problematic for text data analysis, because they have different spelling and similar meaning (e.g. learns, learned, learning, etc.)

This makes sense, since the words have similar meaning. However, some stemmers would reduce the word probate to probe and the word relativity to relate, which convey different meanings. This does not seem to affect results in information retrieval, but it could have some undesirable consequences in classification and clustering. Stemming and the removal of stop words will reduce the size of the lexicon, thus saving on computational resources. The Porter stemming algorithm is a commonly employed stemming procedure [53].

For English it is not a big problem -publicly available algorithms give good results. Most widely used is Porter stemmer at http://www.tartarus.org/~martin/PorterStemmer/

Example cascade rules used in English Porter stemmer

□ATIONAL -> ATE relational-> relate

□TIONAL -> TION conditional-> condition

□ENCI -> ENCEvalenci -> valence

□ANCI -> ANCE hesitanci-> hesitance

□IZER -> IZE digitizer-> digitize

ABLI -> ABLE conformabli-> conformable

ALLI -> AL radicalli-> radical

ENTLI -> ENT differentli-> different

ELI -> E vileli-> vile

OUSLI -> OUS analogousli-> analogous

We use public void StemFile(String filename, String fileout) or public string StemString(String inString) to get result.

Bag-of-words document representation

Word weighting: in bag-of-words representation each word is represented as a separate variable having numeric weight.

The most popular weighting schema is normalized word frequency TFIDF: [54]

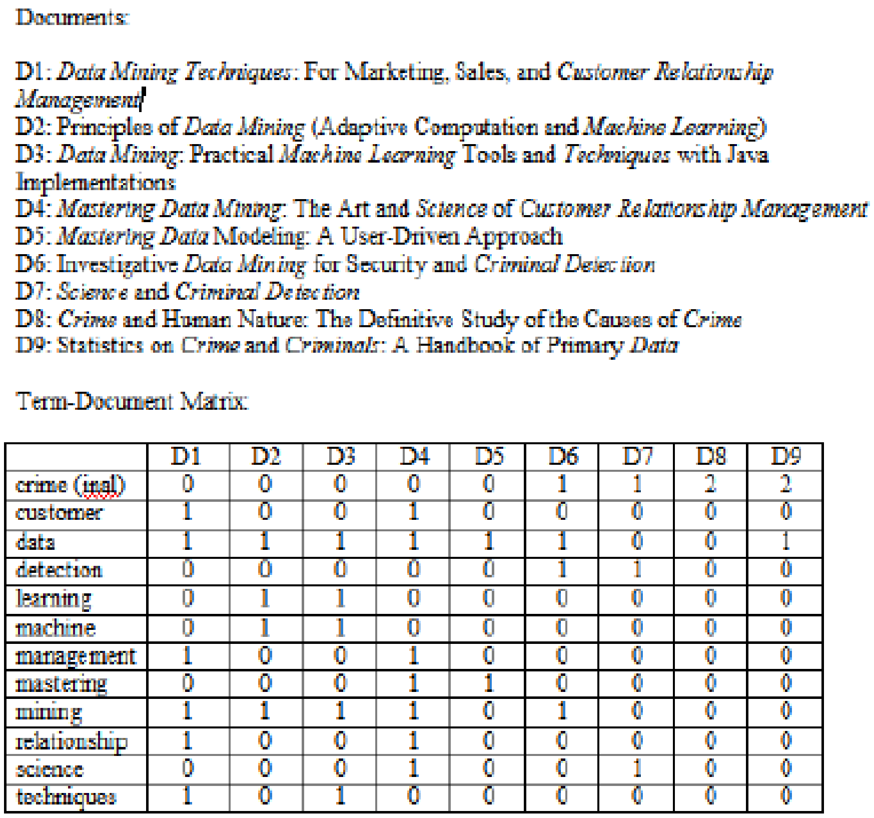
$$tfidf(w) = tf * log\left(\frac{N}{df(w)}\right)$$

Where:

- tf(w) : term frequency (number of word occurrences in a document)
- df(w): document frequency (number of documents containing the word)
- N: number of all documents
- tfidf(w): relative importance of the word in the document.

As we can see from the formula, the word is more important if it appears several times in a target document (tf) and the word is more important if it appears in fewer documents (df).

The corpus is encoded as a term-document matrix X with n rows and p columns, where n represents the number of words in the lexicon (full or pre-processed lexicon), and p is the number of documents. Thus, the xij element of the matrix X contains the number of times the i-th term appears in the j-th document (the term frequency). As we stated before, this could also be a weighted frequency. Encoding the corpus as a matrix allows one to utilize the power of linear algebra to analyze the document collection.



Documents:

D1: *Data Mining Techniques*: For Marketing, Sales, and *Customer Relationship Management*

D2: Principles of *Data Mining* (Adaptive Computation and *Machine Learning*)

D3: *Data Mining*: Practical *Machine Learning* Tools and *Techniques* with Java Implementations

D4: *Mastering Data Mining*: The Art and *Science* of *Customer Relationship Management*

D5: *Mastering Data* Modeling: A User-Driven Approach

D6: Investigative *Data Mining* for Security and *Criminal Detection*

D7: *Science* and *Criminal Detection*

D8: *Crime* and Human Nature: The Definitive Study of the Causes of *Crime*

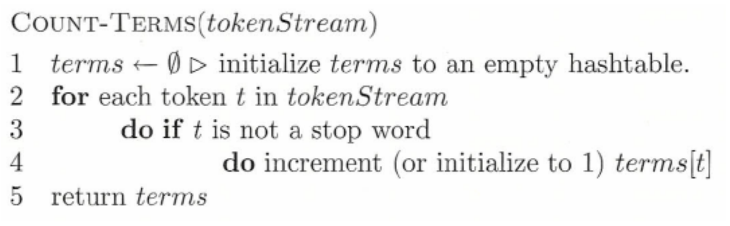
D9: Statistics on *Crime* and *Criminals*: A Handbook of Primary *Data*

Term-Document Matrix:

	D1	D2	D3	D4	D5	D6	D7	D8	D9
crime (inal)	0	0	0	0	0	1	1	2	2
customer	1	0	0	1	0	0	0	0	0
data	1	1	1	1	1	1	0	0	1
detection	0	0	0	0	0	1	1	0	0
learning	0	1	1	0	0	0	0	0	0
machine	0	1	1	0	0	0	0	0	0
management	1	0	0	1	0	0	0	0	0
mastering	0	0	0	1	1	0	0	0	0
mining	1	1	1	1	0	1	0	0	0
relationship	1	0	0	1	0	0	0	0	0
science	0	0	0	1	0	0	1	0	0
techniques	1	0	1	0	0	0	0	0	0

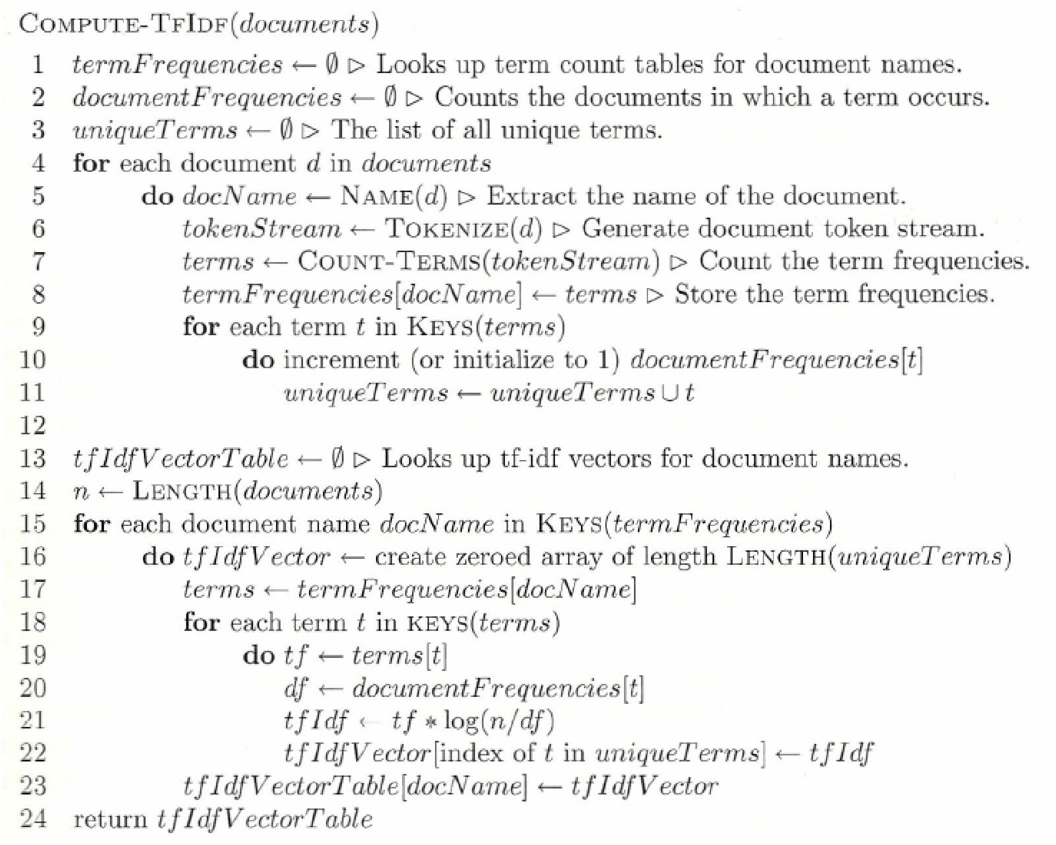
Figure 38: Here we show a small corpus of nine book titles on data analysis and detecting crime; each title is a document. To save space, we are only using the italicized words in the document list. Note that we stemmed the words crime and criminal to crime. The ij-th element of the term-document matrix shows the number of times the i-th word appears in the j-th document. [52]

