P-3612

DYNAMIC AND ADAPTIVE SELF ORGANIZING MAPS AND PARTICLE SWARM OPTIMIZATION FOR TEMPORAL TREND DETECTION

PROJECT REPORT

Submitted by

J.SATHISH (0710108047)

V.SIDDHARTH (0710108050)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology Coimbatore)

COIMBATORE 641049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY: COIMBATORE-641 049

BONAFIDE CERTIFICATE

Certified that this project report entitled **"DYNAMIC AND ADAPTIVE SELF ORGANIZING MAPS AND PARTICLE SWARM OPTIMIZATION FOR TEMPORAL TREND DETECTION"** is the bonafide work of **J.SATHISH and V.SIDDHARTH** who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs.P.Devaki.,M.E.,(Ph.D)

HEAD OF THE DEPARTMENT

Department of Computer

Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049

SIGNATURE

Mrs.D.Chandrakala M.E.,(Ph.D)

COURSE CO-ORDINATOR

Associate Professor

Department of Computer

Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049

The candidate with University Register Nos. 0710108047 and 0710108050 were examined by us in the project viva-voce examination held on 20.4.11

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled **"DYNAMIC AND ADAPTIVE SELF ORGANIZING MAPS AND PARTICLE SWARM OPTIMIZATION FOR TEMPORAL TREND DETECTION"** is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Coimbatore.

Place: Coimbatore

Date: 20.4.11

(J.SATHISH)

V.Siddharth

(V.SIDDHARTH)

ACKNOWLEDGEMENT

We are intend to express our heartiest thanks to our chairman **Arutselvar Dr.N.Mahalingam ,B.sc., F.I.E.,** co-chairman **Dr. B.K. Krishnaraj Vanavarayar, B.Com., B.L.,** and the correspondent **M.Balasubramaniam, M.com., M.B.A.**, for given us this opportunity to embark on this project.

We extend our sincere thanks to our Principal, **Dr. S.Ramachandran Ph.D.,** Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We would like to make a special acknowledgement and thanks to **Dr. S. Thangasamy Ph.D.,** Dean of Research and Development, for his support and encouragement throughout the project.

We are indent to express my heartiest thanks to **Mrs.P.Devaki M.E,(Ph.D)**, *Project coordinator* ,Head of the Department of Computer Science &Engineering, for her valuable guidance and useful suggestions during the course of this project.

We express deep gratitude and gratefulness to **our Guide**

Mrs.D.Chandrakala M.E., *Associate Professor* Department of Computer Science & Engineering, for her supervision, enduring patience, active involvement and guidance.

We would like to convey our honest thanks to **all Faculty members** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

We also thank our **friends and family** who helped us to complete this project fruitfully.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1. Introduction

1.1 Introduction to Data Mining

Data mining, *the extraction of hidden predictive information from large databases*, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

1.1.1Data, Information and Knowledge

Data

The term data refers to qualitative or quantitative attributes of a variable or set of variables. Data (plural of "datum") are typically the results of measurements and can be the basis of graphs, images, or observations of a set of variables. Data are often viewed as the lowest level of abstraction from which information and then knowledge are derived. Raw data, i.e. unprocessed data, refers to a collection of numbers, characters, images or other outputs from devices that collect information to convert physical quantities into symbols.

Information

Information in its most restricted technical sense is an ordered sequence of symbols that record or transmit a message. It can be recorded as signs, or conveyed as signals by waves. Information is any kind of event that affects the state of a dynamic

system. As a concept, however, information has numerous meanings. Moreover, the concept of information is closely related to notions of constraint, communication, control, data, form, instruction, knowledge, meaning, mental stimulus, pattern, perception, representation, and even entropy.

Knowledge

Knowledge is a detailed familiarity with, or understanding of, a person, thing or situation. It can include facts and information, as well as understanding that is gained through experience, education or reason. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); and it can be more or less formal or systematic.In philosophy, the study of knowledge is called epistemology, and the philosopher Plato famously defined knowledge as "justified true belief." There is however no single agreed upon definition of knowledge, and there are numerous theories to explain it.

1.1.2 Data Warehouses

A data warehouse (DW) is a database used for reporting. The data is offloaded from the operational systems for reporting. The data may pass through an operational data store for additional operations before it is used in the DW for reporting. A data warehouse maintains its functions in three layers: staging, integration, and access. Staging is used to store raw data for use by developers (analysis and support). The integration layer is used to integrate data and to have a level of abstraction from users. The access layer is for getting data out for users.

This definition of the data warehouse focuses on data storage. The main source of the data is cleaned, transformed, catalogued and made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support (Marakas & OBrien 2009). However, the means to

retrieve and analyze data, to extract, transform and load data, and to manage the data dictionary are also considered essential components of a data warehousing system. Many references to data warehousing use this broader context. Thus, an expanded definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.

1.1.3 What can data mining do?

Because it can improve customer service, better target marketing campaigns, identify high-risk clients, and improve production processes. In short, because it can help you or your company make or save money. Most businesses and organizations collect data about their operations. They then examine this data for insights into their operations and into the transactions their business performs. This may be as simple as a cursory glance at a spreadsheet, to check the numbers for "sanity". Or it may involve graphical analysis with a full-blown OLAP tool.

However, manual analysis typically has to stop at this point — looking for obvious simple relationships. An automated data mining approach, however, can often find profitable relationships you may not even have suspected existed, or that you knew would take too long to find by manual means. Automated data mining allows the development of models of considerable complexity which can take many more factors into account.

1.1.4 How does data Mining Work?

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Generally, any of four types of relationships are sought:

* Classes: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.

* Clusters: Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.

* Associations: Data can be mined to identify associations. The beer-diaper example is an example of associative mining.

* Sequential patterns: Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Data mining consists of five major elements:

* Extract, transform, and load transaction data onto the data warehouse system.
* Store and manage the data in a multidimensional database system.
* Provide data access to business analysts and information technology professionals.
* Analyze the data by application software.
* Present the data in a useful format, such as a graph or table.

1.2 Data Mining Techniques:

Pre-processing

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns already present in the data, the target dataset

must be large enough to contain these patterns while remaining concise enough to be mined in an acceptable timeframe. A common source for data is a data mart or data warehouse. Pre-process is essential to analyze the multivariate datasets before data mining. The target set is then cleaned. Data cleaning removes the observations with noise and missing data.

Data mining

Data mining commonly involves four classes of tasks:

* Clustering – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.

* Classification – is the task of generalizing known structure to apply to new data. For example, an email program might attempt to classify an email as legitimate or spam. Common algorithms include decision tree learning, nearest neighbor, naive Bayesian classification, neural networks and support vector machines.

* Regression – Attempts to find a function which models the data with the least error.

* Association rule learning – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.

Results validation

The final step of knowledge discovery from data is to verify the patterns produced by the data mining algorithms occur in the wider data set. Not all patterns found by the data mining algorithms are necessarily valid. It is common for the data mining algorithms to find patterns in the training set which are not present in the general data set, this is called over fitting. To overcome this, the evaluation uses a test set of data which the data mining algorithm was not trained on. The learnt patterns are

applied to this test set and the resulting output is compared to the desired output. For example, a data mining algorithm trying to distinguish spam from legitimate emails would be trained on a training set of sample emails. Once trained, the learnt patterns would be applied to the test set of emails which it had not been trained on; the accuracy of these patterns can then be measured from how many emails they correctly classify.

If the learnt patterns do not meet the desired standards, then it is necessary to reevaluate and change the pre-processing and data mining. If the learnt patterns do meet the desired standards then the final step is to interpret the learnt patterns and turn them into knowledge.

1.3 Introduction to Neural Networks

An artificial neural network (ANN), usually called neural network (NN), is a mathematical model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data.

Learning paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning

In supervised learning, we are given a set of example pairs and the aim is to find a function in the allowed class of functions that matches the examples. In other words,

we wish to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error, which tries to minimize the average squared error between the network's output, f(x), and the target value y over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called multilayer perceptrons, one obtains the common and well-known back propagation algorithm for training neural networks.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a "teacher," in the form of a function that provides continuous feedback on the quality of solutions obtained thus far. Basically supervised learning are classified in two types. These are error connection gradient descent and stochastic. Error connection gradient descent are also classified into least mean square and back propagation.

Unsupervised learning

In unsupervised learning, some data is given and the cost function to be minimized, that can be any function of the data and the network's output.

The cost function is dependent on the task (what we are trying to model) and our a priori assumptions (the implicit properties of our model, its parameters and the observed variables).As a trivial example, consider the model , where is a constant and the cost . Minimizing this cost will give us a value of that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: for example, in compression it could be related to the mutual information between x and y, whereas in statistical modeling, it could be related to the posterior probability of the model given the data. (Note that in both of those examples those

quantities would be maximized rather than minimized).Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

Reinforcement learning

In reinforcement learning, data are usually not given, but generated by an agent's interactions with the environment. At each point in time, the agent performs an action and the environment generates an observation and an instantaneous cost, according to some (usually unknown) dynamics. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost; i.e., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally, the environment is modeled as a Markov decision process (MDP) with states and actions with the following probability distributions: the instantaneous cost distribution, the observation distribution and the transition, while a policy is defined as conditional distribution over actions given the observations. Taken together, the two define a Markov chain (MC). The aim is to discover the policy that minimizes the cost; i.e., the MC for which the cost is minimal. Ann's are frequently used in reinforcement learning as part of the overall algorithm. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Recent

developments in this field use particle swarm optimization and other swarm intelligence techniques.

Most of the algorithms used in training artificial neural networks employ some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods simulated annealing, expectation-maximization and non-parametric methods are some commonly used methods for training neural networks.

Temporal perceptual learning relies on finding temporal relationships in sensory signal streams. In an environment, statistically salient temporal correlations can be found by monitoring the arrival times of sensory signals. This is done by the perceptual network.

1.4 Self Organizing Maps

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.

Like most artificial neural networks, SOM's operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. Mapping automatically classifies a new input vector.

A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The

procedure for placing a vector from data space onto the map is to find the node with the closest weight vector to the vector taken from data space and to assign the map coordinates of this node to our vector.

While it is typical to consider this type of network structure as related to feed forward networks where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation. It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to K-means, larger self-organizing maps rearrange data in a way that is fundamentally topological in character.

It is also common to use the U-Matrix. The U-Matrix value of a particular node is the average distance between the node and its closest neighbors. In a square grid for instance, we might consider the closest 4 or 8 nodes, or six nodes in a hexagonal grid. Large SOM's display properties which are emergent. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.

Learning algorithm

The goal of learning in the self-organizing map is to cause different parts of the network to respond similarly to certain input patterns. This is partly motivated by how visual, auditory or other sensory information is handled in separate parts of the cerebral cortex in the human brain.

The weights of the neurons are initialized either to small random values or sampled evenly from the subspace spanned by the two largest principal component eigenvectors. With the latter alternative, learning is much faster because the initial weights already give good approximation of SOM weights.The network must be fed a large number of example vectors that represent, as close as possible, the kinds of vectors expected during mapping. The examples are usually administered several times as iterations.

The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron with weight vector most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU. The update formula for a neuron with weight vector Wv(t) is

$$Wv(t + 1) = Wv(t) + \Theta (v, t)\, \alpha(t)(D(t) - Wv(t)),$$

Where $\alpha(t)$ is a monotonically decreasing learning coefficient and D(t) is the input vector. The neighborhood function Θ (v, t) depends on the lattice distance between the BMU and neuron v. In the simplest form it is one for all neurons close enough to BMU and zero for others, but a Gaussian function is a common choice, too. Regardless of the functional form, the neighborhood function shrinks with time. At the beginning when the neighborhood is broad, the self-organizing takes place on the global scale. When the neighborhood has shrunk to just a couple of neurons the weights are converging to local estimates.

This process is repeated for each input vector for a (usually large) number of cycles λ. The network winds up associating output nodes with groups or patterns in the input data set. If these patterns can be named, the names can be attached to the associated nodes in the trained net.During mapping, there will be one single winning neuron: the neuron whose weight vector lies closest to the input vector. This can be simply determined by calculating the Euclidean distance between input vector and weight vector.

While representing input data as vectors has been emphasized in this article, it should be noted that any kind of object which can be represented digitally and which has an appropriate distance measure associated with it and in which the necessary operations for training are possible can be used to construct a self-organizing map. This includes matrices, continuous functions or even other self-organizing maps.

1.5 Particle Swarm Optimization

In computer science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position and it's also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is originally attributed to Kennedy, Eberhart and Shi and was first intended for simulating social behavior, as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. The book by Kennedy and Eberhart describes many philosophical aspects of PSO and swarm intelligence. An extensive survey of PSO applications is made by Poli.

There will be about 300 particles, with one dimension each. The particles search space is confined between 0 and 1.The particles can move anywhere around 0 to 1. The particles movement are governed by an Global Optimum. The iteration is stopped when Global Optimum is reached. Finally, each position of the particle in the search space is taken as the weight vector.

SYSTEM REQUIREMENTS

Hardware Configuration:

Processor Name	: Intel Core 2 Duo
Processor Speed	: 2 GHz
Memory (RAM)	: 3 GB
Hard Disk	: 320 GB

Software Configuration:

Operating System	-	Windows XP and above
Software Tools	-	Microsoft Visual studio 2008 (C#.Net)
Datastore	-	XML

Software Description:

Microsoft Visual studio 2008 (C#.Net):

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native codes together with managed code for all platforms supported by Microsoft Windows.

C# is a multi-paradigm programming language encompassing imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative C# is one of the programming languages designed for the Common Language Infrastructure. C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

CHAPTER 2
PROBLEM DEFINITION

2.1 Problem Definition:

The ACM,KDD, KD, PKDD, PAKDD, ICDM are the most famous conferences. In a given year, lot of proceedings will happen in a conference on behalf of different titles. For that year, to find the Emergent Trend manually is practically impossible because of large size of datasets. So initially we remove the stop words from all the titles and convert them to compound nouns which might represent a trend. Then we calculate three parameters namely Tf-Idf , Jaccard , Odds based on the formulae. The terms along with these parameters are fed as input to our Dynamic and Adaptive Self Organizing Maps(DASOM) and after providing the neural network with sufficient amount of training, two clusters are formed, Emergent and Subsiding.

In order to optimize the clusters i.e., classify the terms accurately, a Global Stochastic optimization technique namely, Particle Swarm Optimization (PSO) is used. It actually Optimizes the neural network training by providing them with the random weight vectors. Emergent cluster is a cluster of terms which are predicted as the Emergent Trends and similarly terms in the subsiding cluster represent the trends to be of Subsiding nature.

Regression analysis includes any techniques for modelling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. More specifically, regression analysis helps one understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. In all cases, the estimation target is a function of the independent variables called the regression function. Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of

these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables.

The self organizing maps(SOM) has been used as a tool for mapping high-dimensional input data into a low-dimensional feature map, which has significant advantages for text clustering applications. In this paper, a novel dynamic and adaptive SOM algorithm applied to high dimensional large scale text clustering is proposed. The characteristic feature of this novel neural network model is its dynamic architecture which grows during its training process to find the inherent topology structure of the document set. By using unsupervised competitive learning in network, the weight vectors of the winning node and its nearest neighbors are adjusted adaptively (where learning rate is related to similarity in amended learning rule) in this algorithm. The results of the experiments indicated that the algorithm successfully improve quality of text clustering and learning speed of neural network.

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position and it's also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

2.2 Literature Survey:

2.2.1 Dynamic and Adaptive Self Organizing Maps applied To High Dimensional Large Scale Text Clustering

The dynamic and adaptive self organizing maps (DASOM) is a model which has characteristics of a dynamic network and adaptive learning speed. The weight vectors in the network are initialized by selecting text vectors randomly from the input space. In the training phase, cosine similarities of input pattern (text vector) with the weight vectors are computed and select the largest to find the best match (winning) node. The weight vectors of the nodes that lie inside the topological neighborhood of winning node are adjusted by using the amended rule related to similarity. When the similarity is smaller than a given threshold, a new node is inserted into the network and the input pattern is assigned to weight vector of the new node. After the network becomes stable, the similar nodes represented similar classes are combined for optimization.

2.2.2 Detection of Trends of Technical Phrases in Text Mining

In text mining processes, the importance indices of the technical terms play a key role in finding valuable patterns from various documents. Further, methods for finding

emergent terms have attracted considerable attention as an important issue called temporal text mining. However, many conventional methods are not robust against changes in technical terms. In order to detect remarkable temporal trends of technical terms in given textual datasets robustly, we propose a method based on temporal changes in several importance indices by assuming the importance indices of the terms to be a dataset. The method consists of an automatic term extraction method in given documents, three importance indices from text mining studies, and temporal trends detection based on results of linear regression analysis. Empirical studies show that the three importance indices are applied to the titles of four annual conferences KDD,KD,PKDD,ACM about data mining field as sets of documents. After detecting the temporal trends of automatically extracted phrases, we compared the trends of the technical phrases among the titles of the annual conferences.

2.2.3 Self-Organizing Feature Map Based Data Mining

In general, data mining has three major components; they are clustering or classification, association rules mining, and sequential pattern analysis. In classification a set of grouping rules which can be used to classify data in the future are generated. Using SOM as an intermediate step makes the clustering a two-level approach: the data set is first clustered using SOM, and then the SOM is clustered. One big benefit of this approach is that computational load decreases. For this reason, it is convenient to cluster a set of prototypes rather than directly the data.

Association rules mining is to mine some associated relationships among a set of objects in a database, it requires iterative scanning of large relational database which is time consuming in processing. An association rules mining algorithm Apriori has been developed for rule mining in large transaction databases by Combination of this algorithm and SOM neural networks will speed up the Mining. In sequential pattern analysis, the database is explored to discover patterns that occur in sequence. This deals with data that appear in separate transactions. For example, if a customer buys item A in the first month of the year, then she buys item B in the second month and item C in the third month etc.

2.2.4 Analysis of particle interaction in particle swarm optimization

Particle swarm optimization (PSO), introduced by Kennedy and Eberhart in 1995, was proposed based on an inspiration from the social behavior of insects or animals that the exchanging and sharing of information among a group of individuals benefit the group survival by improving the group capability of foraging. In the framework of PSO, the insects or animals are considered as particles flying through the multi-dimensional search space and searching for the optimal position. The movement of particles is affected by three factors: the inertia, personal experience (the cognitive part), and particle interaction (the social part).

