VIETNAM NATIONAL UNIVERISTY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
ADVANCED PROGRAM IN INFORMATION SYSTEMS

TRẦN QUANG HÒA - VÕ HỒ TIẾN HƯNG

FINDING THE CLUSTER OF ACTORS IN SOCIAL NETWORK BASED ON THE TOPIC OF MESSAGES

BACHELOR OF ENGENEERING IN INFORMATION SYSTEMS

HOCHIMINH CITY, 2013

VIETNAM NATIONAL UNIVERISTY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
ADVANCED PROGRAM IN INFORMATION SYSTEMS

TRẦN QUANG HÒA - 09520092
VÕ HỒ TIẾN HƯNG - 09520121

FINDING THE CLUSTER OF ACTORS IN SOCIAL NETWORK BASED ON THE TOPIC OF MESSAGES

BACHELOR OF ENGENEERING IN INFORMATION SYSTEMS

THESIS ADVISOR
ASSOC. PROF. DR. ĐỖ PHÚC

HOCHIMINH CITY, 2013

ACKNOWLEDGEMENT

We would like to express our gratitude to our advisor Associate Professor Dr. Đỗ Phúc who suggests, guides and inspires us in working with this thesis.

We would like to thank Mr. Nguyễn Văn Muôn, Mr. Mai Xuân Hùng and Mr. Bashir Magomedov for their support, their knowledge of Author - Recipient - Topic Model and Kohonen Network.

We also appreciate several people in Social Network Analyst Group such as Mr. Nguyễn Lê Hoàng, Mr. Nguyễn Tiến Long and others for your assistance and advices.

We want to express our thankful to the staffs in faculty of Information Systems of University of Information Technology and our instructors who emit the passion to us.

Finally, we would like to thank our families for providing us the best condition to study in Advanced Education Program and finish this graduation thesis.

Fall 2013

Trần Quang Hòa - Võ Hồ Tiến Hưng

ASSESSMENT (ADVISOR)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ASSESSMENT (COMMITTEE MEMBERS)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Social Network, the most popular Internet service, had miracle rapid increment of number of users in recent years. Therefore, number of potential users exploited by using their mutual interested topics in social network is extremely large. In this thesis, we propose and proceed to an artificial neural network, Kohonen network, to cluster the actors in social network based on topics of messages. This model can indicates that actors in group or community share topics of mutual interest and topics of communication. Moreover, we use Enron email corpus as a sample dataset to evaluate efficiency in Kohonen network. By experimenting on the dataset, we demonstrate that the model is able to extract well and meaningful cluster following the topics. We use F – measure method for this application for testing precision of SOM algorithm. As a result, from our sample tests, the F-measure cites the acceptable accuracy of the SOM method. For example, with Enron Tiny, F – measure is 13.6333 versus 15 clusters by man-made clustering method. Hence, SOM algorithm is suitable for finding a cluster of actors in social network. Based on the classified the clusters of users, application developer can use the result to build a web environment for areas such as social marketing, expressing information and so on.

Chapter 1: Introduction

1.1 Overview

Nowadays, the world is interconnected through many connections. These connections include WEB, blog, email and social network etc. Therefore, the needs of using social network are to change the way people collect data due to the increase of the amount of data from social network. While the social network data becomes larger and more complex, the analysis of these data is also becoming increasingly important, especially, exploitation of data in social network is the most interested topics.

Extracting from social network, communities are groups of individuals connected to each other in some way [1]. A fundamental problem in social network is to detect groups of people or communities. If social network is performed in graph, with nodes are objects and links are connection between each node, a community is identified by high density of links between nodes. An application about using information from social network is to design marketing model or detect terrorist groups. Hence, user data clustering is necessary for exploiting trend in social network.

Data clustering is the process of searching and discovering groups of similar data in large database. Most of the techniques have been used in solving data clustering problems in areas such as finance, geographic information, biology, image recognition, etc. Recently, clustering methods are proposed and used in social network: k-means, hierarchical agglomerative clustering (HAC), modularity optimization, probabilistic method based on latent models, etc. These methods have been successfully applied in the detection of the communities.

SOM method is used in analyzing and clustering user data in social network. SOM commonly also known as Kohonen network (Kohonen 1982, Kohonen 2001) is a computational method for the visualization and analysis of high-dimensional data, especially experimentally acquired information [2]. SOM will change complex relationship between multi-dimensional data object (n-dimensions) to simple relationship (one or two dimensions).

1.2 Problem statement

Nowadays, the rapid development of social networks has created a huge wave of information exchange in the virtual world. We are using social networks in many fields such as economy, education, politic, etc. Exploiting efficiently social networks in these fields requires us to analyze the exchanged content. Besides, the number of people exchanging information in social network is also growing up quickly and changing every day. Hence, one of the most important problems is to cluster people into different topics. In other words, people clustering mean that to divide people into groups of same topic. These topics have been already available in the system; users can specify the number of topics to classify. The system will indicate which group of people is interested in which topic.