The pseudocode of computing tfidf matrix is present in below figure:



```
COUNT-TERMS($tokenStream$)
$terms \leftarrow \emptyset$ ▷ initialize $terms$ to an empty hashtable.
for each token $t$ in $tokenStream$
    do if $t$ is not a stop word
        do increment (or initialize to 1) $terms[t]$
return $terms$
```

Figure 39: the pseudocode of word count [55]



```
COMPUTE-TFIDF($documents$)
$termFrequencies \leftarrow \emptyset$ ▷ Looks up term count tables for document names.
$documentFrequencies \leftarrow \emptyset$ ▷ Counts the documents in which a term occurs.
$uniqueTerms \leftarrow \emptyset$ ▷ The list of all unique terms.
for each document $d$ in $documents$
    do $docName \leftarrow$ NAME($d$) ▷ Extract the name of the document.
       $tokenStream \leftarrow$ TOKENIZE($d$) ▷ Generate document token stream.
       $terms \leftarrow$ COUNT-TERMS($tokenStream$) ▷ Count the term frequencies.
       $termFrequencies[docName] \leftarrow terms$ ▷ Store the term frequencies.
       for each term $t$ in KEYS($terms$)
           do increment (or initialize to 1) $documentFrequencies[t]$
              $uniqueTerms \leftarrow uniqueTerms \cup t$

$tfIdfVectorTable \leftarrow \emptyset$ ▷ Looks up tf-idf vectors for document names.
$n \leftarrow$ LENGTH($documents$)
for each document name $docName$ in KEYS($termFrequencies$)
    do $tfIdfVector \leftarrow$ create zeroed array of length LENGTH($uniqueTerms$)
       $terms \leftarrow termFrequencies[docName]$
       for each term $t$ in KEYS($terms$)
           do $tf \leftarrow terms[t]$
              $df \leftarrow documentFrequencies[t]$
              $tfIdf \leftarrow tf * \log(n/df)$
              $tfIdfVector$[index of $t$ in $uniqueTerms$] $\leftarrow tfIdf$
       $tfIdfVectorTable[docName] \leftarrow tfIdfVector$
return $tfIdfVectorTable$
```

Figure 40: the pseudocode of computing tfidf [55]

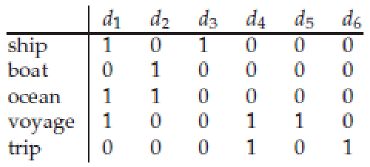
Document Similarity

The standard way of quantifying the similarity between two documents d1 and d2 is to compute the cosine similarity of their vector representations. Each document is represented as a vector of weights D = <x>. Then, Similarity between vectors is estimated by the similarity between their vector representations (cosine of the angle between vectors): [54]

$$Sim(D_1, D_2) = \frac{\sum_i X_{1i}\, X_{2i}}{\sqrt{\sum_j X_j^2}\ \sqrt{\sum_k X_k^2}}$$

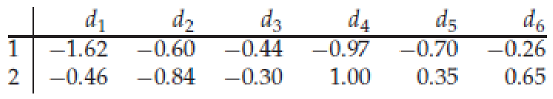
Latent Semantic Indexing

LSI (Latent semantic indexing analysis) [56] is a statistical technique that attempts to estimate the hidden content structure within documents. It uses linear algebra technique Singular-Value-Decomposition (SVD) and it discovers statistically most significant co-occurrences of terms. LSI is an interested method in finding a convenient lower-dimensional space to perform subsequent analysis. This might be chosen to facilitate visualization, clustering, or classification.



	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

the term-document matrix C



	d_1	d_2	d_3	d_4	d_5	d_6
1	−1.62	−0.60	−0.44	−0.97	−0.70	−0.26
2	−0.46	−0.84	−0.30	1.00	0.35	0.65

illustrates the documents in $(V')^T$ in two dimensions.

Figure 41: Example of reducing from 5D to 2D by using LSI [55]

LSI is efficient to reduce dimension of data sets. However, using this technique will deal with a problem which processes a query for searching. For example, from the left figure we have 6 documents with 5 dimensions involving ship, boat, ocean, voyage and trip. After applying LSI (in right figure), dimension of dataset has reduce from 5 to 2 but they change name of attributes; they are “1” and “2”. Now we cannot execute a query which search attribute “ship” in a reduced matrix. Actually, some efforts try to overcome this problem; however, we do not focus on them because they are out of range of our study.

Clustering

As far as we concerned form previous chapter, clustering is a process of finding natural groups in data in an unsupervised way (no class labels pre-assigned to documents). Most popular clustering methods for Document clustering are: K-Means and SOM. Because our model uses an atom metaphor for display and interaction, we need centroid of cluster which represent as neutron of atom. Thus, partition method for clustering, especially K-mean, is a suitable approach. Below is the detail for K-means clustering algorithm.

K-Means clustering

Given:

- set of documents (e.g. TFIDF vectors),
- distance measure (e.g. cosine or Euclidian distance)
- K (number of groups)
- For each of K groups initialize its centroid with a random document
- While not converging
  - Each document is assigned to the nearest group (represented by its centroid)
  - For each group calculate new centroid(group mass point, average document in the group)

A centroid $\vec{\mu}$ of cluster ω: $\vec{\mu}(w) = \frac{1}{|w|} \sum_{\vec{x} \in w} \vec{x}$ where w is cluster and $\vec{x}\ is\ a\ object\ (document) belong\ to\ w$

Algorithms:

Given N is number of documents in data set and K is number of clusters

Documents[N]: array of documents

Clusters[K]: array of cluster centers

Members[N]: array of document belong to cluster (memberships)

```
Kmeans(....)
While $\delta$/N > threshold
    $\delta \leftarrow 0$
    For i = 0 to N-1
        For j = 0 to K-1
            Distance $\leftarrow$ |Documents(i) – Cluster(j)|
            If Distance < $d_{min}$
                $d_{min} \leftarrow$ Distance
                n $\leftarrow$ j
            if Members[i] $\neq$ n
                $\delta \leftarrow \delta + 1$
                Members[i] $\leftarrow$ n
            New_cluster[n] $\leftarrow$ New_cluster[n] + Documents[i]
            New_cluster_size[n] $\leftarrow$ New_cluster_size[n] + 1
        For j = 0 to K -1
            Clusters[j][*] $\leftarrow$ New_cluster[j][*] / New_cluster_size[j]
            New_cluster[j][*] $\leftarrow$ 0
            New_cluster_size[j] $\leftarrow$ 0
```

Figure 42: The sequential algorithm of K-mean based on [3].

Graph based visualization and Organization

After we process all Pdf file to matrix of TFIDF vector, we construct a layout for clusters of the documents. There is general algorithm:

(i) Documents are transformed into the bag-of-words sparse-vectors representation

- Words in the vectors are weighted using TFIDF
- if dimensional space is more lager than one thousand we will use LSI to reduce it to one thousand.

(ii) Build a neutron network for SOM in 3D; based on this map we can estimate value of K groups which should be seperated.