Since its introduction, PSO has been empirically shown to be a very useful and effective optimization framework for the easiness to implement and flexibility to use. Although PSO is widely applied in many research fields nowadays, the theoretical analysis on PSO is still quite limited. To the best of our knowledge, the first analysis was proposed by Kennedy Particle trajectories for design choices were shown. Ozcan and Mohan assumed fixed attractors and constant coefficients to demonstrate the particle trajectory as a sinusoidal wave. With similar assumptions, Maurice and Kennedy simplified PSO to a deterministic dynamical system and analyzed its stability. Such simplified, deterministic versions of PSO or similar systems, employing a single particle, fixed attractors, or constant coefficients, were analyzed by many researchers for stability, convergence, and parameter selection Kadirkamanathan et al. and Jian et al. started to consider the randomness in acceleration coefficients, but attractors were still fixed. Away from the common PSO configuration, Emara and Fattah as well as Gazi and Passino analyzed PSO in a continuous time setting.

Most of the existing studies do not provide analysis on the facet of particle interaction, which is definitely an essential mechanism of PSO. In this paper, under more practical assumptions, including multiple particles, unfixed attractors, and stochastic acceleration coefficients, we make the first attempt to analyze the effect of particle interaction. In particular, we consider the PSO system from a macro state viewpoint, analyze the swarm behavior, and obtain theoretical results on the progress rate as well as the convergence criterion.

2.3 Dblp Dataset Description:

DBLP (Digital Bibliography & Library Project) is a computer science bibliography website hosted at Universität Trier, in Germany. It was originally a database and logic programming bibliography site, and has existed at least since the 1980s. DBLP listed more than 1.3 million articles on computer science in January 2010. Journals tracked on this site include .VLDB, a journal for very large databases, the IEEE Transactions and the ACM Transactions. Conference proceedings papers are also tracked. It is mirrored at five sites across the Internet.

DBL-Browser (Digital Bibliographic Library Browser) is a utility for browsing the DBLP website. The browser was written by Alexander Weber in 2005 at Universität Trier. It is designed for use off-line in reading the DBLP, which consists of 910,000 bibliographic entries, as of July 2007.DBL-Browser is GPL software, available for download from SourceForge. It uses the XML DTD. Written in Java programming language, this code shows the bibliographic entry in several types of screens, ranging from graphics to text:

* Author page
* Article page
* Table of contents
* Related conferences / journals
* Related authors (graphic representation of relationships)
* Trend analysis (graphics histogram)

```
<?xml version="1.0" encoding="utf-8" ?>
<dblp>
 <incollection mdate="2010-01-15" key="conf/ACMace/Grace09a">
    <author>Lindsay Grace</author>
    <title>Critical gameplay.</title>
    <pages>444</pages>
    <year>2010</year>
    <booktitle>Advances in Computer Entertainment Technology</booktitle>
    <ee>http://doi.acm.org/10.1145/1690388.1690492</ee>
    <crossref>conf/ACMace/2009</crossref>
    <url>db/conf/ACMace/ace2009.html#Grace09a</url>
  </incollection>
 <incollection mdate="2010-01-15" key="conf/ACMace/KuribayashiKT09">
    <author>Satoshi Kuribayashi</author>
    <author>Takaki Kimura</author>
    <author>Hiroya Tanaka</author>
    <title>Plant feeling light: a lighting system working with plant biorhythms.</title>
    <pages>458</pages>
    <year>2010</year>
    <booktitle>Advances in Computer Entertainment Technology</booktitle>
    <ee>http://doi.acm.org/10.1145/1690388.1690504</ee>
    <crossref>conf/ACMace/2009</crossref>
    <url>db/conf/ACMace/ace2009.html#KuribayashiKT09</url>
  </incollection>
 <incollection mdate="2010-01-15" key="conf/ACMace/BihlerFH09">
    <author>Pascal Bihler</author>
    <author>Ronald Fromm</author>
    <author>Katja Henke</author>
    <title>Mister X: an innovative location-based multiplayer game.</title>
```

Figure 1: Example Dataset

2.4 OBJECTIVES

The objectives of the system are,

- To preprocess the dataset to make the mining process easier.
- To determine the membership values for the attributes to train the network.
- The critical rules are determined which are used to train the neural network.
- To prove that the system produces consistent result.
- The system is designed to analyze the DBLP dataset, which consist of different types of Conferences held in various years and predict the Emerging and Subsiding Trends

CHAPTER 3
METHODOLOGY

3. SYSTEM METHODOLOGY

Knowledge Discovery in Databases (KDD) means the application of non-trivial procedures for identifying effective coherent potential useful and previously unknown patterns in large databases.

The KDD process generally consists of the following three processes

Pre-processing

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns already present in the data, the target dataset must be large enough to contain these patterns while remaining concise enough to be mined in an acceptable timeframe. A common source for data is a data mart or data warehouse. Pre-process is essential to analyse the multivariate datasets before data mining.

It consists of all the actions taken before the actual data analysis starts. It may be performed on the data for the following reasons. Solving data problems that may prevent us from performing any type of analysis of data, understanding the nature of data, performing a more meaningful data analysis and extracting more meaningful knowledge from a given set of data

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns already present in the data, the target dataset must be large enough to contain these patterns while remaining concise enough to be mined in an acceptable timeframe. Pre-process is essential to analyze the multivariate datasets before data mining.

A) Stop word removal

Removing most frequently occurring words like "is, and, of, with, through, using etc..... Search for the stop words in the document titles and remove the stop words from the document titles.

B) Compound Nouns

Compound nouns are derived from the stop word removed data. The purpose of compound noun is to avoid the dictionary approach. Dictionary approach requires

maintaining the technical terms in a dictionary and it has an over head of updating the dictionary every time, if we use different database.

Compound noun is formed by combining the adjacent words in a stop removed title. The formula used for the formation of the compound noun is as follows:

$$FLR(CN) = f(CN) \times (\prod_{i=1}^{L} (FL(N_i)+1)(FR(N_i)+1))^{\frac{1}{L}}$$

where f(CN) -> frequency of the candidates CN,

FL(Ni) and FR(Ni) -> frequencies of the right and the left of each noun Ni

Description of the database:

The database consists of 1000 titles of various conferences in a particular year. It provides us with various details about the conference like Title, Year, Author, and Conference Name.

No.	Attributes	Details
1	Title	Ex. An Object Oriented Approach to Software Development.
2	Year	2005 to 2010
3	Author	Ex. Satoshi Kuribayashi,Takaki Kimura, Hiroya Tanaka
4	Conference Name	Ex.ACM,ACmaze,ccgrid

Table 1.Database description

The overall system structure of the project is shown in figure, each one of the steps are explained in detailed in the following sections.

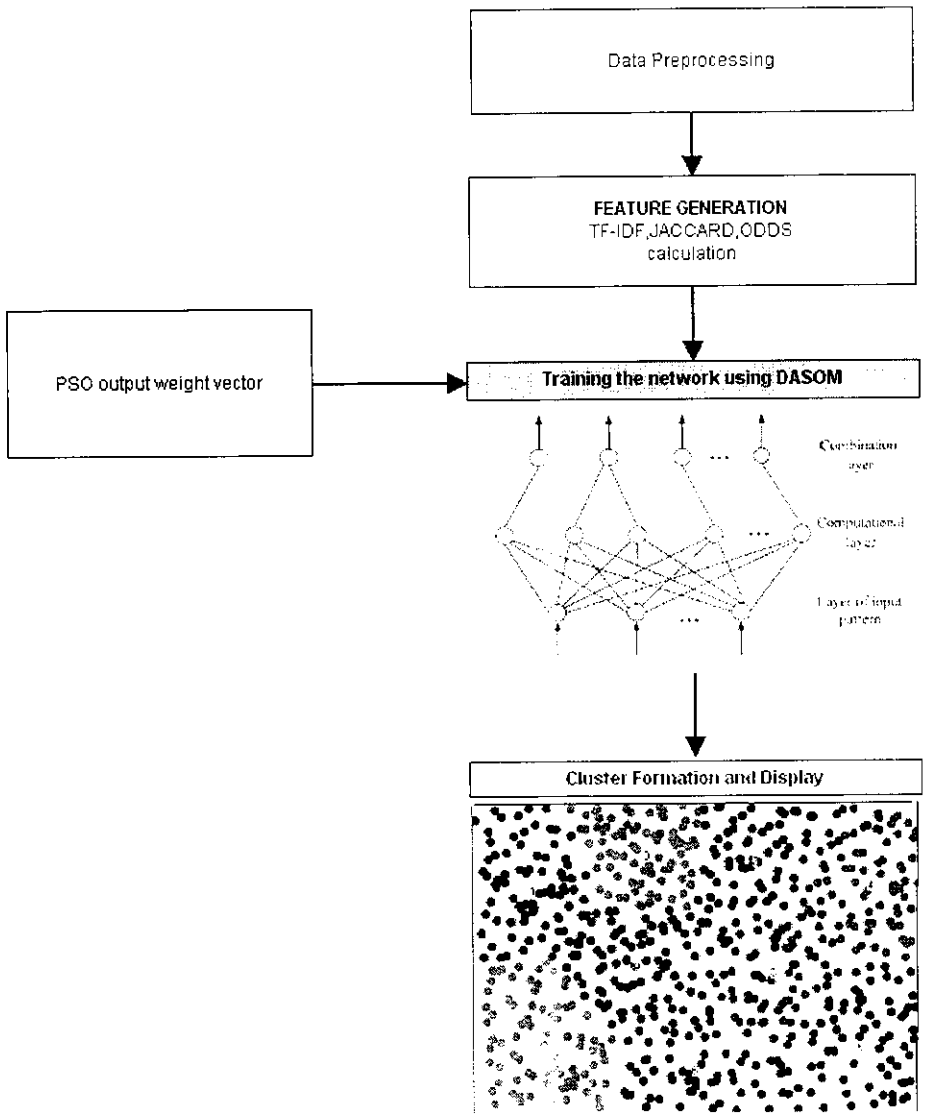

Figure 2.System Architecture

3.1 Data Preprocessing

A) Stop word removal

Removing most frequently occurring words like "is, and, of, with, through, using etc.....

B) Compound Nouns

$$FLR(CN) = f(CN) \times (\prod_{i=1}^{L} (FL(N_i)+1)(FR(N_i)+1))^{\frac{1}{2L}}$$

where f(CN) -> frequency of the candidates CN,

FL(Ni) and FR(Ni) -> frequencies of the right and the left of each noun Ni

3.2 Feature Extraction

A) Tf-idf determination

The tf–idf weight (term frequency–inverse document frequency) is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad \mathrm{idf}_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

$$(\mathrm{tf\text{-}idf})_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

ni,j is the number of occurrences of the considered term (ti) in document d

Denominator is the sum of number of occurrences of all terms in document dj.

| D | : total number of documents in the corpus

$|\{d : t_i \in d\}|$: number of documents where the term ti appears (that is $n_{i,j} \neq 0$).

B) Jaccard's Matching Co-efficient

The Jaccard index, also known as the Jaccard similarity coefficient (originally coined coefficient de communauté by Paul Jaccard), is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$Jaccard(t) = \frac{DF(w1 \cap w2 \cap .....wn)}{DF(w1 \cup w2 \cup .....wn)}$$

where DF(wj) means number of hit documents in the corpus for a word wi. Each value of Jaccard coefficient shows the strength of co-occurrence of multiple words as a importance of the terms in the given corpus.

C) Simple Appearance ratio of terms in a set of documents

The odds in favor of an event or a proposition are expressed as the ratio of a pair of integers, such that the first represents the relative likelihood that the event will happen, and the second, the relative likelihood it won't, as in "the odds that a randomly chosen day of the week is a Sunday are one to six," which is sometimes written 1:6, or 1/7. In probability theoryand statistics, where the variable "p" is the probability in favor of the event, and the probability against the event is therefore 1-p, "the odds" of the event are the quotient of the two, or . That value may be regarded as the relative likelihood the event will happen, expressed as a fraction (if it is less than 1), or a

multiple (if it is equal to or greater than one) of the likelihood that the event will not happen. In the example just given, saying the odds of a Sunday are "one to six" or, less commonly, "one-sixth" means the probability of picking a Sunday randomly is one-sixth the probability of not picking a Sunday. While the mathematical probability of an event has a value in the range from zero to one, "the odds" in favor of that same event lies between zero and infinity. The odds against the event with probability given as "p" are .

$$Odds(t) = \frac{DF(t)}{|D_{period}| - DF(t)}$$

Where DF(t) means the frequency of the appearance of each term t in each set of documents Dperiod .

3.3 Self Organising Maps

A self-organizing map (SOM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), representation of the input space of the training samples, called a map. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space. The Learning Rate required for the mapping function is experimented and fixed.

Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. Mapping automatically classifies a new input vector.

A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from data space onto the map is to find the node with the closest weight vector to the vector taken from data space and to assign the map coordinates of this node to our vector.

Algorithm

1. Randomize the map's nodes' weight vectors
2. Grab an input vector
3. Traverse each node in the map
1. Use Euclidean distance formula to find similarity between the input vector and the map's node's weight vector
2. Track the node that produces the smallest distance (this node is the best matching unit, BMU)
4. Update the nodes in the neighbourhood of BMU by pulling them closer to the input vector
1. $\mathbf{Wv}(t+1) = \mathbf{Wv}(t) + \Theta(t)\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t))$
5. Increase t and repeat from 2 while $t < \lambda$

Wv - weight vector

$\alpha(t)$ – learning rate

$\Theta(t)$ – neighbourhood function

$\mathbf{D}(t)$ – input value of indices

3.4 Training the Network using DASOM

The dynamic and adaptive self organizing maps (DASOM) is a model which has characteristics of a dynamic network and adaptive learning speed. The weight vectors in the network are initialized by selecting text vectors randomly from the input space. In the training phase, cosine similarities of input pattern (text vector) with the weight vectors are computed and select the largest to find the best match (winning) node. The weight vectors of the nodes that lie inside the topological neighborhood of winning node are adjusted by using the amended rule related to similarity. When the similarity is smaller than a given threshold, a new node is inserted into the network and the input pattern is assigned to weight vector of the new node. After the network

becomes stable, the similar nodes represented similar classes are combined for optimization.

1.) Randomize the map's nodes' weight vectors

2.) Grab an input vector

3.) Traverse each node in the map

Use Cosine Similarity formula to find similarity between the input vector and the map's node's weight vector

$$sim(W_k, V_j) = \frac{W_k}{W_k} \bullet \frac{V_j}{V_j} = \frac{\sum_{i=1}^{m} w_{ik} v_{ij}}{\sqrt{\sum_{i=1}^{m} (w_{ik})^2 \bullet \sum_{i=1}^{m} (v_{ij})^2}}$$

Track the node that produces the Maximum Similarity between the chosen input vector and an arbitrary node in the network.

$$sim(W_k, V_l) = \max_{j=1}^{n} (sim(W_k, V_j))$$

4.) Update the nodes in the neighborhood by pulling them closer to the input vector i.e., updating the topological neighborhood position

$$v_{il} = v_{il} + \alpha (w_{ik} - v_{il})$$

$$v_{io} = v_{io} + \alpha' (w_{ik} - v_{io}) \quad o \in NE_l(t)$$

$$\alpha' = \alpha \times sim(V_o, V_l)$$

Where α is the learning rate parameter which decreases monotonically with the increasing time t, but never goes to zero. In this application, for computational reasons, α(t) is taken as,

$$\alpha(t) = \alpha_0(1 - \frac{t}{T})$$

Where α0 is a constant value assigned initially. T is another time constant of the DASOM algorithm.

5.) Optimization is done using the following,

In order to find arbitrary cluster, DASOM utilize TF-IDF based cluster's idea. After training phase, the Euclidian distance of all nodes in the computational layer are compared and nodes representing similar distance are grouped into respective classes i.e., Emergent and Subsiding.