(iii)K-Means clustering algorithm splits the documents into K groups

- Each group consists from similar documents
- Documents are compared using cosine similarity

(iv)K groups form a graph:

- Groups are nodes in graph; similar groups are linked
- Each group is represented by an atomic structure

(v) Using appropriate algorithm to draw a graph: **atom metaphor**

3D-SOM

We built a 3D – Grid (N x N x N) of Neutron. Each neuron has coordinate (X,Y,Z) and weight which is a vector. The vector has the same dimension with pattern (dataset – TFIDF). For example, below figure have 27 neurons (3x3x3).

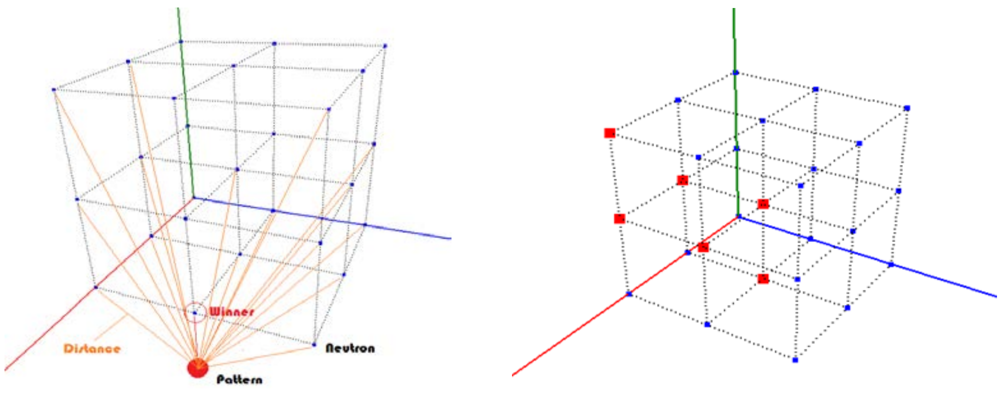


Figure 43: 3D SOM with 3x3x3 neutrons (left) and 6 winners (Red color)

Initially, we assign unique coordinate which get from 3D –grid to each neuron. Weight of neuron $W_{i,j,k}(d1,d2,d3,\ldots,dn)$ are random values in range (0,1).

Next, we will train every neuron from patterns following: [44]

For each input pattern:

Calculate the distance between the pattern and all neurons ($d_{i,j,k} = \|X_p - W_{i,j,k}\|$)

Select the nearest neuron as winner W_{winner}

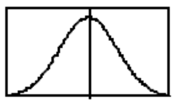
Update weight for each neuron according to the rule:

$$W_{i,j,k} = W_{i,j,k} + \alpha * h\left(winner, W_{i,j,k}, r\right) \|X_p - W_{i,j,k}\|$$

Repeat the process until a certain stopping criterion is met. Usually, the stopping criterion is a fixed number of iterations.

To guarantee convergence and stability of map, the learning rate α and the neighborhood function $h\left(winner, W_{i,j,k}\right)$ are decreased in each iteration, thus converging go to zero.

$$h\left(Winner_{m,n,o}, W_{i,j,k}\right) = e^{-\frac{1}{2}\left(\frac{\sqrt{(i-m)^2+(j-n)^2+(k-o)^2}}{r}\right)^2}$$



This neighborhood function assumed values in [0,1] and is high for neurons that are close in the output space, and small (or 0) for neuron far away. It is usual to select a function that monotonically decreasing and non-zero up to a radius r (call neighborhood radius) and zero from there onwards.

$$\text{Gause:} \quad h\left(\text{Winner}_{m,n,o}, W_{i,j,k}\right) = e^{-\left(\frac{\sqrt{(i-m)^2+(j-n)^2+(k-o)^2}}{e^{-\frac{iterations*length*\log(length)}{1000}}}\right)^2}$$

$$\alpha = e^{-\frac{iteration}{1000}} * 0.1$$

SOM method is not effective approach when vector's dimension is so large, over 1000. However, we can use Latent Semantic Indexing to reduce dimension to about 100.

Another disadvantage of 3D_SOM is that stable map depends on number of iteration of training and number of neuron in network. Both criterion affect to execution time. In fact, if they are not enough large, position (or coordinate) of data point will overlap. Actually, execution time of 3D-SOM is greater than k-mean because it need train NxNxN neuron for each iteration; while k-mean need only N neutron trained. However, if we only use SOM algorithm to cluster data – that mean we don't care about graph layout, 1D- SOM is better than K-means because SOM can classify data without retrain network (k-mean cannot). There are some constrains such as K-mean need number of cluster and SOM need parameter about number of neutron in network and iterations [60].

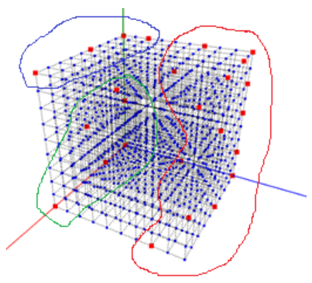


Figure 44: a 10x10x10 grid for 25 documents is generated after using 3D-SOM

After we construct a 3D-SOM, we can estimate value of K based on distances of them so that we can group documents in data sets into K groups. For example, we can estimate K be 3 from above figure.

Now we will build system by using using an atom metaphor for display and interaction in next section.

Atom model for document clustering:

We need to assume that:

a) Each cluster ω is an atom.

Where:

- a center is defined as the mean or centroid $\vec{\mu}$ of the documents CENTROID in a cluster ω: $\vec{\mu}(w) = \frac{1}{|w|} \sum_{\vec{x} \in w} \vec{x}$
- A node (document in cluster) is an electron $\vec{x}$
- Electron has located in its orbit around center. This orbit will be computed by energy level which is distance between electron and center.

$$Energy(\vec{x}, \vec{\mu}) = Distance(\vec{x}, \vec{\mu}) = \frac{\vec{x} \cdot \vec{\mu}}{|\vec{x}||\vec{\mu}|}$$

- Two electrons have the same energy level will be in the same orbit. It should be equally distributed in a sphere.
- An electron is changed its energy by user (by adjusting value of element in the vector). If its energy is greater than threshold, it will break link and become freedom electron from this atom. Then it will join to another atom which can make link with it.

b) If we have many atoms (clusters), we have two approach:

The first approach: their location will be calculated by Size of atom based on the number of electron (data point - document) following:

- Every electron is the same size
- Size of atom = number of electron * size of electron
- Divide main circle in to k angles (parts) which correspond with ratio of atom's size.
- Radius of every part will be in bisector of its angle. The maximum radius will be equal to Radius of main circle and other radius has ratio with max radius based on size of atom.
- Compute radius's location for every part.

For example, in below figure we have 4 cluster C1, C2, C3, C4 with number of electrons respectively n1 < n2 < n3 < n4. In the left, we can see area and size of atom which have been calculated by number of their electrons. Then we adjust the distances from each neutron to center based on a ratio radius/number electron (the right figure).

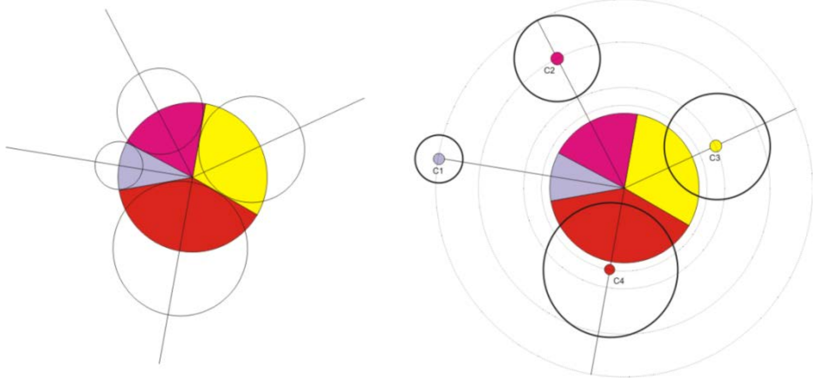


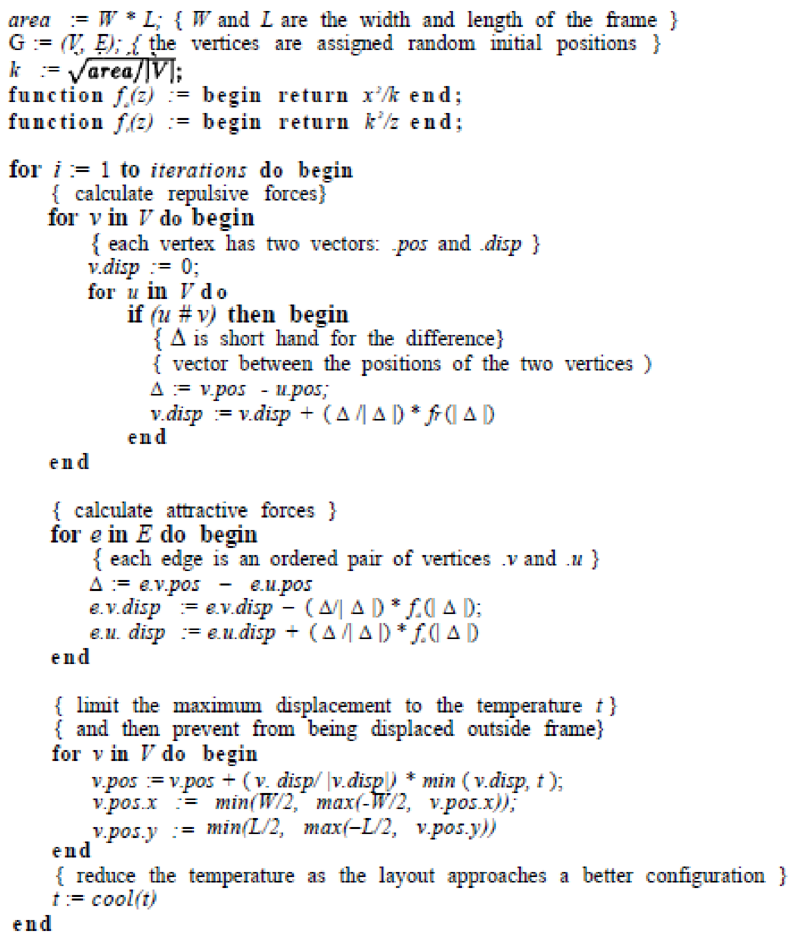
Figure 45: Layout for 4 clusters based on size of electron in Atom

The second approach: we use force-directed algorithm [61, 62] to obtain optimal layout for neutrons. We apply this approach in our program because it presents relationship between neutrons (centroids). In addition, distance between neutrons can be calculated by using Euclidian Distance or cosine of the angle between vectors.

$$Distance\ by\ Cosin(\overrightarrow{Neutron1}, \overrightarrow{Neutron2}) = \frac{\overrightarrow{Neutron1}.\overrightarrow{Neutron2}}{|\overrightarrow{Neutron1}||\overrightarrow{Neutron2}|} = \frac{\sum_i X_{1i}\, X_{2i}}{\sqrt{\sum_j X_j^2}\,\sqrt{\sum_k X_k^2}}$$

$$Euclidian\ Distance(\overrightarrow{Neutron1}, \overrightarrow{Neutron2}) = \sqrt{\sum_i (X_{1i} - X_{2i})^2}$$

Below figure illustrate a graph drawing algorithms by force-directed (FRUCHTERMAN & REINGOLD [61]). We build a graph G(V,E), where V is a set of Neutrons (centroid of clusters) and E is a set of Distances between Neutrons.



```
area  := W * L; { W and L are the width and length of the frame }
G := (V, E); { the vertices are assigned random initial positions }
k  := $\sqrt{area/|V|}$;
function $f_a(z)$ := begin return $x^2/k$ end;
function $f_r(z)$ := begin return $k^2/z$ end;

for i := 1 to iterations do begin
    { calculate repulsive forces}
    for v in V do begin
        { each vertex has two vectors: .pos and .disp }
        v.disp := 0;
        for u in V do
            if (u # v) then begin
                { Δ is short hand for the difference}
                { vector between the positions of the two vertices )
                Δ := v.pos - u.pos;
                v.disp := v.disp + ( Δ /| Δ |) * fr(| Δ |)
            end
    end

    { calculate attractive forces }
    for e in E do begin
        { each edge is an ordered pair of vertices .v and .u }
        Δ := e.v.pos  -  e.u.pos
        e.v.disp  := e.v.disp – ( Δ/| Δ |) * fa(| Δ |);
        e.u. disp  := e.u.disp + ( Δ /| Δ |) * fa(| Δ |)
    end

    { limit the maximum displacement to the temperature t }
    { and then prevent from being displaced outside frame}
    for v in V do begin
        v.pos := v.pos + ( v. disp/ |v.disp|) * min ( v.disp, t );
        v.pos.x  :=  min(W/2,  max(-W/2,  v.pos.x));
        v.pos.y  :=  min(L/2,  max(–L/2,  v.pos.y))
    end
    { reduce the temperature as the layout approaches a better configuration }
    t := cool(t)
end
```

Figure 46: Force-directed placement [61]

Because the K-mean algorithm uses Euclidian distance, we will apply Euclidian distance for these neutrons too. However, we need multiply these distances with a given ratio to prevent overlap orbits of electrons.

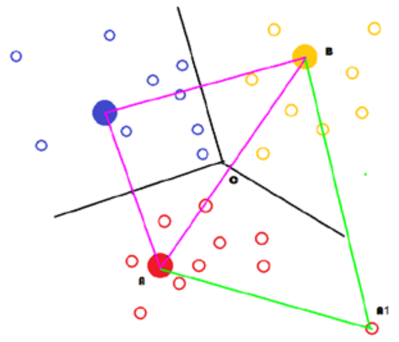


Figure 47: an outlier from K-mean partition

Let consider a 2D layout with 3 clusters after using K-mean in below figure:

AB is distance between centroid A and centroid B, therefore AC = BC.

Electron A1 (a outlier of data point) has distance AA1 < BA1 $\rightarrow$ A1 belong to Atom A (cluster), but AA1 > AC. As a result, when electron A1 rotates around neutron A (centroid) it will trespass on area of atom B. To guarantee overlap, all distance must multiply with a ratio alpha.

In our layout – Atomic structure model in 3D, we present energy of electron in 5 levels. As we known, distances from an electron to its neutron are in$\Re$ after all electrons (vector) were normalized, range [0, max distance]. Therefore, we cannot gain a beautiful layout if we render huge electron's orbits. In a certain level, electron will be evenly distributed on a sphere. The sphere's radiuses correspond with its level.

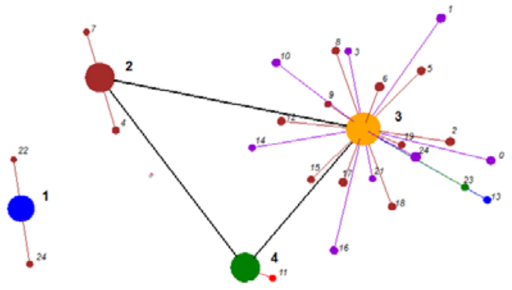


Figure 48: 4 atoms with a neutron and 25 electrons which distribute on 5 levels (red, brow, violet, green and blue, respectively – red level is the closest and blue level is the most far).

As we can see from above figure that 13th electron is so far from 3rd neutron because it has a weak energy (energy = Euclidian distance). Therefore, the electron will move to another atom if we cluster these documents into more groups (atom). Indeed, we can see from figure, neutron 4 (green color) is near neutron 3 (orange color) so 13th electron may move to neutron 4 in next clustering.

In this approach, we have intuitive view to analyze data set from inside of a cluster and relationship between two clusters. Based on 3D layout, space limitation for visualization has been taken into account efficiently.

Let consider an example for visualization with 200 documents.

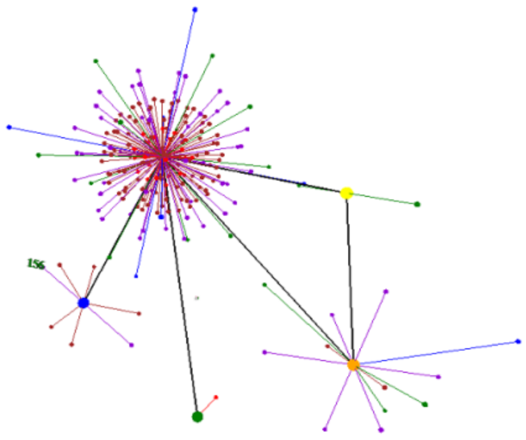


Figure 49: 200 documents are represented by 200 electrons in 5 clusters.