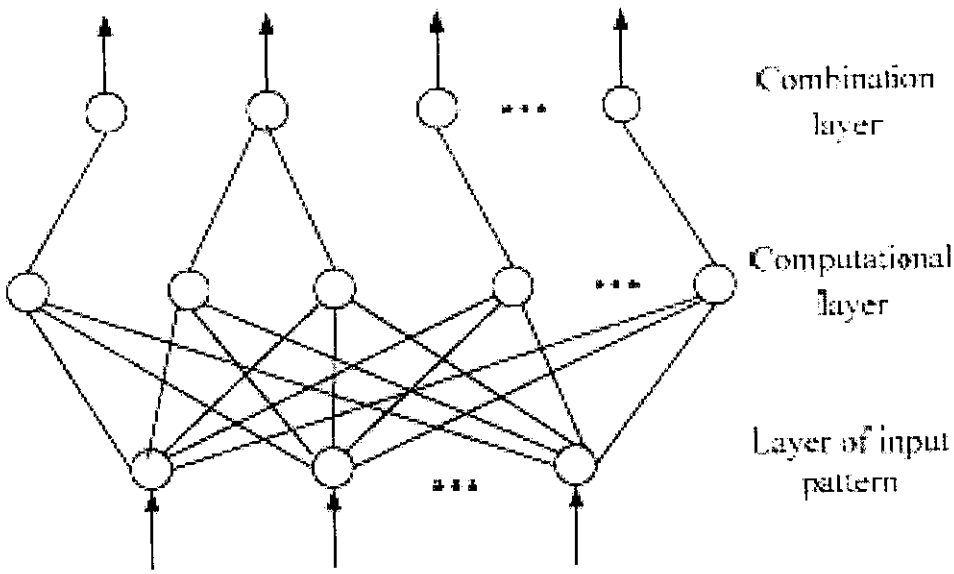

Figure 3.DASOM Network Diagram

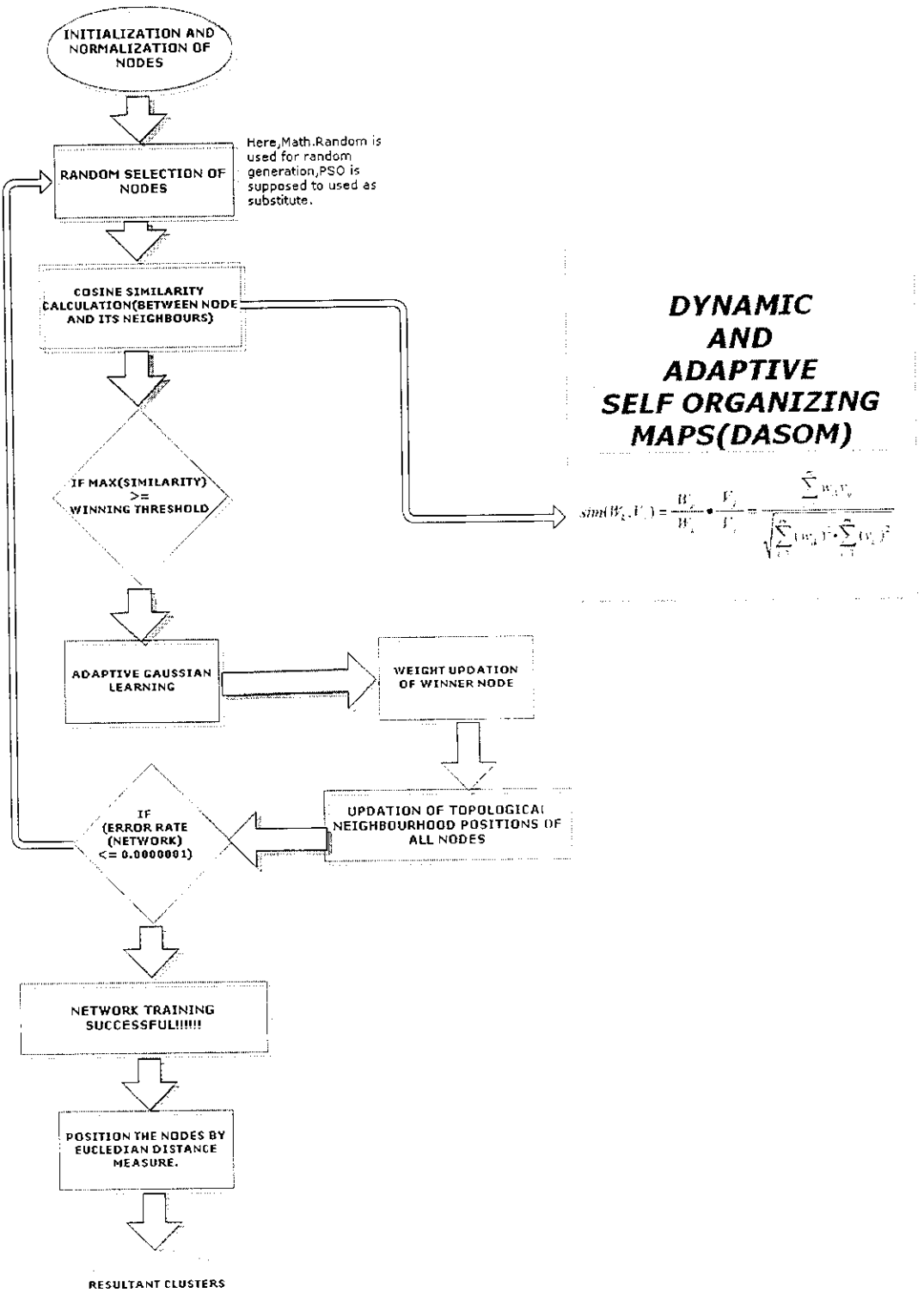

Figure 4.DASOM Flowchart

3.5 Particle Swarm Optimization

The PSO algorithm is the result of inspiration caused by social behaviour of flock of birds.

PSO is a global optimization problem which will iteratively improve the candidate solution with respect to a fitness measure.

PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae.

The output of PSO is fed as the input weight vectors (Three Dimension) for the nodes in the network to be trained by the DASOM algorithm.

Due to its simple implementation, minimum mathematical processing and good optimization capability, PSO has attracted more attentions. As Vassiliadis and Dounias summarized, PSO can be considered the most popularly and frequently applied among nature-inspired techniques according to related publications. PSO simulates the behavior of a bird flock.

When a flock of birds forage for food, two simple and important strategies are:
(a) searching for the peripheral region around the bird that is nearest to the food;
(b) Judging the position of the food by its own flying experience.

Inspired by the two strategies, the search space of optimization problem is regarded as birds' flying space; every bird is abstracted as a particle with non-quality and non-volume so as to denote a candidate solution; and the optimal solution to be searched for the problem is the food to all the particles. Then, the basic PSO algorithm comprises a swarm of particles moving in the D-dimensional search space which includes all possible candidate solutions.

We denote the ith particle as Xi = (xi,1,xi,2, ...xi,d), and its flying velocity as Vi = (vi,1,vi,2. . . . ,vi. d). Depending on the two strategies above, when each particle "flies" in the search space. Pi = (pi,1,pi,2, . . . , pi .d) denotes the personal best position the 'i'th particle has found so far, and Pg = (pg,1.pg.2, ...pg ,d) is the

global best position discovered by the swarm. At each time step t, both of each particle's velocity and position are updated so that a particle moves to a new position. The following two equations are employed to calculate the velocity and position:

$$V_i^{t+1} = V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t)$$
$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

Where c1 and c2 are two positive constants (acceleration constants), r1 and r2 are two uniform random numbers in [0, 1]. A constant Vmax is often used to limit each particle's velocity to guarantee that most of the new positions of the particles will be restricted in the search space of feasible region at each iteration. Eqs. (1) and (2) lead to the movement of each particle towards its cognitive best position (Pi) and "social" best position (Pg) with random perturbations caused by c1r1 and c2r2, respectively. The quality of a particle's position is measured by its fitness value, or namely, the better fitness value means the better position.

In this paper, we discuss minimization problems, that is, the smaller the objective value, the better the particle's position and fitness value. Of course, when solving maximization problems, we could transform the maximization problem into minimization problem. Shi and Eberhart modified the original PSO by introducing an inertia weight (x) to Eq. (1) to balance exploitation and exploration. The modified Eq. (1) is as follows:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t)$$

According to the numerical tests in , the adoption of x improves the performance of PSO largely, so the combination of Eqs. (3) and (2) is always considered as the standard PSO by many researchers. Fig. 1 is a schematic drawing that illustrates how the position of a particle updates. and Fig. 2 presents the pseudo-code of

standard PSO. Since PSO was proposed, investigations have been made theoretically and experimentally to analyze and improve PSO.

Ozcan and Mohan were the first to analyze PSO in theoretical aspects, and the trajectories of particles were analyzed in one dimension under a simplified model. Following this, the theoretical analysis was further conducted in multi-dimensional search space and analytical results show that particles oscillate in a sinusoidal-wave manner. Clerc and Kennedy explored how PSO works from a mathematical perspective, introduced a constriction factor v to guarantee the convergence of PSO, and analyzed the trajectory of a single particle in both discrete time and continuous time.

Van den Bergh and Engelbrecht analyzed how the inertia weight and acceleration constants affect the trajectories of particles and provided theoretical findings on the dynamics of the PSO systems. These studies provided theoretical supports for the research on the improvement of PSO.In order to enhance the performance of PSO, attempts have been made, including restricting the value of velocity, integrating mechanisms of different optimization algorithms, studying neighborhood topology of particles in the swarm of PSO, and developing comprehensive learning strategies.

PSO Algorithm:-

// Initialization:

for i=1 **to** the swarm size **do**

Initialize X_i within the search range of (X_{min}, X_{max}) randomly;

Initialize V_i within the velocity range of (V_{min}, V_{max}) randomly;

$P_i = X_i$;

end for

Evaluate each particle;

Identify the best position P_g;

// Loop:

While (stop criterion is not satisfied & t < maximum iteration times) **do**

for i=1 **to** the swarm size **do**

$V_i^{t+1} = aV_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t)$

$X_i^{t+1} = X_i^t + V_i^{t+1}$

$P_i^{t+1} = P_i^t$

$P_g^{t+1} = P_g^t$

Evaluate $fitness(X_i^{t+1})$;

if $fitness(P_i^{t+1}) < fitness(X_i^{t+1})$ **then**

Update P_i^{t+1};

end if

if $fitness(P_g^{t+1}) < fitness(P_g^t)$ **then**

Update P_g^{t+1};

end if

end for

end while

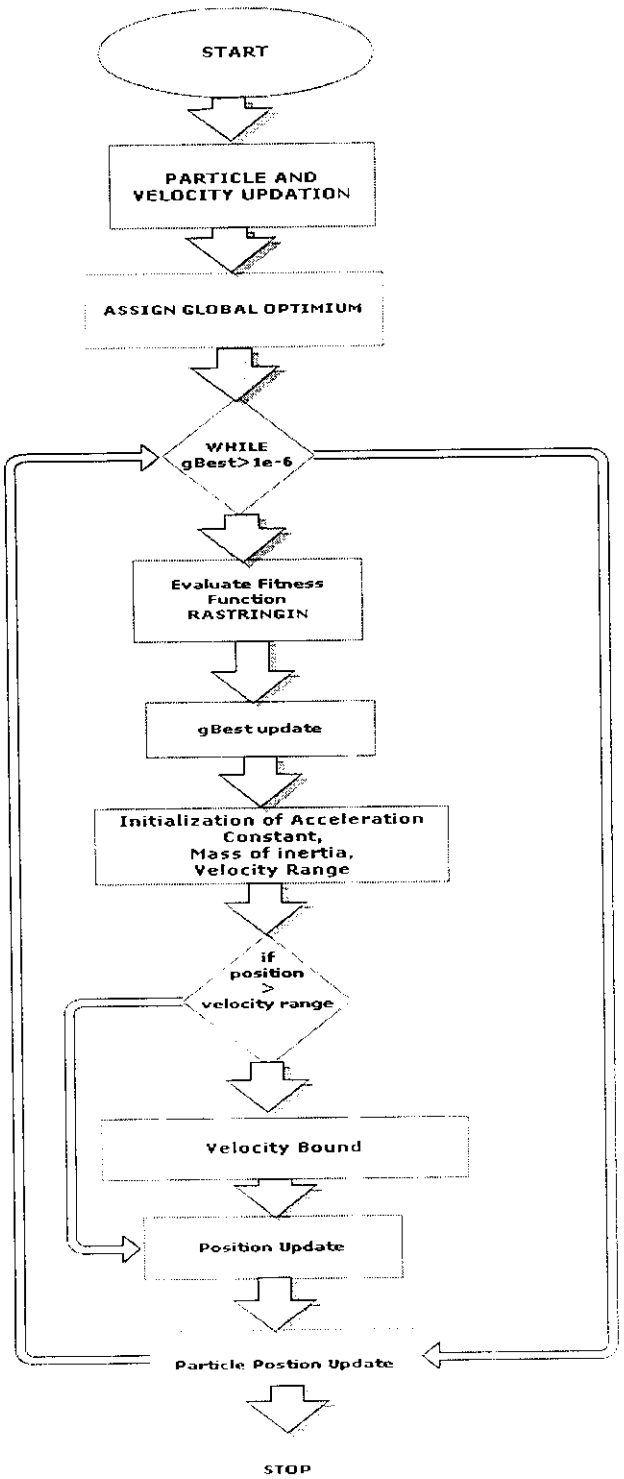

Figure 5. PSO Flowchart

3.6 Regression Analysis

Regression analysis includes any techniques for modelling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. More specifically, regression analysis helps one understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables — that is, the average value of the dependent variable when the independent variables are held fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables. In all cases, the estimation target is a function of the independent variables called the regression function. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the regression function, which can be described by a probability distribution.

Linear regression is an approach to modeling the relationship between a scalar variable y and one or more variables denoted X. In linear regression, data are modeled using linear functions, and unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X.

The linear regression analysis technique is used to detect the degree and intercept of existing trends based on the importance indices. The degree of each term t is calculated

as the following:

$$\mathrm{Deg}(t) = \frac{\sum_{i=1}^{M}[(y_i] - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{M}(x_i - \bar{x}^2)}$$

where is the average of the M time points, and $\bar{y}$ is the

average of each importance index for the period. Simultaneously,

we calculate the intercept Int(t) of each term t as follows:

$$Int(t) = \bar{y} - Deg(t)\bar{x}$$

Emergent terms are determined either by sorting the degree in ascending order or sorting the intercept in descending order.

Subsiding terms are determined either by sorting the degree in descending order or sorting the intercept in ascending order

Performance Evaluation

Performance of the DASOM model and DASOM with PSO with various input values are found using sensitivity, specificity, accuracy, Precision, Error Rate formulas from confusion matrix and dataset are partitioned in various percentage for training and testing the model.

Confusion Matrix:

		Prediction outcome		**TOTAL**
		P	n	
Actual	p'	True Positive	False Negative	**P'**
	n'	False Positive	True Negative	**N'**

Table 2.Confusion Matrix

Sensitivity:

Sensitivity (also called recall rate in some fields) measures the proportion of actual positives which are correctly identified as such (e.g. the percentage of sick people who are correctly identified as having the condition).

Sensitivity = (TP / (TP + FN))/100;

Specificity:

Specificity measures the proportion of negatives which are correctly identified (e.g. the percentage of healthy people who are correctly identified as not having the condition).

Specificty = (TN / (FP + TN))/100;

Accuracy:

Accuracy of a measurement system is the degree of closeness of measurements of a quantity to its actual (true) value. The precision of a measurement system, also called reproducibility or repeatability, is the degree to which repeated measurements under unchanged conditions show the same results.Although the two words can be synonymous in colloquial use, they are deliberately contrasted in the context of the scientific method.

Accuracy = ((TP + TN) / (TP + TN + FP + FN))/100;

Precision:

The Precision of a system is called reproducibility or repeatability, is the degree to which repeated measurements under unchanged conditions show the same results. Although the two words can be synonymous in colloquial use, they are deliberately contrasted in the context of the scientific method.

Precision = (TP / (TP + FP))/100;

CHAPTER 4

RESULTS

4.1 Experimental Results:

This chapter is devoted to explain the simulation results because results of the project promise the effective operation of the model that has been designed. The Classification of the patterns is multistage system with data preprocessing module, Parameter Extracting and Weight Vector Calculation and Euclidean distance calculation module and classification module which are clearly analyzed in this chapter.

The Cluster formation system is tested on Intel Pentium Dual Core,2.66 MHz Clock Speed,3 GB RAM Laptop. For testing the validity of the proposed model, the DBLP database is selected from DBLP website. The proposed version is the advanced method of basic Self Organizing Maps (SOM), Dynamic and Adaptive Self Organizing Maps (DASOM) and the Particle Swarm Optimization Algorithm (PSO).

Self Organizing Map:

A **self-organizing map** (**SOM**) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), representation of the input space of the training samples, called a **map**. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space. The Learning Rate required for the mapping function is experimented and fixed.

The SOM classifier is run with 40 % of the data taken for training and 60% taken for testing for the ACM dataset of year 2010.

Table 3 SOM performance with change in Learning Rate values

Learning Rate vs. Classification Accuracy

Dataset – ACM, Samples – 1000, Training - 40%, Testing - 60%,
No of input neuron: 3, Initial Weight vectors: Random,
No of output neuron: 1, No of iterations: 1.

Learning Rate (η)	Classification Rate (%)
0.1	35
0.2	29
0.3	40.3
0.4	60.9
0.5	43.1
0.6	62.76
0.7	**73.14**
0.8	39.59
0.9	47.86

Table 3 shows that the Classification Accuracy is Maximum when the learning rate value is set to 0.7. Graph 1 represents the variation of Classification Accuracy in correspondence to the change in the learning rate values.

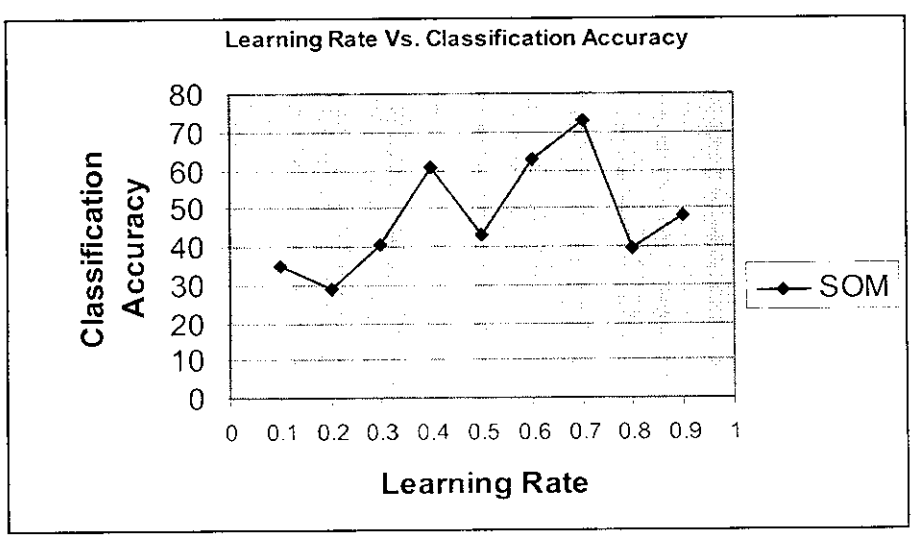

Figure 6: Learning rate vs. classification accuracy

Scale:

X-axis: 1 unit = 0.1

Y-axis: 1 unit = 10% accuracy

Table 4. Performance of SOM for the year 2010

SOM Performance:

Samples – 1000

No of input neuron: 3

Learning rate: 0.7

Initial Weight vectors: Random

No of output neuron: 1

No of iterations: 1

Training Samples (%)	Testing Samples (%)	SOM		
		Classification Rate (%)	Training Time (s)	Testing time(s)
10	90	**59**	6	10
20	80	**61.1**	6.3	8.9
30	70	**62.55**	6.6	8.8
40	60	**71.13**	7.3	7.7
50	50	**72.6**	7.2	7.2
60	40	**73.14**	7.6	7.3
70	30	**77.42**	8.8	6.6
80	20	**78.39**	8.9	6.3
90	10	**37.49**	10	6

The above shows the variation in **Classification Accuracy** of values with respect to **Learning Rate** values. It is represented in the graph 1. It shows that the accuracy is Maximum when the Learning Rate is **0.7.** Therefore for the testing samples the value is fixed at 0.7.Table 5 shows the Classification Accuracy of SOM for the testing samples after the parameters are fixed.

Dynamic Adaptive SOM

The principle of this algorithm is to create the set of initial text vectors for representing the documents; the term frequencies for each term were measured. The text vectors are then formed by using the term frequencies as the feature values. The document is represented as a text vector and each unique term is one dimension of this vector. Let m denote the dimension of the text vector space.