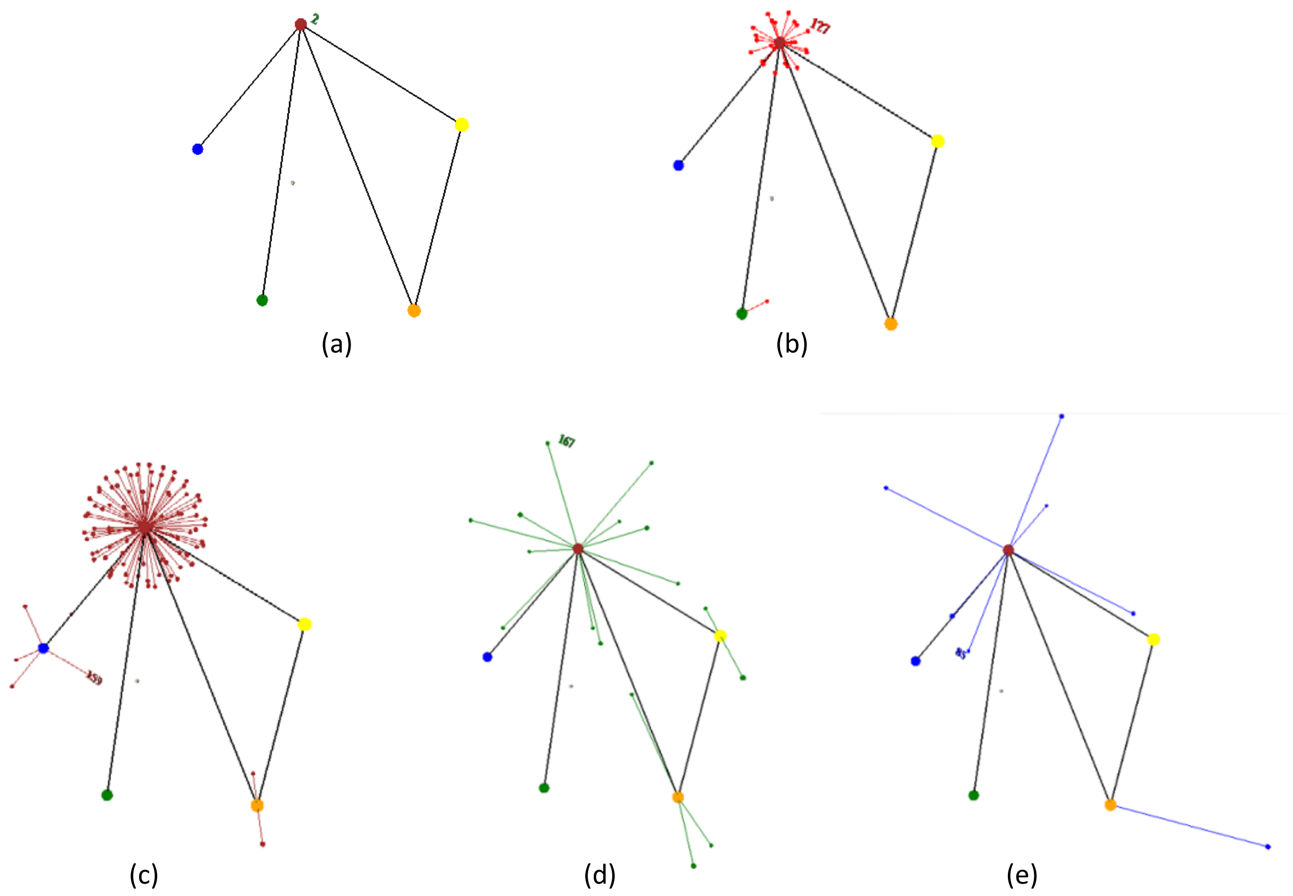


Figure 50: Illustration for electrons by level. Level 0 – only show centroid (a), level 1 (b), level 2(c), level 3(d), level 5 (e). There is no electron in level 4.

Another approach is that we evenly distribute all neutrons around the center point (a vector which all elements are zero) like electron distribute around its neutron. Then we adjust distance from neutron to the center point but they lay on a line (ray) from last its position to the center point. (Not implement yet – I suppose that it is not good way).

Because every document is represented by an electron in this model, we will present technique to draw a sphere in screen and how to evenly distribute some electron on a sphere following section.

Draw a Sphere:

Use recursive function draw triangle with 20 triangles at start level and Ndiv is level of recursion.

We can also make a nice looking sphere by subdividing an icosahedron (complex 20-sided regular polyhedron, the more you subdivide it, the closer it gets to a smooth a sphere..). The code based on the code in the OpenGL red book (chapter 2, the example at the end)[63].

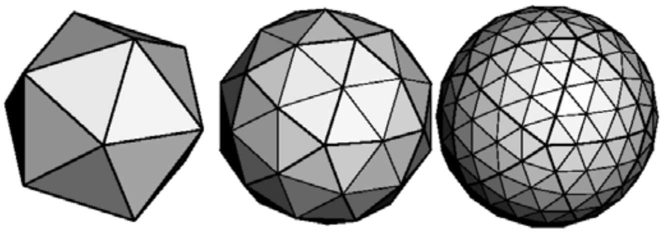


Figure 51: Subdividing to Improve a Polygonal Approximation to a Surface [63].

Evenly Distribution

This is a problem such as "How do I evenly distribute N points on a sphere?" and "How do I tessellate a sphere?"

I refer from http://www.math.niu.edu/~rusin/known-math/96/repulsion and http://www.mathworks.com/matlabcentral/newsreader/view_thread/21747.

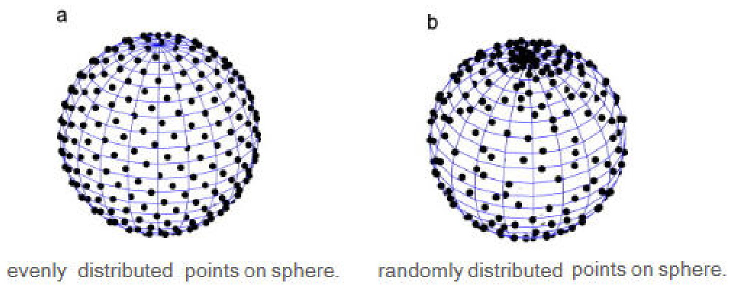


Figure 52: Evenly distributed points on sphere [from above link]

This is general algorithm to distribute N points "equally" about a unit sphere.

Given: N The number of points to distribute.

Each point is a vector have (x,y,z) and radius r .

Calculate the smallest linear distance between two neighboring points. If the function is run several times for the same N, r should not change by more than the convergence criteria, which is ±0.01 on a unit sphere. Demo value for distance is 0 or 1. If 1, the points are displayed over a unit sphere for each iteration. Demo defaults to 0 if only N is entered. Distributes N points about a unit sphere so that the straight line distance between neighboring points is roughly the same. The actual criterion for stopping is slightly different. The difference between a given point and every other point is stored in a matrix. The iterations stop once the maximum difference between any elements in successive distance matrices is less than 0.01. An absolute criteria was chosen due to self-distances being 0, and programming around this for relative convergence seemed too much work for too little reward.

The algorithm first generates N random points. Then a repulsive force vector, based on 1/r^2, is calculated for each point due to all of the other points. The resultant force vector for each point is normalized, and then each point is displaced a distance S = 1 in the direction of the force. Any value of S between 0.0 and 1. 0 seems to work with most values between 0.2 and 1 taking an average of 20 to 30 iterations to converge. If S is too high, too much "energy" is being added to the system and it won't converge. If S is too low, the points won't be evenly distributed even though the convergence criterion is met. Generally speaking, the larger N is the larger S can be without causing instabilities. After displacement, the point is projected back down onto the unit sphere.

When the system nears convergence, the displacement vector for a given point is nearly in the same direction as the radius vector for that point due to the points being equally distributed. A good check to make sure the code is working is to specify N = 4 and then check that the resulting points form a regular tetrahedron (or whatever it's called). How you would do this I don't know (check angles maybe). That's why I supplied the demo option so you could look at it in progress.[from above link]

Query

In data mining, especially in text mining, searching is an interesting procedure and it needs to implement in a Visual Clustering Analysis system. For example, after we cluster a set of document in K groups, now we want to find out some documents which contain some key words.

Thanks to representing document as vectors, we can view a query as a vector [54].

Let consider the query q = (visualization, cluster) from the tf-idf vectors of 3 documents,

term	Doc 1	Doc 2	Doc 3
computer	0.996	0.993	0.847
visualization	0.087	0.120	0.466
cluster	0.017	0	0.254

Now we turn the query into the unit vector:

q = (0, visualization, cluster)

$\vec{q} = (0,1,1) \rightarrow$ unit vector $\vec{q} = \left(0, \frac{1}{\sqrt{0^2+1^2+1^2}}, \frac{1}{\sqrt{0^2+1^2+1^2}} \right) = (0, 0.707, 0.707)$

The key idea now: to assign to each document d a score equal to the dot product

$\vec{v}(q).\vec{v}(d)$, then we can use the cosine similarity between the query vector and a document vector as a measure of score of the document for that query:

$$\text{Score}(q,d) = \frac{\vec{v}(q).\vec{v}(d)}{|\vec{v}(q)|\,|\vec{v}(d)|} \quad [54]$$

A document may have a high cosine score for a query even if it does not contain all query term.

	Doc 1	Doc 2	Doc 3
Score	0.074	0.085	0.509

The Doc 3 is the top-scoring document for the query q = (visualization, cluster). That mean Doc 3 is the most relative document for this query.

Interaction

We implement our system in 3D environment with full task for visualization such as overview, zoom, filter, Detail-on-demand, relative and extract [64]. Many interactive techniques proposed for interactive 3D rotation techniques involving Virtual Sphere, Arcball, 3D ball and Tracer [65]. However, Arcball which is an improvement Virtual Sphere is an appropriate technique for our system because we only interact to objects in 3D environment by mouse 3D, whereas 3D ball and Tracer need a special device.

Arcball and camera:

An arcball [66] is an interface for manipulating a 3D world in an intuitive way, and can be thought of as a virtual trackball. Imagine a virtual ball that is just behind the screen. By clicking it with mouse, we would pinch it, and by moving mouse we would make the ball spin around its center. And the same rotation would be applied to an object in the OpenGL scene! It allows us to freely rotate the world in 3D dimensions about a specified point. In other words, Arcball is an input technique for 3-D computer graphics, using a mouse to adjust the spatial orientation of an object.

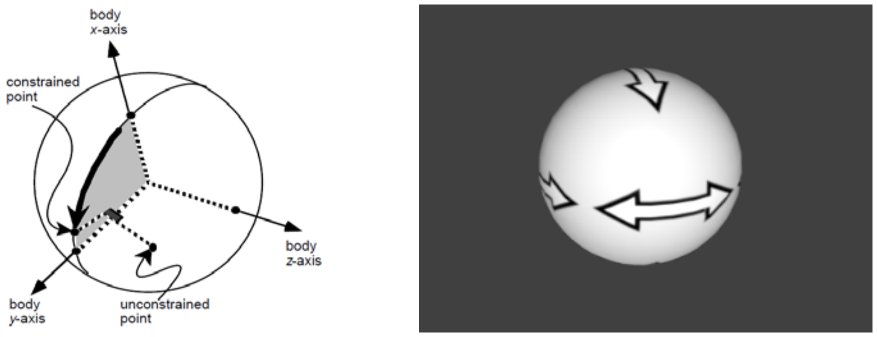


Figure 53: Rotation based on Arcball technique [66]

Result and Evaluation

Cluster validation which aim to assess the quality of clustering results plays the critical role of cluster analysis. For a specific application – document clustering, we use two effective clustering algorithms involving K-means and SOM for medium sized datasets, about 200-1000 articles.

We deal with two problems: how many clusters will be separated and high dimensional space.

The initial estimation of the number of cluster is very important in the first step of clustering algorithms.

Some previous research show that SOM can substitutes for k-means [44]. The first and most important is that SOM is less prone to local optima than k-means. On the other hand, k-means gradient orientation forces a premature convergence which, depending on the initialization, may frequently yield local optimum solutions. It is important to note that there are certain conditions that must be observed in order to render robust performances from SOM. Moreover, visualization is very powerful in showing cluster and exposing gaps in datasets (revealing trends). Atomic structure model in 3D space (and 3D-SOM) will support users an intuitive picture to understand the relationship among the datasets.

In our approach, we use both of them as support together, rather than try to prove what algorithm is better.

First of all, we generate a NxNxN grid which N is estimated by number of document.

Then we use K-means to separate the dataset to 3 groups. We find out a problem which is different between result of K-means and SOM clustering with the same dataset. This is not our problem. However, I concern it because we use SOM as intuitive map to estimate number of cluster for K-means. For example, we can see from SOM and estimate 3 groups can be separated, the 1st group has 16 elements, the second group has 6 elements, and the third group has 4 elements. But when we use k-means with 3 clusters, the first group has 6 elements, the second has 5, and the second group has 2 elements. Absolutely, SOM can substitute for k-means but they cannot prove that the result is the same. Our concern is that readers will compare 3D-SOM to Atomic structure model, results are usually not consistent.

To prove result’s difference from some clustering algorithm, we try to replace K-means by 1D-SOM, it means that using 3D-SOM to estimate number of cluster then use 1D-SOM to separate them to clusters (Picture).

To evaluation the system, we propose 3 criterions:

1. Visualization aspect
2. Performance
3. Validation

(1): The first most important criteria is that the arrangement of these objects within a drawing affects its understandability, usability, and aesthetics.

Our graph layout presents effectively relationship of individual objects in a dataset. In fact, we can see from below figure, they illustrate Euclidian distance between objects which is TFIDF vector and their centroid. In left figure, centroid’s positions are indicated by using Force-directed (FruchtermanReingold algorithms). In right figure, the positions are calculated based on Euclidian distance between centroid vector and number objects which belong to these clusters.

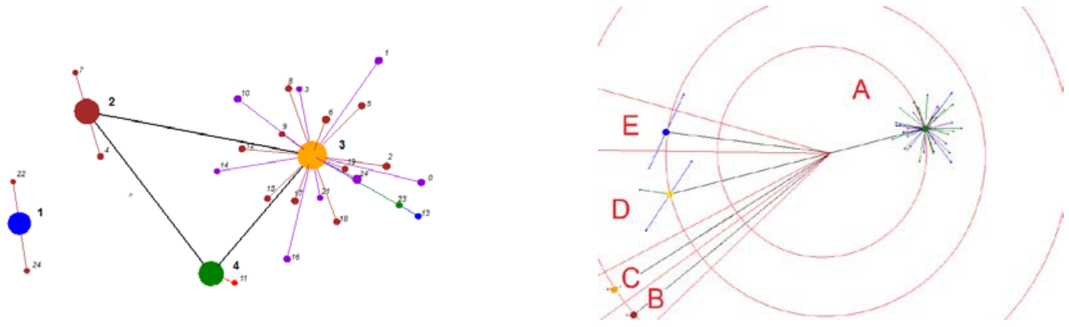


Figure 54: comparison of Force-directed layout and new layout

Both of them are show its understandability, usability. However, in aesthetics, the approach in the left is better. In contradiction, the right gives more information than another.

Indeed, we can compare Euclidian distances between neutrons (the centroid). However, we cannot know relative position of these centroids. If we want to know relationship of centroids in 3D, we can use 3D-SOM Neuron Network trained to classify them.

(2): Performance is one of some crucial requirements of a system. We consider some tasks which decide execution time of our system.

- Built vector space model from dataset: depends on number of documents and its words. We use hash table to process tokens and build bag of words or TFIDF vector. We can reduce execution time by increase stop words list and/or using Latent Semantic Indexing to decrease dimension of TFIDF vector. However, if we want to guarantee that the system will be accuracy, especially in searching query, we should remain original dimension (it not matter if the modern computer process Matrix sized 1000x10,000). Amortized time of accessing hash table is O(1). Therefore, time for Computing TF_IDF is about O(NumDoc x NumTerm)
- Build and training Neuron Network for 3D-SOM: the time really depends on number of neuron of Network, Number of Doc, Dimension and iterations (I). Recall, neuron's weight which is a vector has the same dimension with TFIDF vector. The overall complexity is typically O(I x NumDoc x NumTerm x $NumNeuron^3$).
- Training time for K-means: Most of the time is spent on computing vector distances. One such operation costs O(M) – where M is dimension of vector. The reassignment step computes K*N distances, so its overall complexity is O(KNM) . In the re-computation step, each vector gets added to a centroid once, so the complexity of this step is O(NM) . For a fixed number of iterations I, the overall

complexity is therefore O(IKNM) . Thus, that k-means is linear in all relevant factors: iterations (I), number of clusters (K), number of vectors (N) and dimensionality of the space (M).