Simulation Results

Table 1: Classification rate for varying learning rate

Dataset – DBLP
Number of samples – 1000 titles
Number of training samples - 400 (40%)
Number of testing samples - 600 (60%)
Initial Weight vectors: Random
No of input neuron: 3
No of output neuron: 1
Winning Threshold (ρ) = 0 to 1
Number of iterations (t) =27574
Time constant (T) = 1000

Table 5. Behavior of DASOM Neural Network for variable Learning Rate

Learning Rate (α)	Classification Rate (%)	Execution Time (s)
0.1	89	9.05
0.2	87	9.03
0.3	89	9.03
0.4	88	9.33
0.5	**92**	**9.98**
0.6	86	9.92
0.7	86	9.98
0.8	87	10.00
0.9	78	10.05

The behavior of the DASOM Neural Network with the varying Learning Rate is tabulated above in Table 3.The learning rate with the Higher Classification rate is the efficient and that learning rate can be fixed throughout the Training and Testing process of DASOM Neural Network. For sake of clarity, the Execution time for variable learning rates is also tabulated.

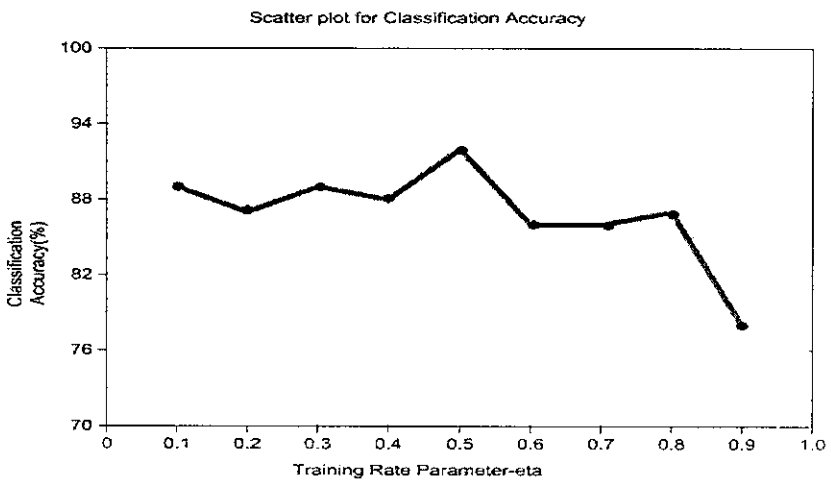

Figure 7: Scatter Plot of Classification rate for varying learning rate

The above figure shows the scatter plot of Classification rate for variable learning rate in the DASOM Neural Network. The Classification Accuracy for various learning rate are measured and plotted. In the graph, for training rate 0.5, we obtain a peak. so, 0.5 is fixed as the standard learning rate for various Training and Testing process in DASOM Neural Network.

Table 6: Emergent Terms as given by DASOM Neural Network

For the year 2010

Emergent Terms	Subsiding Terms
Sensor Networks	network information
Wireless Sensors	intrusion tolerant
Social Network	flow analysis
Web Service	delay energy
Cloud Computing	web systems
Virtual Machine	weak references
Pervasive Environment	view digital
Recommender System	verifying address
Service Oriented	user privacy
Routing Protocol	entrusted clients
Machine Translation	trust management
Load Balancing	tool interface
Information Security	Secure group
Colony Optimization	strategy active
Swarm Optimization	Strategies predicate
Semantic Web	subtle expressions

The above table lists the Emergent and subsiding terms as output by the DASOM Neural Network. The first column gives the list of top Emergent words and second column lists the Subsiding Words. These words are taken from direct output of the DASOM Neural Network.

Particle Swarm Optimization:

Due to its simple implementation, minimum mathematical processing and good optimization capability, PSO has attracted more attentions. As Vassiliadis and Dounias summarized, PSO can be considered the most popularly and frequently applied among nature-inspired techniques according to related publications.PSO simulates the behavior of a bird flock.

Table 7: Performance of DASOM for the year 2010

DASOM Parameters:

Number of input neurons =3
Initial Weight vectors= Random
Radius of the Cluster = 3 to 8
Winning Threshold (ρ) = 0 to 1
Learning rate = 0.5
No of output neuron= 1
No of iterations (t) = 27314
Time constant (T) = 1000

PSO Parameters:

Initial Population Size = 300
Social Co-efficient (c2) = 2
pBest = Random
Mass of Inertia (w)=0.9
Fitness Function = Rastrigin Function
Radius of the Cluster = 3 to 11
Cognitive Factor (c1)=2
r1, r2 =Random(0 to 1)
gBest=1e^6
Vmax=0.5, Vmin= -0.5
No. of generations=300

Training Samples (%)	Testing Samples (%)	DASOM without PSO			DASOM with PSO		
		Classification Rate (%)	Training Time (s)	Testing time(s)	Classification Rate (%)	Training Time (s)	Testing time(s)
10	90	85.95	8.093	8.9	87.64	8.015	9.1
20	80	84.52	8.75	8.4	86.63	8.750	9.0
30	70	83.54	8.765	8.5	85.29	8.765	8.9
40	60	82.99	9.784	8.7	88.97	10.031	8.8
50	50	83.47	9.875	8.6	84.17	9.859	8.6
60	40	82.50	10.484	8.5	91.66	10.421	8.4
70	30	78.60	10.516	8.1	86.73	10.453	8.0
80	20	86.97	12.063	8.3	87.06	12.078	7.6
90	10	84.93	11.453	8	86.94	11.406	7.5

The above table shows the performance comparison of DASOM Neural Network vs. DASOM with PSO Neural Network. The performance is listed for

Training and Testing Combinational Input sets. The classification Accuracy for the Input Samples is calculated for both variants of DASOM Neural Networks and tabulated above. Further, the Execution time taken for Training and Testing Samples are also listed in the Table. The inference incurred from the table is, DASOM with PSO performs relatively better than the DASOM itself.

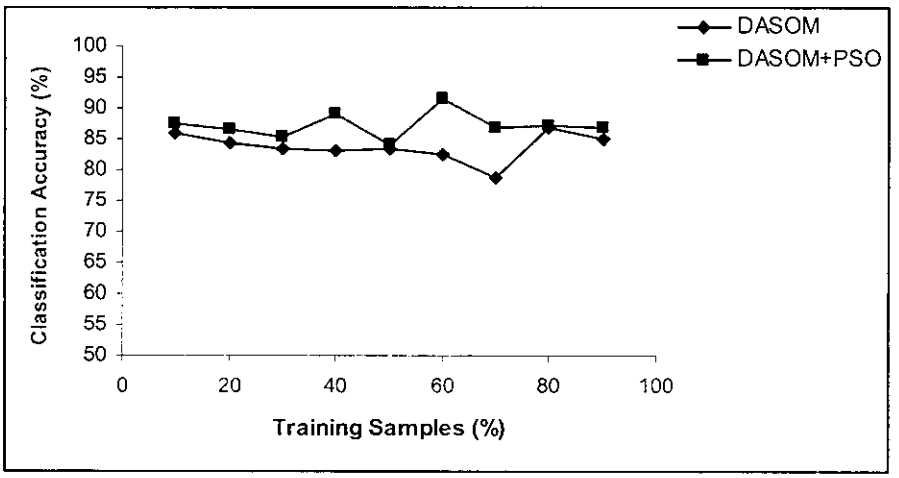

Figure 8: Scatter plot for comparison of DASOM without PSO and with PSO

The above figure shows the Scatter plot for comparison of performance of DASOM Without PSO and with PSO. The classification accuracy for various training samples is plotted. This clearly shows that DASOM with PSO has better classification accuracy than DASOM itself.

Table 8: Performance of DASOM for the year 2010

Testing Set (%)	Number of True Positive terms	Number of True Negative terms	Number of False Positive terms	Number of False Negative terms	Sensitivity (%)	Precision (%)	Specificity (%)	Error Rate (%)
90	505	3719	203	487	50.90	71.32	94.82	14.04
80	537	3155	103	573	48.37	83.90	96.83	15.47
70	434	2759	153	476	47.69	73.93	94.74	16.45
60	354	2364	157	400	46.95	69.27	93.77	17.00
50	45	2233	384	67	40.17	10.06	85.32	16.52
40	14	1787	296	86	14.06	4.51	85.79	17.49
30	19	1267	207	143	11.72	8.40	85.95	21.39
20	104	844	35	107	49.28	74.82	90.76	24.06
10	46	416	13	69	40.06	77.96	96.97	15.07

The above table shows necessary parameters like True positive,True negative ,False positive, False negative which are necessary to calculate confusion matrix . The confusion matrix is used to calculate Performance Metrics like Sensitivity ,Precision ,Specificity and Error rate.These values indicate the output of DASOM Neural Network for the Testing set samples.

Table 9: Performance of DASOM with PSO for the year 2010

Testing Set (%)	Number of True Positive terms	Number of True Negative terms	Number of False Positive terms	Number of False Negative terms	Sensitivity (%)	Precision (%)	Specificity (%)	Error Rate (%)
90	345	3962	25	582	37.21	93.24	99.37	12.35
80	328	3456	12	572	36.44	96.47	99.65	13.37
70	287	2973	29	533	35	90.32	99.03	14.70
60	239	2675	31	330	42.06	88.51	98.85	11.02
50	214	2083	7	425	33.49	96.83	99.66	15.83
40	162	1839	148	34	82.65	52.25	92.55	8.33
30	139	1280	87	130	51.67	61.50	93.63	13.26
20	80	869	59	82	49.38	57.55	93.64	12.93
10	37	436	21	50	42.52	63.79	95.40	13.05

The above table shows necessary parameters like True positive,True negative ,False positive, False negative which are necessary to calculate confusion matrix .The confusion matrix is used to calculate Performance Metrics like Sensitivity ,Precision ,Specificity and Error rate.These values indicate the output of DASOM with PSO Neural Network for the Testing set samples.

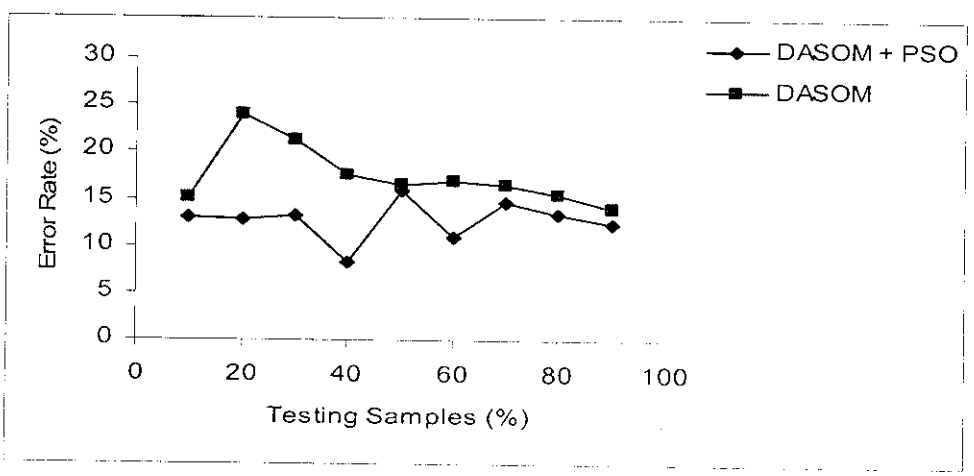

Figure 9: Comparison of Error rate of DASOM with PSO and DASOM

The above figure shows the Comparison of Error rate of DASOM with PSO and DASOM for various testing samples tested with DASOM and DASOM with PSO .It is obvious that the Error rate of DASOM with PSO is appreciably less than DASOM itself.

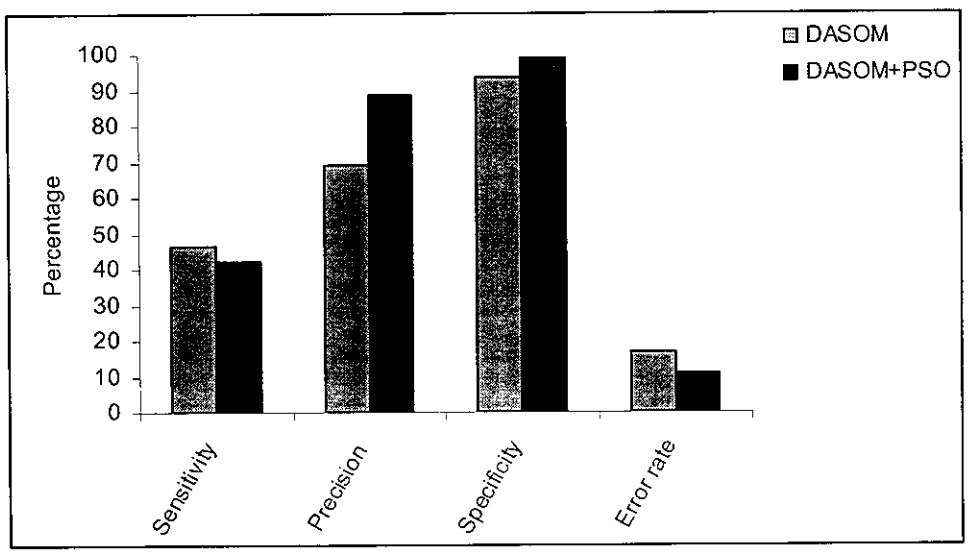

Figure 10. Comparison of Sensitivity, Specificity, Precision and Error rate of DASOM with PSO and DASOM without PSO

The above Figure shows the Comparison of Sensitivity, Specificity, Precision and Error rate of DASOM with PSO and DASOM without PSO .The output of DASOM with PSO with respect to precision Specificity is appreciably higher and the error rate is lower when compared to DASOM itself.

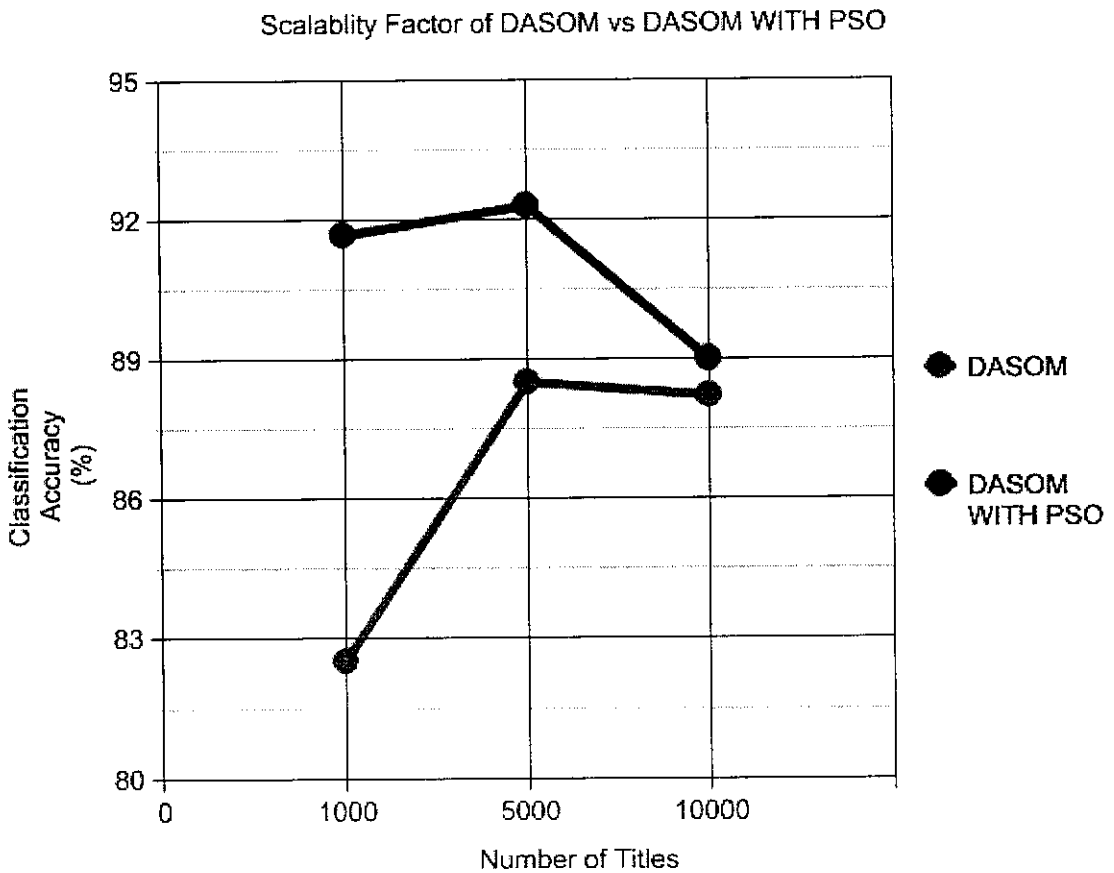

Figure 11 Scalability factor of DASOM vs. DASOM with PSO

The above figure shows the Scalability factor of DASOM vs. DASOM with PSO for various sets of inputs like 1000 Titles,5000 Titles and 10000 Titles.The Scalablity of DASOM increases upto a certain level and remains consistent after that level whereas DASOM with PSO has better Classification Accuracy than the DASOM itself and it seems to be consistent upto a certain level.

Table 10: Scalability Comparison for DASOM and DASOM with PSO

No of Titles	True Positive	True Negative	False positive	False Negative	Sensitivity	Accuracy	Precision	Specificity	Error rate	Time Taken
5000 without PSO	1987	22468	56	3118	38.92	88.51	97.26	99.75	11.49	38.65
5000 With PSO	1937	23562	106	2024	48.90	92.29	94.81	99.55	7.70	39.06
10000 without PSO	4404	44208	105	6328	40.85	88.23	97.67	99.76	11.76	1m46 sec
10000 With PSO	4876	44164	410	5645	46.35	89.01	92.24	99.08	10.99	1m46ssec
5000 (60%) Without pso	1987	22468	56	3118	38.92	88.51	97.26	99.75	11.49	29.46
5000 (60%) With pso	1937	23562	106	2024	48.90	92.29	94.81	99.55	7.70	51
10000 (60%) Without Pso	4404	44208	105	6378	40.85	88.23	97.67	99.76	11.76	56
10000 (60%) With Pso	4876	44164	410	5645	46.35	89.01	92.24	99.08	10.99	51

The above table shows necessary parameters like True positive,True negative ,False positive, False negative which are necessary to calculate confusion matrix .The confusion matrix is used to calculate Performance Metrics like Sensitivity ,Precision ,Specificity and Error rate.These values indicate the output of DASOM and DASOM with PSO Neural Network with respect to Scalablity Factor.

Regression Analysis

Linear regression is an approach to modeling the relationship between a scalar variable y and one or more variables denoted X. In linear regression, data are modeled using linear functions, and unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X.

The linear regression analysis technique is used to detect the degree and intercept of existing trends based on the importance indices. The degree of each term t is calculated

as the following:

$$\text{Deg (t)} = \frac{\sum_{i=1}^{M}[(y_i] - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{M}(x_i - \bar{x}^2)}$$

where is the average of the M time points, and $\bar{y}$ is the

average of each importance index for the period. Simultaneously,

we calculate the intercept Int(t) of each term t as follows:

$\text{Int(t)} = \bar{y} - \text{Deg(t)}\bar{x}$

Emergent terms can be determined either by sorting the degree in ascending order or sorting the intercept in descending order.

Subsiding terms can be determined either by sorting the degree in descending order or sorting the intercept in ascending order.

Table 11. Emergent Terms in 2010

Time for calculation of 1000 documents = 4 min 52 sec

Rank	Tf-Idf Co-efficient			Odds Co-efficient			Jaccard Co-efficient		
	Term t	Deg(t)	Int(t)	Term t	Deg(t)	Int(t)	Term t	Deg(t)	Int(t)
1	Web Service	-0.00037	0.5768	Wireless Sensor	1.05E-17	5.0001	Scalable Barrier	-0.3021	608.7
2	Supply Chain	-0.00023	0.4723	Web Service	3.49E-18	5.0000	Cryptographic Role	-0.2841	572.7
3	User Interface	-0.00023	0.4721	Cloud Computing	2.80E-18	4.0000	Cache Partitioning	-0.2728	550.2
4	Semantic Web	-0.00021	0.4288	Performance Evaluation	2.80E-18	4.0000	Open Source	-0.2621	528.9
5	Sensor Network	-0.00012	0.2521	Intrusion Detection	2.80E-18	4.0000	State Replication	-0.2570	519.1
6	Service Discovery	-0.00011	0.2215	Access Control	2.45E-18	4.0000	Biometric Identification	-0.2461	595.8
7	Virtual Machine	-0.00007	0.2016	Virtual Machine	2.45E-18	2.0000	Setup Strategy	-0.2350	472.9
8	Service Composition	-0.00005	0.1055	Wireless Networks	1.75E-18	2.5000	Strategic Role	-0.2222	449.5
9	Cloud Computing	-0.00004	0.0992	Social Network	1.40E-18	5.5000	Video Detection	-0.2166	437.6
10	Smart Card	-0.00004	0.0876	Machine Learning	1.40E-18	2.0000	Recognition Performance	-0.2056	415.5

The above table shows top ten phrases extracted from the titles of the four conferences based on the trend criteria of the three indices to annual sets of the documents.

Table 12. Subsiding Terms in 2010

Time for calculation of 1000 documents = 4 min 52 sec

Rank	Tf-Idf Co-efficient			Odds Co-efficient			Jaccard Co-efficient		
	Term t	Deg(t)	Int(t)	Term t	Deg(t)	Int(t)	Term t	Deg(t)	Int(t)
1	Tightly Coupled	-9.40E-06	0.0193	image extraction	3.50E-19	5.0000	Evidential Probabilities	0.4230	355.6
2	Error Detection	-9.40E-06	0.0187	dynamic data	3.50E-19	5.0000	Investigating Limitations	0.4230	355.6
3	Network Security	-9.11E-06	0.0187	persuasive technology	3.50E-19	5.0000	Untrusted Clients	0.4230	355.6
4	Resource Allocation	-6.77E-06	0.1025	information protocol	3.50E-19	5.0000	Transient Faults	0.4230	355.6
5	Adaptive Routing	-3.87E-06	0.0074	service programs	3.50E-19	5.0000	Soap Messages	0.4230	355.6
6	Support Vector	-7.69E-05	0.1558	routing strategy	3.50E-19	5.0000	Queuing Mechanisms	0.4230	355.6
7	Memory Systems	-4.34E-05	0.0876	network construction	3.50E-19	5.0000	Buffer Overflow	0.4230	355.6
8	Genetic Algorithms	-1.24E-05	0.0262	service component	3.50E-19	5.0000	Vulnerability Monitors	0.4230	355.6
9	Key Management	-1.43E-05	0.0301	virtualization technique	3.50E-19	5.0000	Quantifying Forensic	0.4230	355.6
10	Distributed Systems	-1.22E-05	0.0258	Tightly Coupled	3.50E-19	5.0000	Directed Diffusion	0.4230	355.6

By sorting the temporal degree of the three indices with ascending order, we got the top ten subsiding phrases in the four conferences as shown in above Table.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In this project work, Classification of Emergent and subsiding trends by the Dynamic and Adaptive Self Organizing Maps (DASOM) Neural Network were proposed. To get more accurate classification results, this method uses Particle Swarm Optimization (PSO), an Optimization Algorithm in combination with DASOM. Finally the two desired clusters namely the Emergent and subsiding clusters are formed. The given input terms which are actually emergent are mostly classified into the Emergent Cluster and likewise remaining terms into Subsiding Term. For instance, when 60% of dataset is taken for training and 40% is given for testing purpose, DASOM produces Classification Accuracy of 82% and DASOM with PSO produces 92% of Classification Accuracy which is 30-40% higher than basic SOM algorithms. The results of the proposed model are consistent and hence encouraging.

5.2 Future Scope:

The Dynamic and Adaptive Self Organizing Maps (DASOM) with Particle Swarm Optimization (PSO) classifies the clusters more precisely than its basic variant Self Organizing Maps(SOM).Our Model proves the Better Classification Accuracy in the range 30% to 40% more when compared to conventional Self Organizing Maps(SOM).This model can be trained with huge dataset to get cent percent result and can be further developed for predicting Emergent and Subsiding Trends for a given Year.