- Draw sphere which represent for each element in dataset: recursively subdivides the triangles to the proper depth. If the depth value is 0, no subdivisions are performed, and the triangle is drawn as is. If the depth is 1, a single subdivision is performed, and so on. Time complexity for drawing a sphere is about T(n) = 4 T(n-1) + 1, equal O(4^n). Therefore, total time is NumDoc x O(4^n).
- Distribute evenly these electrons on a sphere (its engine level): O(I x N^2) where I is iteration and N is number points (objects) will be distributed in a sphere.
- Force-Directed performance (Fruchterman and Reingold): Each iteration the basic algorithm computes O(|E|) attractive forces and O(|V|) repulsive forces. Totally time is O(Ix|E|) + O(Ix|V|)

Performance	25 x 5052	50 x 8865	100x12348	200x19981
Build TFIDF	00:00:17.66	00:00:53.93	00:02:36.86	00:10:12.41
K-means	00:00:00.23	00:00:00.91	00:00:03.24	00:00:08.55
Force-directed	00:00:00.10	00:00:00.09	00:00:00.10	00:00:00.10
Training Neuron Network SOM	00:00:24.15	00:00:41.16	00:02:01.55	00:06:30.71
Draw all sphere	00:00:00.01	00:00:00.01	00:00:00.03	00:00:00.03
Distribute evenly	00:00:00.03	00:00:00.10	00:00:00.44	00:00:00.18

Table 1: Performance of our application (seconds) with 3 clusters for 25, 50, 100 and 200 documents (vectors), respectively 5052, 8865, 12348 and 19981 terms (dimension)

(3) As we mentioned above, there is a problem when we compare two famous clustering algorithms. However, in our work, we try to find out the new layout in 3D space to discovery dataset, rather than research a new clustering algorithm.

CHAPTER IV

SUMMARY AND CONCLUSION

This thesis has proposed a visual 3D graph layout called an atomic structure model for document cluster, to assist data miners in cluster analysis of high dimensional datasets.

Our approach employs quantified domain knowledge and explorative observation as prediction to map high dimensional data onto 3D space for revealing the relationship among documents. Data miners can perform an intuitive visual assessment of the consistency of the cluster structure. The model shows quantitatively data distribution in 3D environment. In this model, it also illustrates relationship among centroids of clusters by using force-directed algorithm and our algorithm based on their distances. Moreover, our approach can overcome the limit of space in compare with previous 2D approach by using arcball technique for interaction. In addition, our approach avoids arbitrary selection of k clusters by combination from SOM and K-mean. As a consequence, this model provides users a purposeful and effective visual method on cluster analysis.

Experiments show that our approach can improve the effectiveness of visualization for cluster analysis. It not only support quantified domain knowledge validation, but also utilizes the statistical measurements of the dataset. As a result, with advantages of this model, data miners can easily estimate the cluster number in the pre-processing stage of clustering and they have an effective guidance for having more precise cluster information in data mining, especially Document mining.

REFERENCES

[1] A. Jain, M. N. Murty and P. J. Flynn (1999). “Data Clustering: A Review”. ACM Computing Surveys, Vol. 31(3), pp. 264-323.

[2] Berkhin, P et al (2006). A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data, Springer Press pp. 25-72

[3] J. B. MacQueen (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, pp.281-297

[4] Kaufman, L. and Rousseeuw, P.J. (1987). Clustering by means of Medoids, in Statistical Data Analysis Based on the L_1–Norm and Related Methods. Edited by Y. Dodge, North-Holland, 405–416

[5] Peter J. Rousseeuw (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. Computational and Applied Mathematics 20: 53–65.

[6] J. Han and M. Kamber (2006). Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers

[7] Zhang, RAMAKRISHNAN, LIVNY (1997). BIRCH: A New Data Clustering Algorithm and Its Applications. Data Mining and Knowledge Discovery, 1, 141–182, © 1997 Kluwer Academic Publishers.

[8] Zhang T., Ramakrishnan R. and Livny M (1996). BIRCH: An efficient data clustering method for very large databases. In Proc. of SIGMOD96, Montreal, Canada , p103-114

[9] Guha S., Rastogi R., Shim K (1998). CURE: An efficient clustering algorithm for large databases. In Proc. of ACM SIGMOD Int'l Conf. on Management of Data, ACM Press, p73--84

[10] G. Karypis, E.-H. S Han, and V. Kumar (1999). Chameleon: hierarchical clustering using dynamic modeling. IEEE Computer, vol. 32(8), pp.68–75.

[11] M. Ester, H-P Kriegel, J. Sander, X. Xu (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), pp.226-231

[12] M. Ankerst, M. Breunig, H.-P. Kriegel, J. Sander "OPTICS: Ordering Points To Identify the Clustering Structure", Proceedings of ACM SIGMOD '99, International Conference on Management of Data, Philadelphia, PA. pp. 49-60 (1999).

[13] W. Wang, J. Yang, and R. Muntz. (1997). STING: A statistical information grid approach to spatial data mining. Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB97), pp.186-195.

[14] G. Sheikholeslami, S. Chatterjee, A. Zhang (1998). Wavecluster: A multi-resolution clustering approach for very large spatial databases. Proceedings of Very Large Databases Conference (VLDB98), pp.428-439

[15] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan (1998). Automatic subspace clustering of high dimensional data for data mining applications. Proceedings of the ACM SIGMOD Conference, Seattle, WA., pp.94-105.

[16] D. Fisher (1987). Improving Inference through Conceptual Clustering. Proceedings of 1987 AAAI Conferences, Seattle Washington, pp.461-465

[17] B. Shneiderman (2001). Inventing Discovery Tools: Combining Information Visualization with Data Mining. Proceedings of Discovery Science 2001,Lecture Notes in Computer Science Vol 2226, pp.17-28

[18] S. K. Card, J. D. Mackinlay, and B. Shneiderman (1999) . Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco.

[19] E. Pampalk, W. Goebl, and G. Widmer (2003). Visualizing Changes in the Structure of Data for Exploratory Feature Selection. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD '03), August 24-27, 2003, Washington, DC, USA pp.157-166

[20] A. Hinneburg, D. A. Keim., M, Wawryniuk (1999). HD-Eye:Visual Mining of High-Dimensional Data. IEEE Computer Graphics and Applications, Volume 19, Issue 5 (September 1999), pp.22-31

[21] R.M. Rohrer, J.L. Sibert, D.S. Ebert (1999). Shape-based Visual Interface for Text Retrieval. IEEE Computer Graphics and Applications, Vol. 19(5).

[22] T.C. Sprenger, R. Brunella, M. H.Gross (2000). H-BLOB: a hierarchical visual clustering method using implicit surfaces. Proceedings of Visualization 2000, pp. 61-68

[23] A. Morrison, G. Ross and M. Chalmers (2002). Combining and comparing clustering and layout algorithms. University of Glasgow

[24] Colin Ware (2004). Information Visualization – Perception for design. Second edition, Morgan Kaufman

[25] Riccardo Mazza (2009). Introduction to Information Visualization. © Springer-Verlag London Limited

[26] A. Inselberg (1985). The Plane with Parallel Coordinates, Special Issue on Computational Geometry. The Computer, Vol. 1, pp. 69-97

[27] D. F. Andrews (1972). Plots of High-Dimensional Data. Biometrics, Vol. 29, pp. 125-136

[28] J.H. Siegel, R. M. Goldwyn and H. P. Friedman (1971). Irregular polygon to represent multivariate data (with vertices of equal intervals, distanced from the centre proportionally to the value of the variable). USA (1971) October

[29] H. Chernoff (1973). The Use of Faces to Represent Points in k-Dimensional Space Graphically. Journal Amer. Statistical Association, Vol. 68, pp.361-368

[30]R. M. Pickett (1970). Visual Analyses of Texture in the Detection and Recognition of Objects. Picture Processing and Psycho-Pictorics, Lipkin B. S., Rosenfeld A. (eds.), Academic Press, New York

[31] D. A. Keim and H.-P. Kriegel (1994). VisDB: Database Exploration Using Multidimensional Visualization. IEEE Computer Graphics and Applications, 14(5) pp. 40-49

[32] D. A. Keim (1996). Pixel-oriented visualization techniques for exploring very large databases. Computational and Graphical Statistics, 5(1):58-77.

[33] M. Ankerst, D. A. Keim, H.-P. Kriegel (1996). Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets. Proceedings of Visualization '96, Hot Topic Session, San Francisco, CA.

[34] Daniel A. Keim, Hans-Peter Kriegel (1996). Visualization Techniques for Mining Large Databases: A Comparison", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6.

[35] J. LeBlanc, M. O. Ward, N. Wittels (1990). Exploring N-Dimensional Databases. Proc. Visualization '90, San Francisco, CA, pp. 230-237.

[36] G. Robertson, S. Card, J. Mackinlay (1991). Cone Trees: Animated 3D Visualizations of Hierarchical Information. Proc. ACM CHI Int. Conf. on Human Factors in Computing, pp. 189-194.