APPENDIX 1
SCREENSHOTS

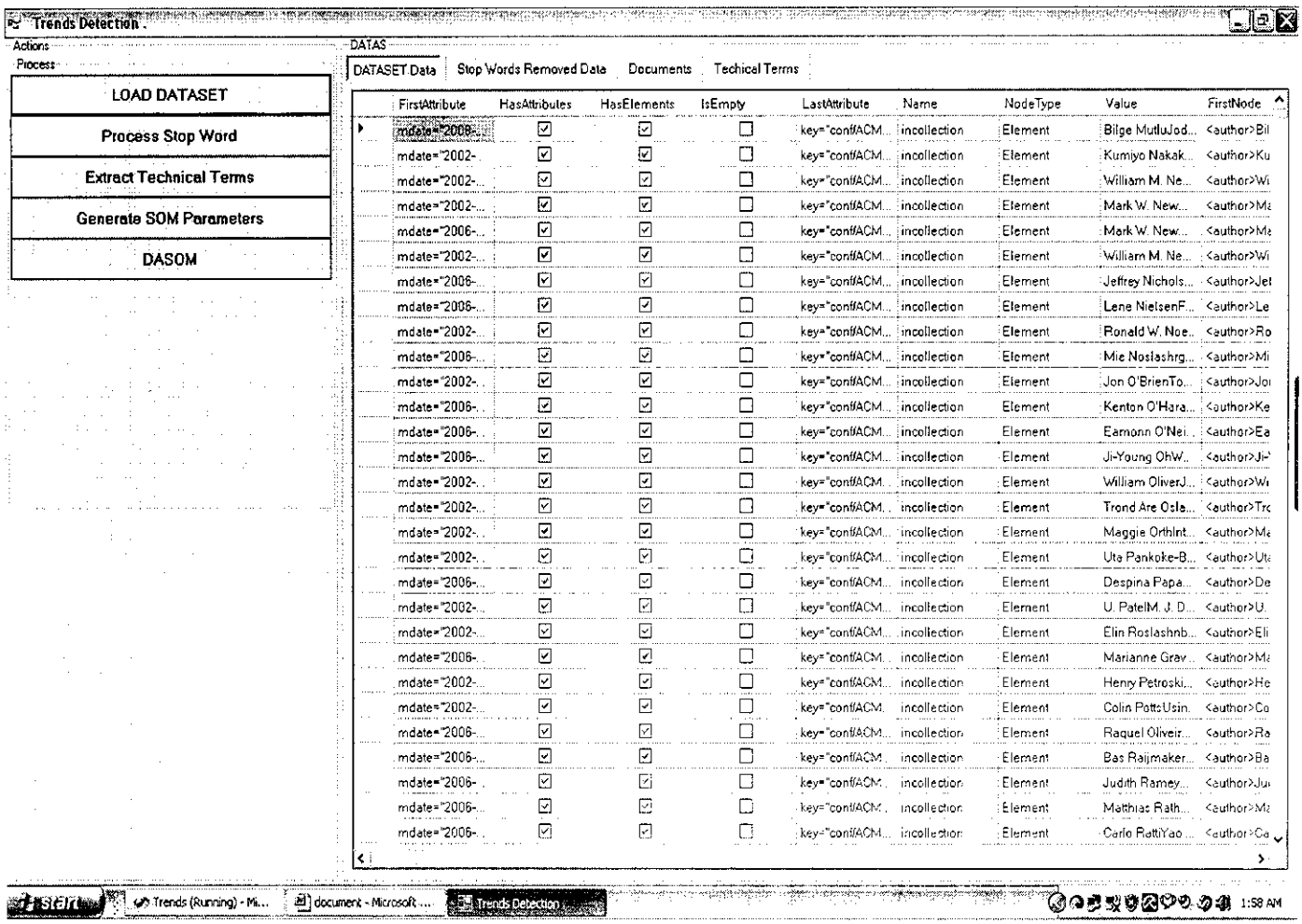

Figure 11. Showing Loaded Dataset

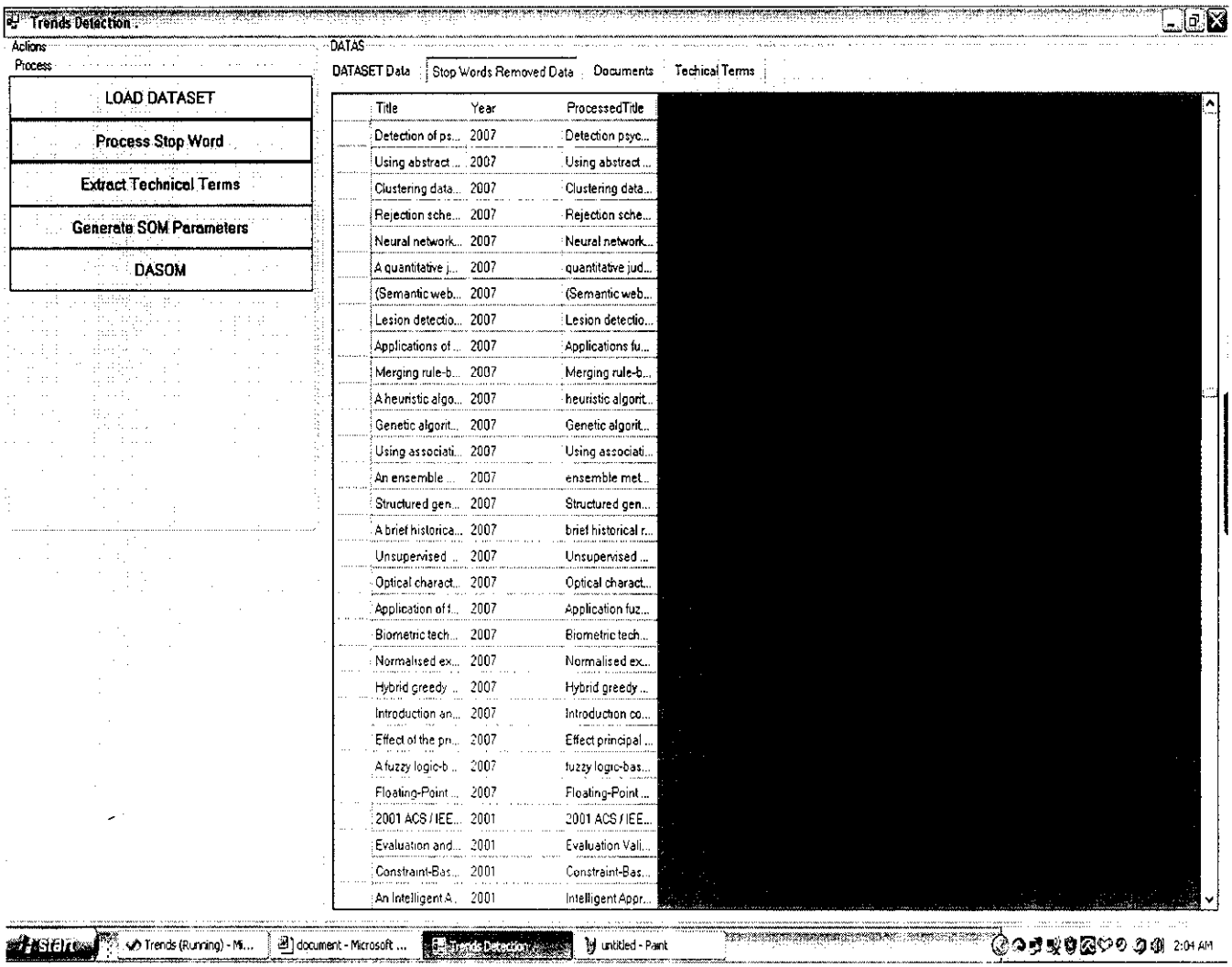

Figure 12. Showing Stop Words Removed Data

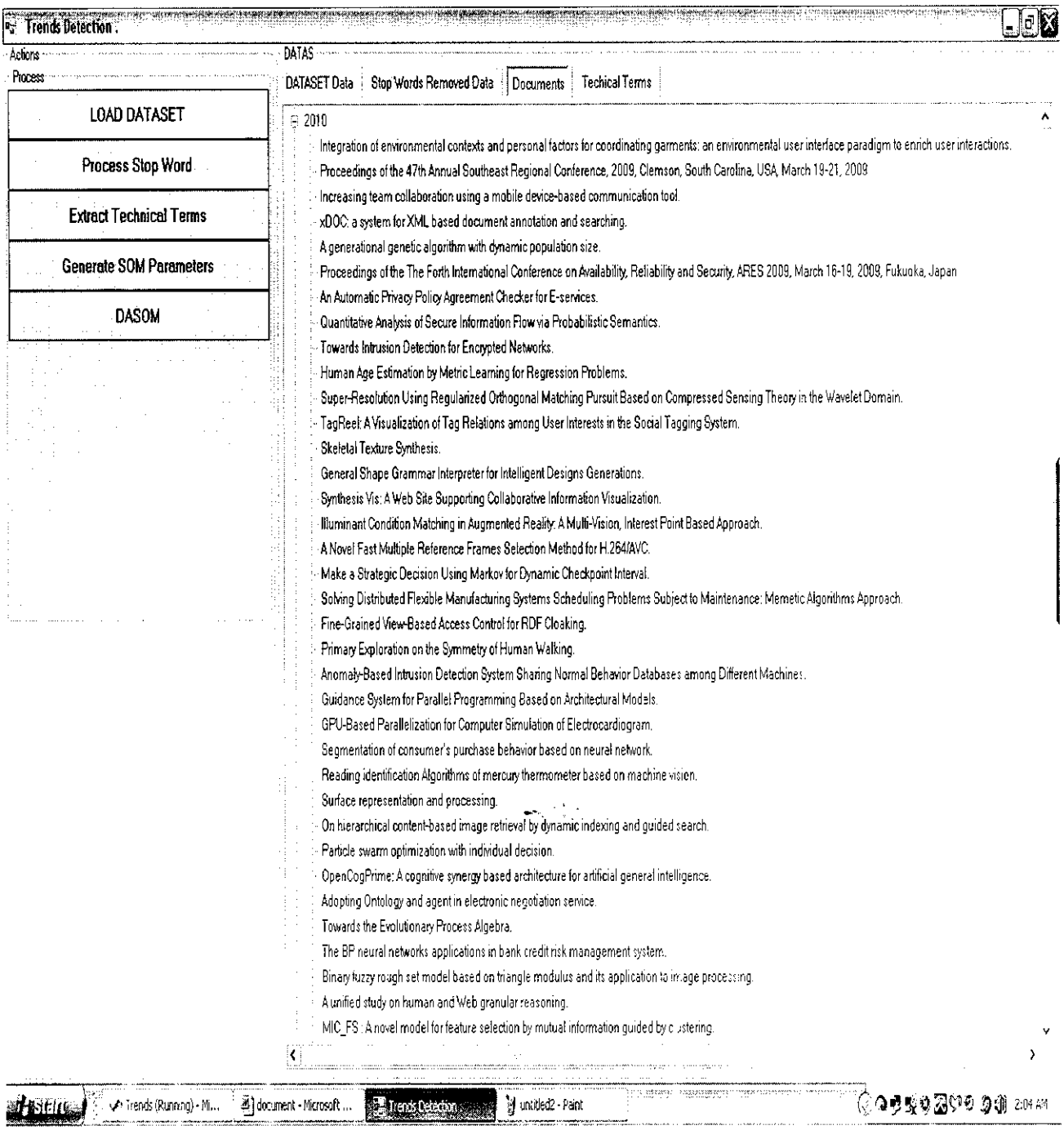

Figure 13. Showing documents classified year wise

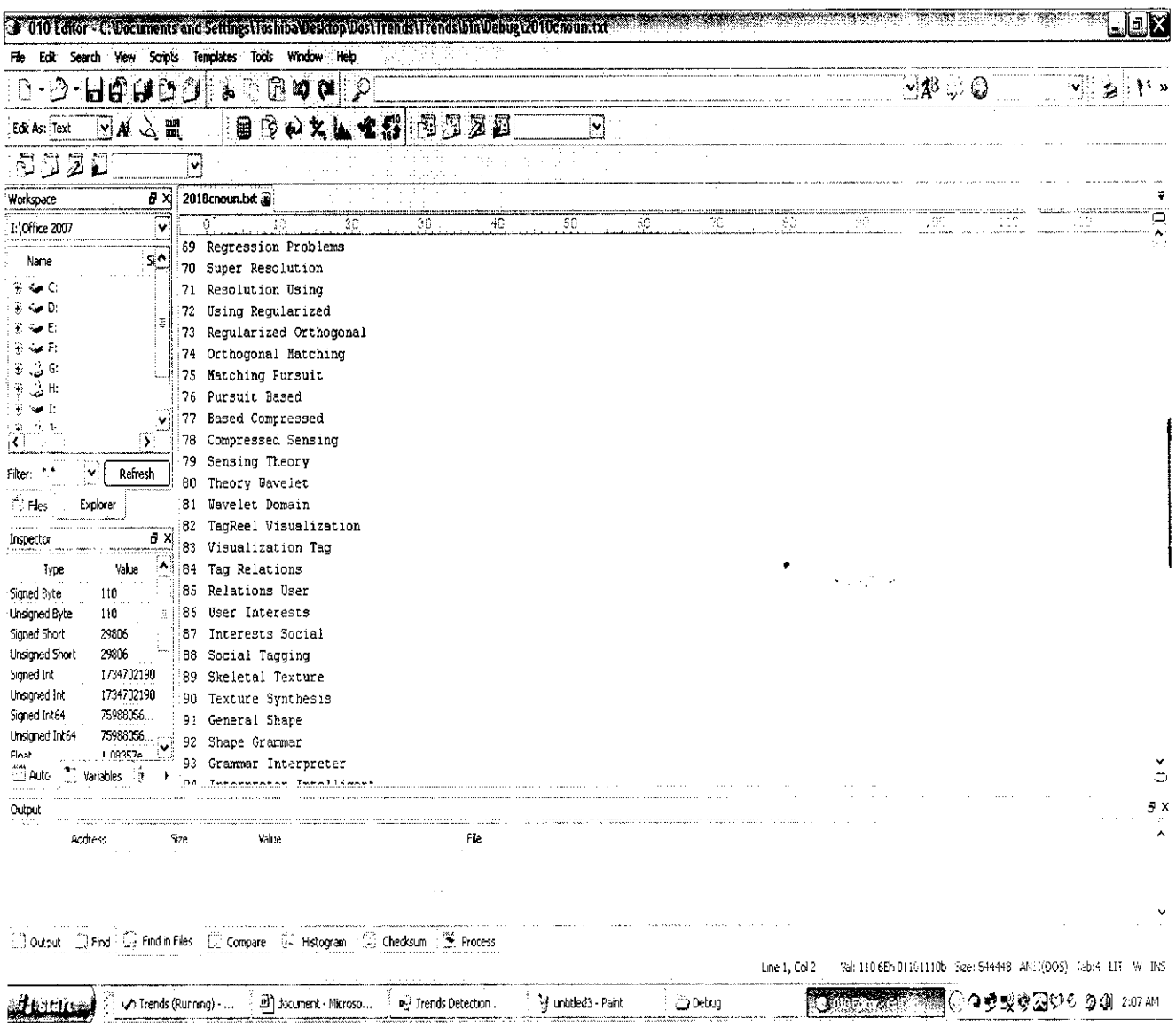

Figure 14. Showing Compound Nouns

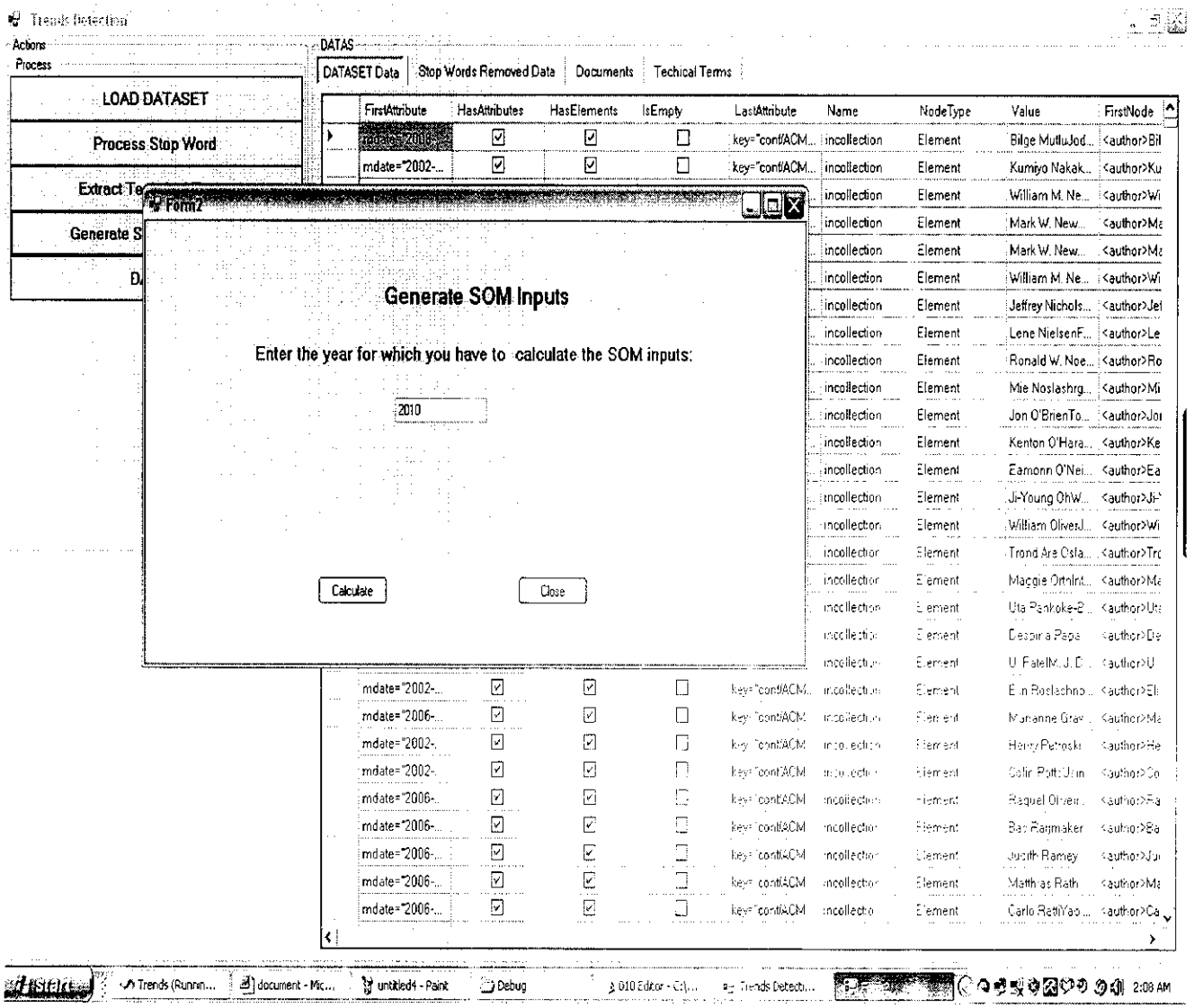

Figure 15. Showing DASOM Parameter Calculator

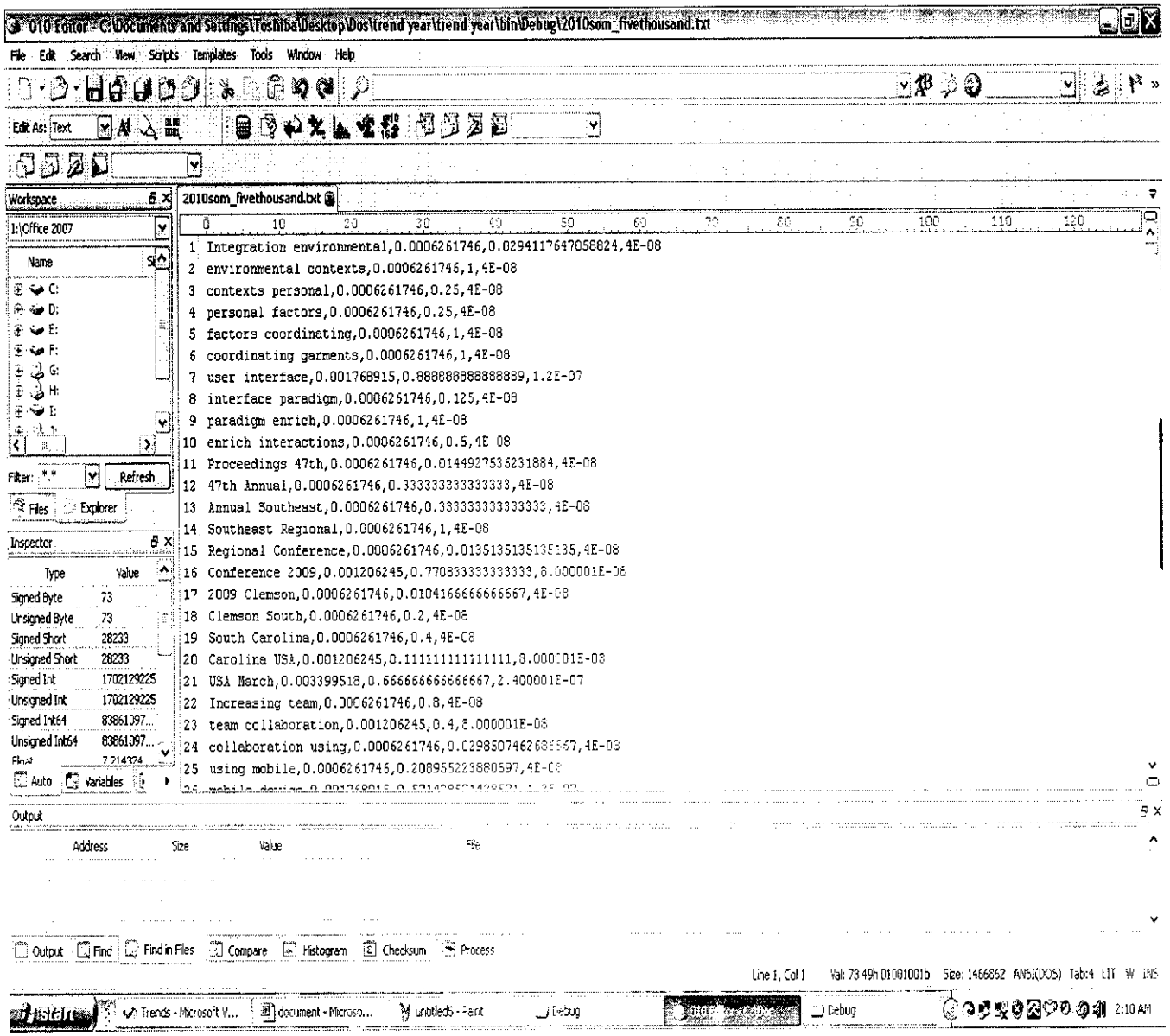

Figure 16. Showing Input to DASOM and DASOM with PSO

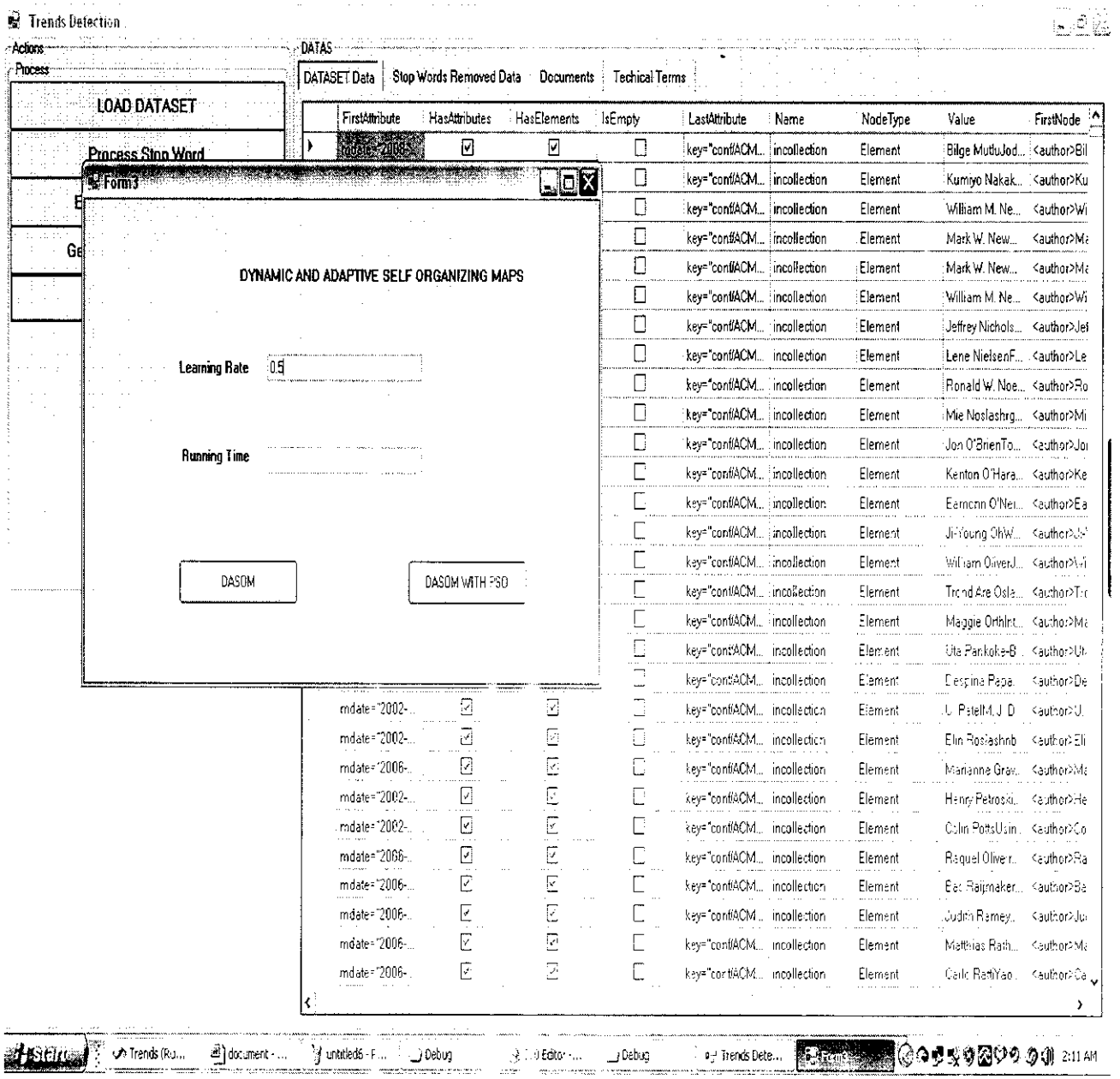

Figure 17. Showing DASOM Algorithm Execution after Learning Rate Parameter set

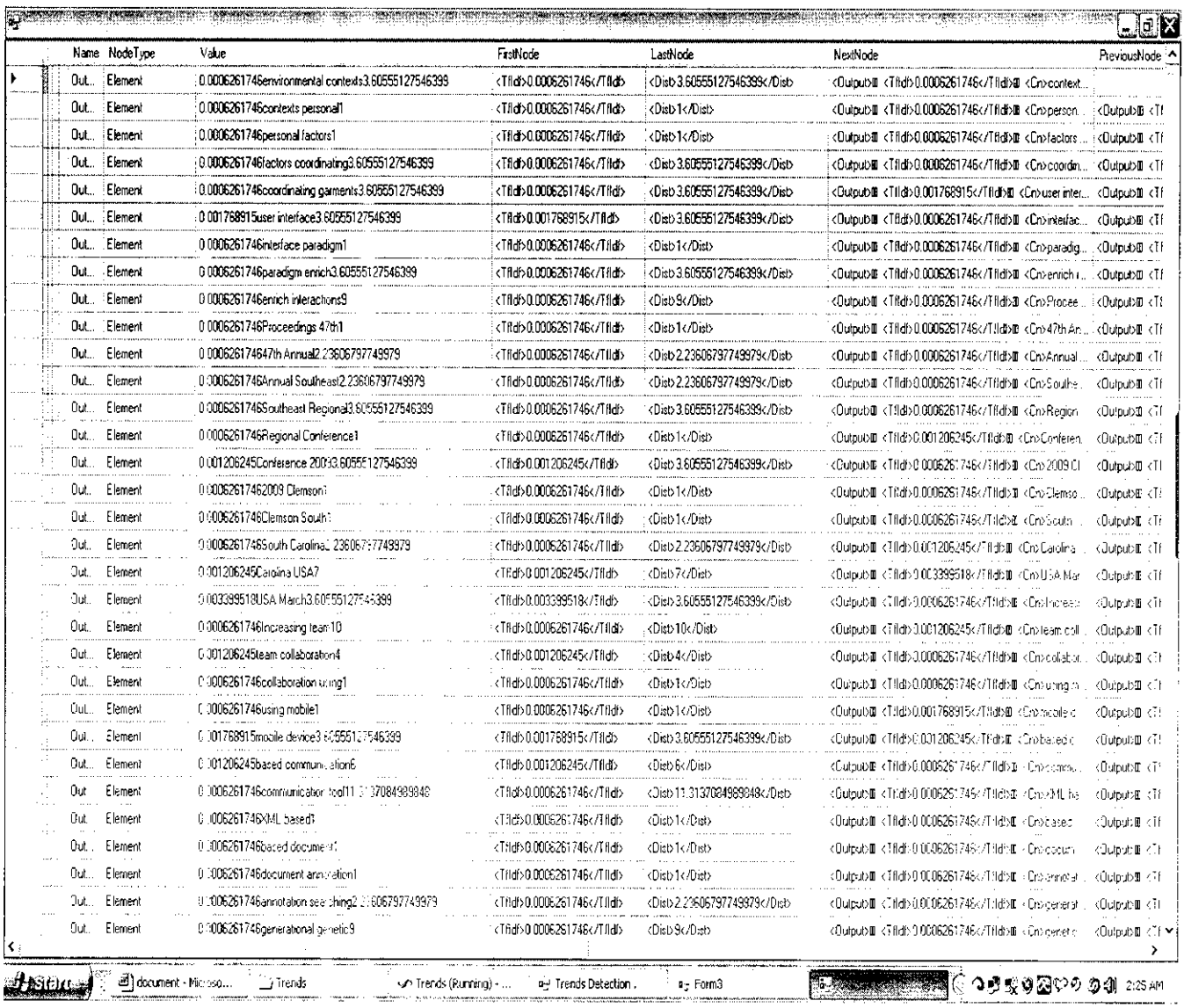

Figure 18. Showing output of DASOM Algorithm

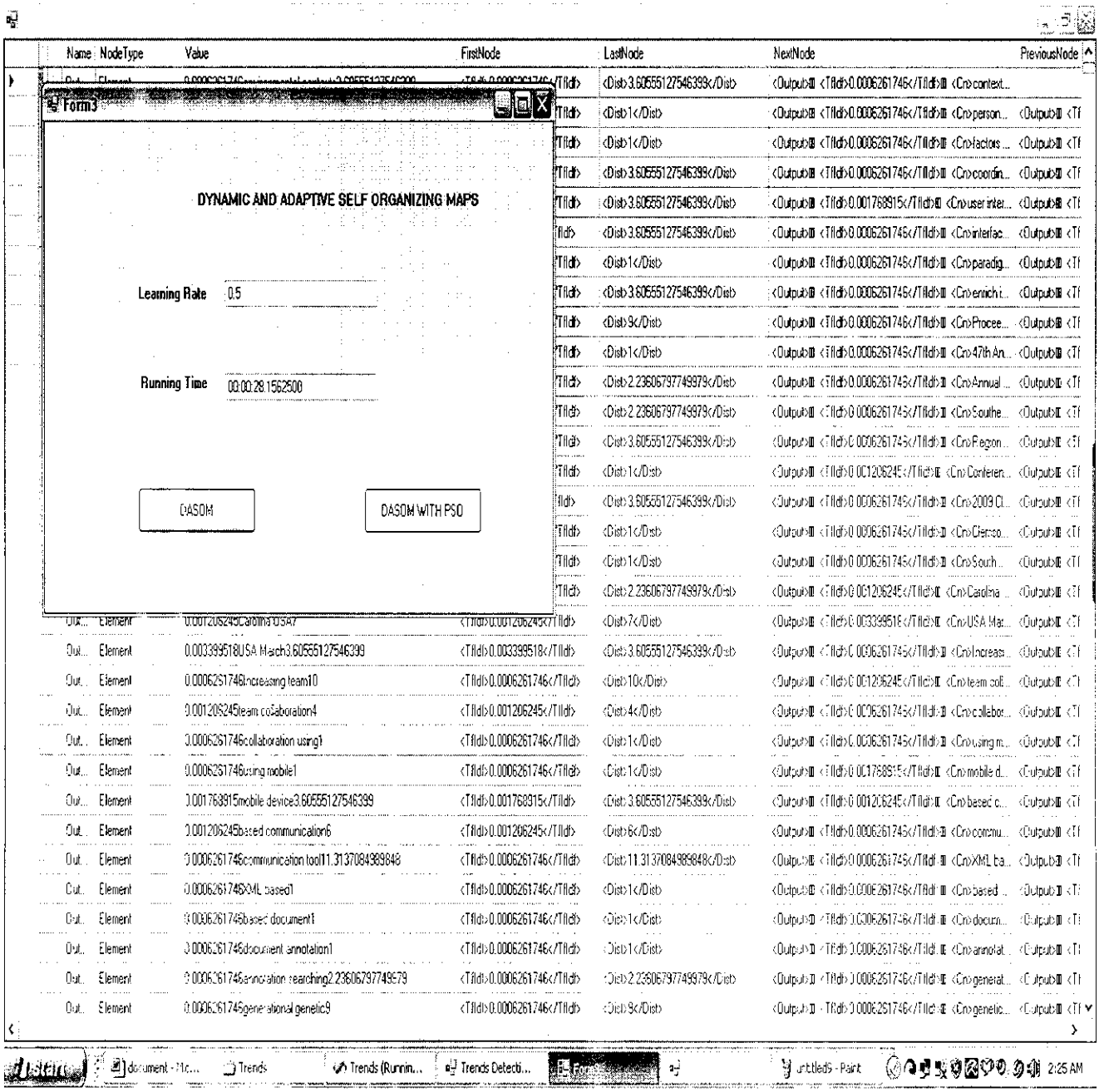
Form3
DYNAMIC AND ADAPTIVE SELF ORGANIZING MAPS
Learning Rate
0.5
Running Time
00:00:28.1562500
DASOM
DASOM WITH PSO
Name
NodeType
Value
FirstNode
LastNode
NextNode
PreviousNode

Figure 19. showing running time of DASOM Algorithm

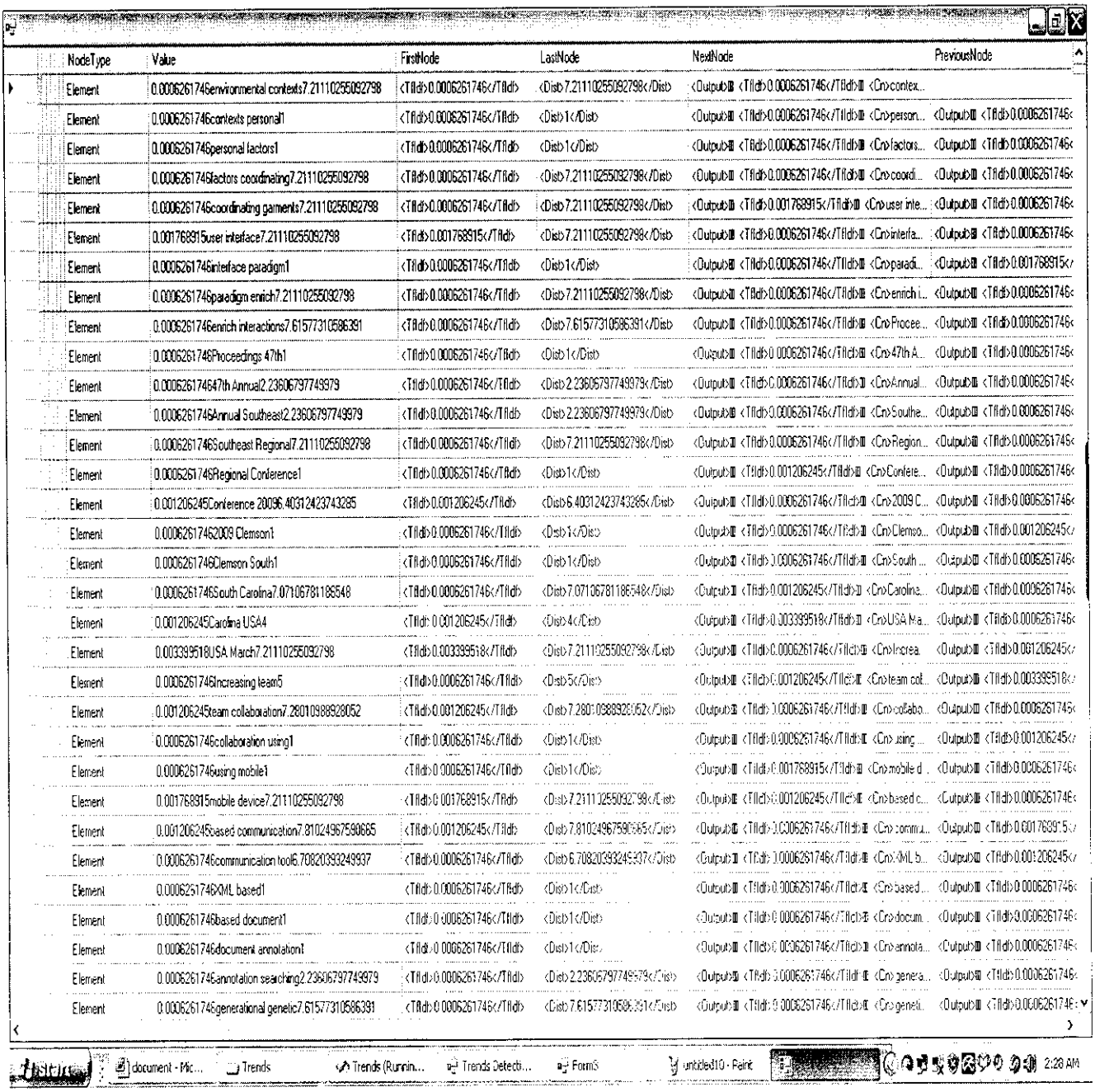

Figure 20. Showing DASOM with PSO Algorithm Execution after Learning Rate Parameter set

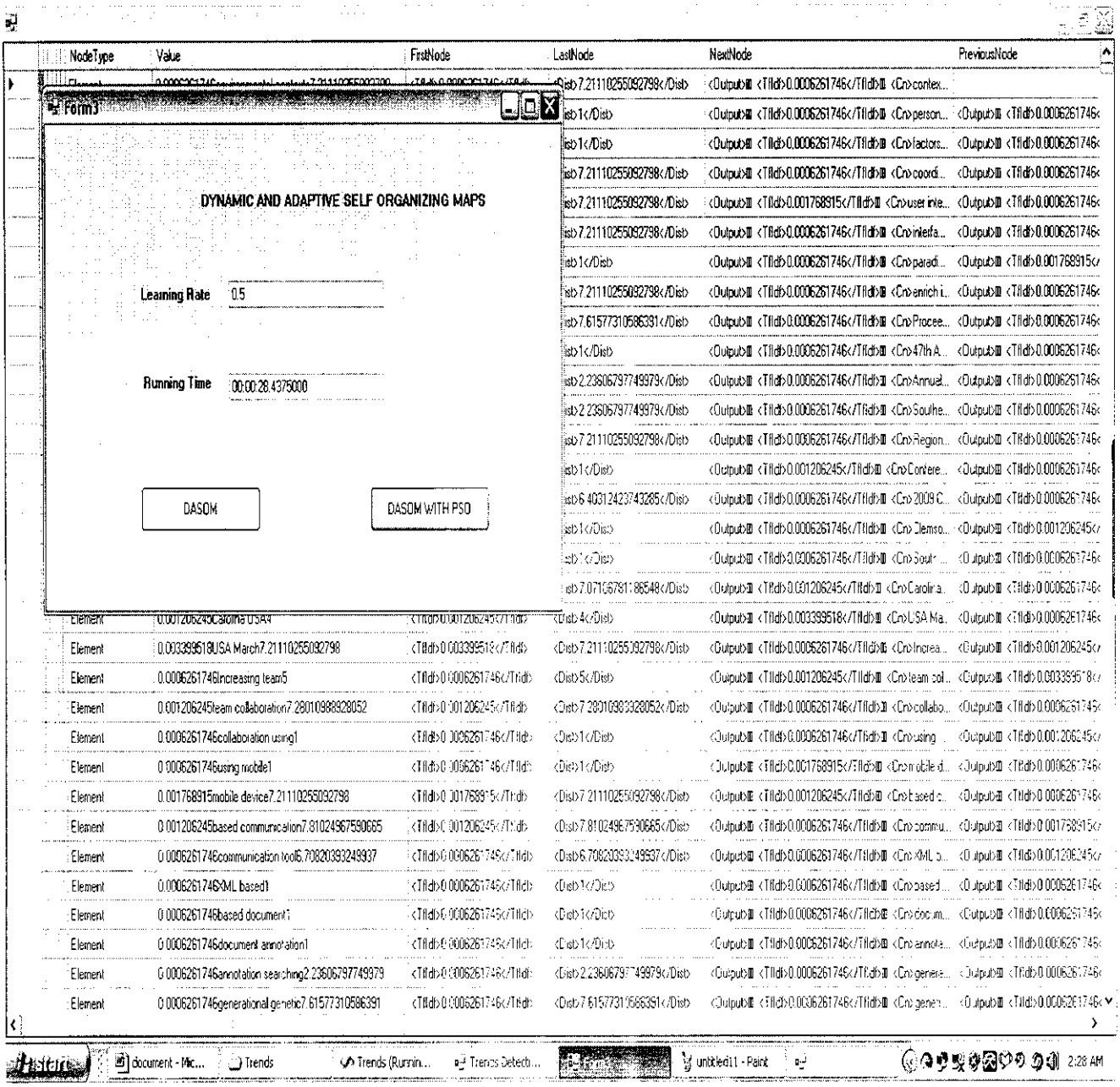

Figure 21. Showing running time of DASOM Algorithm

APPENDIX II
SAMPLE CODING

Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Trends
{
  static class Program
  {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new Form1());
    }
  }
}
```

Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml.Linq;
using System.IO;

namespace Trends
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }


    public static int yrcnt = 0;

    public static int[] yer = new int[40];



    private void button1_Click(object sender, EventArgs e)
    {

      XElement X = XElement.Load(@"TestData0.xml");
      var LoadedData = from k in X.Elements() select k;
      dataGridView1.DataSource = LoadedData.ToList();


    }
```

```
List<TrendType> Titles = new List<TrendType>();

private void button2_Click(object sender, EventArgs e)
{

   XElement X = XElement.Load(@"TestDatao.xml");

   var LoadedData = from k in X.Elements("incollection") select new { C =
(string)k.Attribute("key").Value, T = (string)k.Element("title").Value, yr =
(string)k.Element("year").Value };


   foreach (var s in LoadedData)
   {
      TrendType Temp = new TrendType();
      Temp.Title = s.T;
      Temp.Year = s.yr;

      Temp.ProcessedTitle = ProcessStopWord.RemoveStopwords(s.T);
      Titles.Add(Temp);


   }



   TextWriter tw = new StreamWriter(@"C:\Documents and
Settings\Toshiba\Desktop\Dos\Trends\Trends\stopjaccard.txt");
```

```
            var docu = from dk in Titles group dk by dk.Year into g select new { key
= g.Key, datas = g };
            foreach (var d in docu)
            {

               treeView1.Nodes.Add(d.key, d.key);

               foreach (var dc in d.datas)
               {
                  string styrt = dc.Year + "title.txt";
                  string styrs = dc.Year + "swrd.txt";
                  TextWriter tyrt = new StreamWriter(styrt,true);
                  TextWriter tyrs = new StreamWriter(styrs,true);
                  treeView1.Nodes[d.key].Nodes.Add(dc.Title);
                  tyrt.WriteLine(dc.Title);
                  tyrs.WriteLine(dc.ProcessedTitle);
                  tyrs.Close();
                  tyrt.Close();

               }
            }
            dataGridView2.DataSource = Titles.ToList();



         }
```

```
static List<TechnicalTerm> Lists = new List<TechnicalTerm>();

private void button3_Click(object sender, EventArgs e)
{
  TextWriter tw1 = new StreamWriter(@"C:\Documents and Settings\Toshiba\Desktop\Dos\Trends\Trends\compoundnoun.txt");

    foreach (TrendType str in Titles)
    {

      string[] source = str.ProcessedTitle.Split(new char[] { '.', '?', '!', ' ', ';', ':', ',', '-' }, StringSplitOptions.RemoveEmptyEntries);
      var matchQuery = from word in source
                       select word;
      int wordCount = matchQuery.Count();
      int wc = wordCount;
      var wordmatches = matchQuery.ToList();
      for (int i = 0; i <= wordmatches.Count - 1; i++)
      {
        string s = wordmatches[i].ToString().Trim();
        if ((i > 0) && (i != wordCount))
        {
          if (s.Length > 2 && wordmatches[i - 1].Length > 2)
          {
            s.Replace('(', ' ');
            TechnicalTerm TempT = new TechnicalTerm();
```

```
                    TechnicalTerm TempT1 = new TechnicalTerm();
                    TempT.Terms = wordmatches[i - 1] + " " + s;

                    TempT.Year = str.Year;
                    Lists.Add(TempT);



                    tw1.WriteLine(TempT.Terms);

                }
            }
            else
            {



            }
        }



    }

    var docu = from dk in Lists group dk by dk.Year into g select new
{ key = g.Key, datas = g };
    foreach (var d in docu)
    {
```

```
foreach (var dc in d.datas)
{
   string styrc = dc.Year + "cnoun.txt";
   int f=0;
   TextWriter tyrc = new StreamWriter(styrc, true);

   tyrc.WriteLine(dc.Terms);
   tyrc.Close();
   int intyr = Convert.ToInt16(dc.Year);
   if (yrcnt.Equals(0))
   {
      yer[yrcnt] = intyr;
      yrcnt++;
   }

      for (int l = 0; l <yer.Length; l++)
      {
         if (yer[l].Equals(intyr))
         {
            f = 1;


         }


      }

      if (f.Equals(0))
      {
         yrcnt++;
         yer[yrcnt]= intyr;
      }
```

```
        }
      }
    tw1.Close();



      dataGridView3.DataSource = Lists.ToList();
}

private void button4_Click(object sender, EventArgs e)
{



}

private void button4_Click_1(object sender, EventArgs e)
{
   trend_year.som g = new trend_year.som();
   g.Show();
}

private void groupBox3_Enter(object sender, EventArgs e)
{


}

private void button5_Click(object sender, EventArgs e)
{
   Form3 g = new Form3();
   g.Show();
}
```

```
    }
}
```

Form2.cs(Trend Year-DASOM Parameters):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Xml.Linq;


namespace trend_year
{
  public partial class som : Form
  {
    public som()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      int doccnt;
```

```
float docfeq;
float invdocfreq;
float terfreq;
float tfidf;
float odd;
double jac;
string tfidfs;
string[] words = new string[2];
string twy = textBox1.Text + "som.txt";
string str = textBox1.Text + "cnoun.txt";
string dy = textBox1.Text + "swrd.txt";
TextWriter tw = new StreamWriter(twy);
XElement XDoc = XElement.Load("XMLFile1.xml");
StreamReader re = File.OpenText(str);
string input = null;
while ((input = re.ReadLine()) != null)
{
   doccnt = 1;//GetDocCnt(input,dy);
   if (doccnt.Equals(1))
   {
      docfeq = GetDocFreq(input, dy);
      invdocfreq = GetInverseDocumentFrequency(docfeq, str);
      terfreq = GetTermFrequency(input, str);
      tfidf = terfreq * invdocfreq;
      words = input.Split(' ');
      jac = jaccard(words[0], words[1], dy);
      odd = odds(docfeq, dy);
      tfidfs = tfidf.ToString();

      if (!tfidfs.Equals("Infinity"))
      {
```

```
            tw.WriteLine(input + "," + tfidf + "," + jac + "," + odd);


            XElement X = new XElement("Trend",
               new XAttribute("Year", textBox1.Text),
               new XAttribute("Term", input),
               new XAttribute("TfIdf", tfidf),
               new XAttribute("Jac", jac),
               new XAttribute("Odds", odd));
            XDoc.Add(X);
        }
    }