[37] B. Shneiderman (1992). Tree Visualization with Treemaps: A 2D Space-Filling Approach. ACM Transactions on Graphics, Vol. 11, No. 1, pp. 92-99.

[38] F. Oliveira, H. Levkowitz (2003). From Visual Data Exploration to Visual Data Mining: A Survey. IEEE Trans.Vis.Comput. Graph, Volume 9(3), pp.378-394

[39] D. A. Keim (2002). Information Visualization and Data Mining. IEEE Transactions on Visualization and Computer Graphics, Vol. 7(1), January-March 2002, pp.100-107

[40] D. Keim, M. Ward (2002). Visual Data Mining Techniques. First public in Intelligent Data Analysis: An Introduction / Michael Berthold ... (eds.). Berlin: Springer

[41] D. Keim (2001). Visual exploration of large databases. Communications of the ACM, 44(8):38–44.

[42] L. Yan (2000). Interactive exploration of very large relational data sets through 3d dynamic projections. SIGKDD Int. Conf. On Knowledge Discovery & Data Mining (KDD 2000), Boston, MA.

[43] Eser Kandogan (2000). Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. Proceedings of IEEE information Visualization Symposium (Hot Topics), pages 4–8.

[44] T. Kohonen (1997). Self-Organizing Maps. Springer, Berlin, second extended edition.

[45] S. Kaski, J. Sinkkonen. and J. Peltonen (2001). Data Visualization and Analysis with Self-Organizing Maps in Learning Metrics. DaWaK 2001, LNCS 2114, pp.162-173.

[46] Juha Vesanto (1999). SOM−Based Data Visualization Methods", In Intelligent Data Analysis, Volume 3, Number 2, Elsevier Science, pp. 111−126.

[47] K. Lagus, T. Honkela, S. Kaski, T. Kohonen (1996). Self-organizing maps of document collections: A new approach to interactive exploration. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, California, pp. 238–243.

[48] Ke-Bing Zhang, Mehmet A. Orgun, Kang Zhang (2006). HOV3: An Approach for Visual Cluster Analysis. In Proceedings of The 2nd International Conference on Advanced Data Mining and Applications. (ADMA 2006), XiAn, China, Aug. 14-16, 2006. Lecture Notes in Computer Science series, Vol. LNAI 4093, pp.316-327 Springer Press.

[49] Andrew Vande Moere (2008). A Model for Self-Organizing Data Visualization Using Decentralized Multi-agent Systems. Advances in Applied Self-organizing Systems - Advanced Information and Knowledge Processing, *Springer Press* pp 291-324.

[50] Reynolds, C. W, (July 1987). Flocks, herds, and schools: A distributed behavioral model. *SIGGRAPH* Computer Graphics 21(4):25–34.

[51] Xiaohui Cui and Thomas E. Potok, (2006). A Distributed Agent Implementation of Multiple Species - Flocking Model for Document Partitioning Clustering. Proceeding CIA'06 Proceedings of the 10th international conference on Cooperative Information Agents, Pages 124-137

[52] Jeffrey Solka, (2008). Text Data Mining: Theory and Methods. *Statistical Surveys* **2,** 94—112.

[53] Porter, M. F. (1980). Algorithm for suffix stripping. Program, 130–137.

[54] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, (2009). An Introduction to Information Retrieval. *Cambridge University Press*

[55] Matthew Ward, Georges Grinstein, Daniel Keim, (2010). Interaction Data Visualization – Foundation, Technique, and Applications. Copyright ® by A K Peter, Ltd.

[56] Deerwester, S., Dumais, S. T., Furnas, G. W., and Landauer, T. K. (1990). Indexing by latent semantic analysis. Journal of the Am. Soc. for Information Science 41, 6, 391–407.

[57] Pavel Berkhin, (2006). A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data, Springer Press pp. 25-72

[58] Jain A., Murty M. N., and Flynn P.J (1999). Data Clustering: A Review. ACM Computing Surveys Volume: 31(3), 264-323.

[59] Anil K. Jain, (June, 2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, v.31 n.8, p.651-666.

[60] Fernando Bação , Victor Lobo , Marco Painho, (May 22-25, 2005). Self-organizing maps as substitutes for k-means clustering. Proceedings of the 5th international conference on Computational Science, Atlanta, GA.

[61] T. Fruchterman and E. Reingold (1991). Graph drawing by force-directed placement. Softw. – Pract. Exp., 21(11):1129–1164.

[62] Chris Walshaw, (2000). A Multilevel Algorithm for Force-Directed Graph Drawing. Proceeding GD '00 Proceedings of the 8th International Symposium on Graph Drawing Pages 171 – 182.

[63] Dave Shreiner, (2010). OpenGL® Programming Guide Seventh Edition - The Official Guide to Learning OpenGL®, Versions 3.0 and 3.1, Copyright © 2010 Pearson Education, Inc. ISBN 13: 978-0-321-55262-4

[64] Shneiderman, Plaisant, (2010). Designing the user interaction interface: strategies for effective Human-Computer Interaction. 5th edition, Addison Wesley, ISBN 13: 978-0-321-53735-5

[65] Hinckley, K., Tulio, J., Pausch, R., Proffitt, D, Kassell, N.,. (1997). Usability Analysis of 3D Rotation Techniques. In Proceedings of the 10th annual ACM symposium on User interface software and technology, pp. 1-10.

[66] Shoemake (1992). ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse", Proceedings of Graphics Interface'92, pp. 151-156.

APPENDIX

APPENDIX A

INTRODUCTION TO OPENTK

From The Open Toolkit Manual:

The Open Toolkit is a free project that allows you to use OpenGL, penGL|ES, OpenCL and OpenAL APIs from managed languages.

OpenTK started life as an experimental fork of the Tao framework before during the summer of 2006. It is original intention was to provide a cleaner wrapper than Tao.OpenGL, but it quickly grew in focus: right now, it provides access to various Khronos and Creative APIs and handles the necessary initialization logic for each API. As such, the Open Toolkit is most similar to projects like Tao, SlimDX, SDL or GLFW.

Unlike similar libraries, OpenTK attempts provide a consistent interface that utilizes the superior managed runtime. Instead of untyped pointers, OpenTK provides generics. Instead of plain constants, OpenTK uses strongly-typed enumerations. Instead of plain function lists, OpenTK separates functions per extension category. A common math library is integrated and directly usable by each API.

Features:

- Written in cross-platform C# and usable by all managed languages (F#, Boo, VB.Net, C++/CLI).
- Consistent, strongly-typed bindings, suitable for RAD development.

- Usable stand-alone or integrated with Windows.Forms, GTK#, WPF.
- Cross-platform binaries that is portable on .Net and Mono without recompilation.
- Wide platform support: Windows, Linux, Mac OS X, with iPhone port in progress.

The Open Toolkit is suitable for games, scientific visualizations and all kinds of software that requires advanced graphics, audio or compute capabilities. Its license makes it suitable for both free and commercial applications.

BIOGRAPHICAL SKETCH

Khanh Vinh Nghi is a faculty in Tra Vinh University (TVU), Vietnam. He got a Bachelor of degree in Computer Engineer from Ho Chi Minh City University of Technology (belong to Vietnam National University) in 2002. Before joined to TVU, he worked as programmer in Control.ltd (Ho Chi Minh City) which develops interactive system between computer and device for two years. In 2004, he worked at TVU and managed NIIT Travinh Center, which operate between NIIT institute (India) and TVU. At that time, he had taught many subjects such as Microsoft SQL, Visual Studio .Net, Network Administrator, Digital System, Computer Architect. He had been recognized as Excellence of Award for Center Head in Vietnam Team – from NIIT India, 2007. In addition to serving actively in TVU activities, he was Chair of Travinh Alumna Union in HoChiMinh City, Vietnam (2006-2008). In 2011, he got a full scholarship from Travinh – 100 project for study abroad and became a graduate student in Computer Science in The University of Texas – Pan American from 2011 to 2013. Besides studying at UTPA, he currently served as teaching assistant to help undergrad students in “Introduction to Computer Science” class. His Research Interests involve Human Computer Interaction, Information Visualization, Computer Architecture, Embedded System, Robotic Control and Artificial Intelligence. An important aspect of his life is family -- his wife (another academic in linguistics and Teaching English as a Second Language in TVU), and two daughters. After he got Master degree in Computer Science from UTPA, he came back his country and lives in a cozy house in 112 A1 Tran Phu Street, Ward 2, Tra Vinh City, Vietnam.