}
tw.Close();
//re.close();

XDoc.Save("XMLFile1.xml");



TextWriter twt = new StreamWriter("Tfidf.txt");
XElement XDoct = XElement.Load("XMLFile1.xml");
var tst = from k in XDoct.Elements("Trend")
          group k by k.Attribute("TfIdf").Value into G
          orderby G.Key descending
          select new
          {
             Key = G.Key,
             Items = from kk in G
```

```
                    orderby kk.Attribute("TfIdf").Value descending
                    select new { Year = (string)kk.Attribute("Year").Value,
Term = (string)kk.Attribute("Term").Value, TfIdf =
(string)kk.Attribute("TfIdf").Value }
                };
            //treeView2.Nodes.Add("Emergent Terms").ForeColor = Color.Red;
            foreach (var t in tst)
            {
              // treeView2.Nodes.Add(t.Key, t.Key);
              var tt = t.Items.Take(10);
              int count = t.Items.Count();
              foreach (var v in tt)
              {
                //treeView2.Nodes[t.Key].Nodes.Add(v.Term + " [ " +
v.Conference + " ].").ForeColor = Color.Green;
                twt.WriteLine(v.Term + "," + v.TfIdf);

              }

            }

            MessageBox.Show("You are Done!!!!!!");
        }


        public int GetDocCnt(string term, string y)
        {
            int cnt = 0; ;
            //string y = "1993swrd.txt";
            string[] lines = System.IO.File.ReadAllLines(y);
            //string word = lines[term];
```

```
    //len = lines.GetLength(1);



    foreach (string line in lines)
    {
        if (line.Contains(term))
        {

            cnt++;
        }

    }
    if (cnt > 2)
        return 1;
    else
        return 0;
}

public float GetDocFreq(string term, string y)
{
    int cnt = 0; ;
    //string y = "1993swrd.txt";
    string[] lines = System.IO.File.ReadAllLines(y);
    //string word = lines[term];
    //len = lines.GetLength(1);

    int count = System.IO.File.ReadAllLines(y).Length;

    foreach (string line in lines)
    {
```

```
        if (line.Contains(term))
        {
            cnt++;
        }

    }
    /* int freq = _termFreq[term][doc];
     int maxfreq = _maxTermFreq[doc];*/

    return (float)cnt / (float)count;

}

public float GetTermFrequency(string term, string y)
{
    // int len;
    int cnt = 0; ;
    //string y = "1993cnoun.txt";
    string[] lines = System.IO.File.ReadAllLines(y);
    //string word = lines[term];
    //len = lines.GetLength(1);

    int count = System.IO.File.ReadAllLines(y).Length;

    foreach (string line in lines)
    {
        if (line.Contains(term))
        {
            cnt++;
        }
```

```
    }
    /* int freq = _termFreq[term][doc];
     int maxfreq = _maxTermFreq[doc];*/

    return (float)cnt / (float)count;
}

public float GetInverseDocumentFrequency(float df, string ye)
{
    float y = 0;
    //string ye = "1993swrd.txt";
    //string[] lines = System.IO.File.ReadAllLines(ye);
    //string word = lines[term];
    //len = lines.GetLength(1);

    int count = System.IO.File.ReadAllLines(ye).Length;

    try
    {
        //int df = _docFreq[term];
        y = (float)Math.Log((float)(count) / (float)df);
    }
    catch (Exception ex)
    {
    }
    return y;


}
```

```
public double jaccard(string s1, string s2, string y)
{
  //List s1 = CreateShingles(page1);
  //List s2 = CreateShingles(page2);

  // var commonelements = s1.Intersect(s2);
  // var totalElements = s1.Union(s2);
  int cnt1 = 0;
  int cnt2 = 0;
  int max;
  int min;
  string[] lines1 = System.IO.File.ReadAllLines(y);
  foreach (string line1 in lines1)
  {


    if (line1.Contains(s1))
    {
      cnt1++;
    }
    if (line1.Contains(s2))
    {
      cnt2++;
    }

  }

  lines1 = null;

  if (cnt1 > cnt2)
```

```
            {
                max = cnt1;
                min = cnt2;
            }
            else
            {
                max = cnt2;
                min = cnt1;
            }
            double jaccardCoef = Convert.ToDouble(min) /
Convert.ToDouble(max);

            return jaccardCoef;
        }



        public float odds(float df, string y)
        {
            float newodd;
            //string y ="1993swrd.txt";
            int count = System.IO.File.ReadAllLines(y).Length;

            newodd = df / (count - df);

            return newodd;
        }

    }
}
```

Form3.cs(DASOM and DASOM with PSO):

```
class Mappso
{
private Neuron[,] outputs;  // Collection of weights.
private int iteration;      // Current iteration.
private int length;         // Side length of output grid.
private int dimensions;     // Number of input dimensions.
private Random rnd = new Random();

private List<string> labels = new List<string>();
private List<double[]> patterns = new List<double[]>();
double[] inputbo = new double[75000];
private List<int[]> opt = new List<int[]>();
TextWriter tw = new StreamWriter(@"twod.xml");
TextWriter wo = new StreamWriter(@"cool.txt");
TextWriter t = new StreamWriter(@"row.txt");
/* static void Main(string[] args)
{

  //Console.ReadLine();
}*/

public Mappso(int dimensions, int length, string file,double d)
{

  this.length = length;
  this.dimensions = dimensions;
  Initialise();
  LoadData(file);
```

```
    NormalisePatterns();
    Train(0.0000001,d);
    DumpCoordinates();
}

private void Initialise()
{
    StreamReader reader = File.OpenText("pso_dasom.txt");
    reader.ReadLine();
    string[] line = reader.ReadLine().Split(',');
    Random rand = new Random();
    outputs = new Neuron[length, length];
    for (int i = 0; i < length; i++)
    {
        for (int j = 0; j < length; j++)
        {
            outputs[i, j] = new Neuron(i, j, length);
            outputs[i, j].Weights = new double[dimensions];
            for (int k = 0; k < dimensions; k++)
            {
                outputs[i, j].Weights[k] = Convert.ToDouble(line[k]);
                //outputs[i, j].Weights[k] = rand.NextDouble();
                //t.WriteLine("{0}", outputs[i, j].Weights[k]);
            }
        }
    }
    wo.Flush();
    t.Flush();
}

private void LoadData(string file)
```

```
{
   StreamReader reader = File.OpenText(file);
   reader.ReadLine(); // Ignore first line.
   int m = 0;
   while (!reader.EndOfStream)
   {
      string[] line = reader.ReadLine().Split(',');
      labels.Add(line[0]);
      double[] inputs = new double[dimensions];
      for (int i = 0; i < dimensions; i++)
      {
         inputs[i] = double.Parse(line[i + 1]);
      }
      inputbo[m] = double.Parse(line[1]);
      patterns.Add(inputs);
      m++;
   }
   reader.Close();
}

private void NormalisePatterns()
{
   for (int j = 0; j < dimensions; j++)
   {
      double sum = 0;
      for (int i = 0; i < patterns.Count; i++)
      {
         sum += patterns[i][j];
      }
      double average = sum / patterns.Count;
      for (int i = 0; i < patterns.Count; i++)
```

```
        {
            patterns[i][j] = patterns[i][j] / average;
        }
    }
}

private void Train(double maxError,double d)
{
    double currentError = double.MaxValue;
    while (currentError > maxError)
    {
        currentError = 0;
        List<double[]> TrainingSet = new List<double[]>();

        foreach (double[] pattern in patterns)
        {
            TrainingSet.Add(pattern);
        }
        //  for (int h = 0; h < 6; h++)
        //{
        for (int i = 0; i < (patterns.Count); i++)
        {
            double[] pattern = TrainingSet[rnd.Next((patterns.Count) - i)];
            currentError += TrainPattern(pattern,d);
            TrainingSet.Remove(pattern);
        }
        //    foreach (double[] pattern in patterns)
        // {
        //    TrainingSet.Add(pattern);
        //}
        //}
```

```
            //tw.WriteLine(currentError.ToString("0.0000000"));
            //  Console.WriteLine(currentError.ToString("0.0000000"));
            //t.WriteLine("{0}", "----------delimiter----------");
            //h++; l++;
          }
        }

        private double TrainPattern(double[] pattern,double d)
        {
          double error = 0, dd = 0;
          Neuron winner = Winner(pattern);
          winner.UpdatewinnerWeight(pattern, winner, iteration,d);
          for (int i = 0; i < length; i++)
          {
            for (int j = 0; j < length; j++)
            {
              dd = sim(outputs[i, j].Weights, winner.Weights); error += outputs[i,
j].UpdateWeights(pattern, winner, iteration, dd,d);
            }
          }
          iteration++;
          //t.WriteLine("{0}", iteration);
          return Math.Abs(error / (length * length));
        }

        private void DumpCoordinates()
        {
          System.Random RandNum = new System.Random(); double v;
          tw.WriteLine("{0}", "<dasom>");
          for (int i = 0; i < patterns.Count; i++)
```

```
    {
        Neuron n = Winnerup(patterns[i]);
        double MyRandomNumber = RandNum.NextDouble();
        decimal fg = Convert.ToDecimal(n.X + MyRandomNumber);
        decimal x = Math.Round(fg, 2);
        decimal bo = Convert.ToDecimal(n.Y + MyRandomNumber);
        decimal y = Math.Round(bo, 2);
        v = dist(n.X, n.Y);
        wo.WriteLine("{0},{1}", v, labels[i]);
        tw.WriteLine("{0}", "<Output>");
        tw.WriteLine("{0}{1}{2}", "<Tfldf>", inputbo[i], "</Tfldf>");
        tw.WriteLine("{0}{1}{2}", "<Cn>", labels[i], "</Cn>");
        tw.WriteLine("{0}{1}{2}", "<Dist>", v, "</Dist>");
        tw.WriteLine("{0}", "</Output>");
        // Console.WriteLine("{0},{1}", x, y);
    }
    tw.WriteLine("{0}", "</dasom>");
    tw.Flush();
    wo.Flush();
    t.Flush();
    tw.Close();
}

private Neuron Winner(double[] pattern)
{
    Neuron winner = null;
    double max = double.MaxValue;
    for (int i = 0; i < length; i++)
        for (int j = 0; j < length; j++)
        {
            // double d = Distance(pattern, outputs[i, j].Weights);
```

```
            double d = sim(pattern, outputs[i, j].Weights);
            if (d < max)
            {
               max = d;
               winner = outputs[i, j];
            }
         }
      //   if (max >= 0.5)
      t.WriteLine("{0}", max);
      return winner;
      /* else
       {
         Console.WriteLine("ooooooo...");
         return winner;
       }*/
   }
   private Neuron Winnerup(double[] pattern)
   {
      Neuron winner = null;
      double max = double.MaxValue;
      for (int i = 0; i < length; i++)
         for (int j = 0; j < length; j++)
         {
            double d = Distance(pattern, outputs[i, j].Weights);
            // double d = sim(pattern,outputs[i, j].Weights);
            if (d < max)
            {
               max = d;
               winner = outputs[i, j]:
            }
         }
```

```
    //    if (max >= 0.5)
    return winner;
    /* else
     {
        Console.WriteLine("ooooooo...");
        return winner;
     }*/
  }

  private double sim(double[] pattern, double[] p)
  {
    double value = 0;
    double arr = 0, rew = 0;
    double arrl = 0;
    for (int i = 0; i < pattern.Length; i++)
    {
      value += (pattern[i] * p[i]);
      arr += Math.Pow(pattern[i], 2);
      arrl += Math.Pow(p[i], 2);
    }
    rew = value / (Math.Sqrt(arr) * Math.Sqrt(arrl));
    //Console.WriteLine(rew);
    //Console.ReadLine();
    return rew;
  }
  private double Distance(double[] vector1, double[] vector2)
  {
    double value = 0, l = 0;
    for (int i = 0; i < vector1.Length; i++)
    {
      value += Math.Pow((vector1[i] - vector2[i]), 2);
```

```
    }
    l = Math.Sqrt(value);
    //Console.WriteLine(l);
    //        Console.ReadLine();
    return l;
  }
  private double dist(double x, double y)
  {
    double value = 0, l = 0;
    value = Math.Pow((0 - x), 2);
    value += Math.Pow((0 - y), 2);
    l = Math.Sqrt(value);
    //Console.WriteLine(l);
    //        Console.ReadLine();
    return l;
  }

public class Neuron
{
  public double[] Weights;
  public double X;
  public double Y;
  private int length;
  private double nf;
  public Neuron(int x, int y, int length)
  {
    X = x; Y = y;
    this.length = length;
    nf = 1000 / Math.Log(length);
  }
```

```
        private double Gauss(Neuron win, int it)
        {
          double distance = Math.Sqrt(Math.Pow(win.X - X, 2) +
Math.Pow(win.Y - Y, 2));
          return Math.Exp(-Math.Pow(distance, 2) / (Math.Pow(Strength(it), 2)));
        }
        private double LearningRate(int it, double dd,double d)
        {
          return ((Math.Exp(-it / 1000) * d) * dd);//((0.9 * (1 - (it / 992))));
        }
        private double LearningRate(int it,double d)
        {
          return (Math.Exp(-it / 1000) * d);//((0.9 * (1 - (it / 992))));
        }
        private double Strength(int it)
        {
          return Math.Exp(-it / nf) * length;
        }

        public double UpdateWeights(double[] pattern, Neuron winner, int it,
double dd.double d)
        {
          double sum = 0;
          for (int i = 0; i < Weights.Length; i++)
          {
            double delta = LearningRate(it, dd,d) * Gauss(winner, it) * (pattern[i]
- Weights[i]);
            Weights[i] += delta;
            sum += delta;
```

```
            }
            return sum / Weights.Length;
        }
        public double UpdatewinnerWeight(double[] pattern, Neuron winner, int
it,double d)
        {
            double sum = 0;
            for (int i = 0; i < winner.Weights.Length; i++)
            {
                double delta = LearningRate(it,d) * Gauss(winner, it) * (pattern[i] -
winner.Weights[i]);
                winner.Weights[i] += delta;
                sum += delta;
            }
            return sum / winner.Weights.Length;
        }
    }
}
}
```

Gridview.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml.Linq;
```

```
namespace Trends
{
  public partial class Form4 : Form
  {
    public Form4()
    {
      InitializeComponent();
      XElement X = XElement.Load(@"twod.xml");
      var LoadedData = from k in X.Elements() select k;
      dataGridView2.DataSource = LoadedData.ToList();
    }

    private void dataGridView2_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {

    }
  }
}
```

Particle Swarm Optimization(PSO):

RunPso.m

```
function RunPso(obj_func, nParticals, MinVars, MaxVars)

%Initialize a particle swarm with nParticals
for i=1:nParticals
  x{i}=RandInRange(MinVars, MaxVars);
end;
%disp(x);
```

```
global gBestX; %Global Best Design vector
global lBestX;  %Local Best Design vector


global V;      % Velosity
global gBestY;  % output

gBestY=1e6;
lBestX=[];
gBestX=[];

x=RunMoveOnce(x,obj_func)
while gBestY> 1e-6
   x=RunMoveOnce(x,obj_func);
   % disp ([gBestX, gBestY]);
end;
end
```

RunMoveOnce.m

```
function x=RunMoveOnce(x, obj_func)
global gBestX;  %Global Best Design vector, updated on each move
global lBestX;  %Local Best Design vector, updated on each move
global V;      % Velosity, updated on each move
global gBestY;  % Global best. may or may not be updated on each move

for i=1:length(x)
      err{i}=obj_func(x{i});
end;

ErrMat=cell2mat(err);   % Convert cell to array
minErr=min(ErrMat);     % Find the minimum
```

```
idx=find(ErrMat==minErr);  % Find which X is minimum

lBestX=x{idx};

if(isempty(gBestX))
   %For each particle, evaluate the objective function
   gBestX=x{idx};
   gBestY=minErr;
   x=x;
   V=0;
   return;
else
   if(minErr<gBestY)
      gBestX=x{idx};
      gBestY=minErr;
   end;

   alpha1=2.0;    % Local best influence;
   alpha2=2.0;    % Global best influence;
   Fai=0.9;       % inertia Weight;

   VMax=0.5;
   VMin=-0.5;
   Gamma1=rand(); %
   Gamma2=rand();

   for i=1:length(x)
      %V[k+1]= Fai * V[k] + alpha1 * Gamma1 * (lBestX-X(i)) ++ alpha2 *
      %Gamma2 * (gBestX-x{i}) :
      V=Fai*V + alpha1 * Gamma1 * (lBestX-x{i}) + alpha2 * Gamma2 *
(gBestX-x{i}) ; % update V;
```

```
        if(V>=VMax)
           V=VMax;
        elseif(V<=VMin)
           V=VMin;
        end;

        x{i}= x{i} + V;
      end;
    end;
    end
```

RandInRage.m:

```
function X=RandInRange(xMin, xMax)
% Generate a design vector range from MaxVars to MinVars;
n=length(xMin);

A=xMin;
B=xMax;
X=A + (B-A) .* rand(1,n)
end
```

ZeroFindObjFun.m:

```
function Y=ZeroFindObjFun(x)
%Minimize Y=(x+1)^2
%solution x=-1;
Y=10+(x^2-10*cos(2*3.14*x));
```

Main.m:

```
RunPso(@ZeroFindObjFun,4, -2,2, @ZeroFindplotUpdate_func);
```

6. REFERENCES:

[1] Hidenao Abe Shusaku Tsumoto, **Detection of Trends of Technical Phrases in Text Mining,** Shimane University, School of Medicine, IEEE Explore

[2] Zhonghui Feng, Junpeng Bao,Junyi Shen,(2010) **Dynamic and Adaptive Self Organizing Maps applied to High Dimensional Large Scale Text Clustering,**Institute of Computer Software ,IEEE

[3] Shangming Yang and Yi Zhang(2004)
Self-Organizing Feature Map Based Data Mining
University of Electronic Science and Technology of China,
Springer

[4] MASAHIRO ENDO, MASAHIRO UENO AND TAKAYA ANABE (2002) **A Clustering Method Using Hierarchical Self-Organizing Maps,**NTT Cyber Space Laboratories, Journal of VLSI Signal Processing

[5] Xinzheng Xu1, Wenhua Zeng2, and Zuopeng Zhao(2007)
A Structural Adapting Self-organizing Maps Neural Network
School of Computer Science and Technology,China University of Mining and Technology, Springer