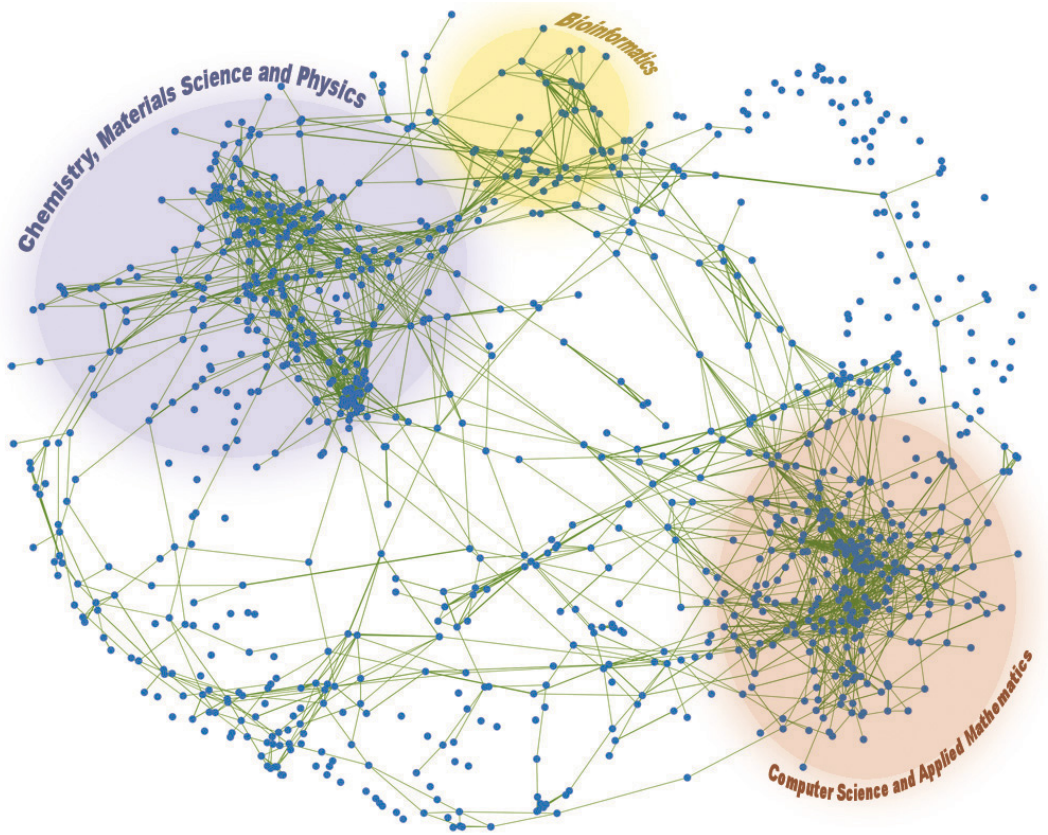


Figure 1.1. Clustering users in social network

1.3 Solution methodology

Cluster of actors in social network is one of study fields which have many applications in our life such as marketing online, expressing information, investigation, etc. In this case, SOM (Self Organizing Map) is known as an animated visualized tool to finding cluster in social network:

- Finding out the topics of actors based on the message of actors on social network. Each actor will correspond with a vector that has K dimensions; K is a number of topics, each part of vector reflect actor attention on topic.
- Applying Kohonen network (SOM) application to group actors into topics. Users can lookup actor in topics.

1.4 Structure of thesis

The thesis is organized as follows:

Chapter 1: Introduction

In this chapter, we give a short brief about our problem and our methodology to solve the problem.

Chapter 2: Fundamental theory

In chapter 2, we introduce about social network definition and Kohonen network model as well as how to identify interested topic of each actor in social network by using ART, AT model and Kohonen Network Model.

Chapter 3: Clustering Actors in Social Network by Kohonen Network

In this chapter, we have two main parts: Fundamental of clustering actors in social network and clustering actors by Kohonen network.

Chapter 4: System Implementation and Testing

In this chapter, we show our implementation and discuss experimental results of our problem.

Chapter 5: Conclusions and Future works

We summarize what we did and propose research directions for the future.

Chapter 2: Fundamental Theories

2.1 Social network

According to Wikipedia, a social network is a social structure made up for a set of actors (such as individuals or organizations) and the dyadic ties between these actors. The social network perspective provides a clear way of analyzing the structure of whole social entities [3]. Moreover, according to webopedia, a social network represents relationships and flows between people, groups, organizations, animals, computers or other information/knowledge processing entities [4].

There are a lot of social networks that people can join for free since the first time it appeared. Some of them are very popular, such as Myspace, Foursquare in North America, Tumblr in North Europe, and Facebook, Twitter, Google+ in many places in the

Figure 2.1. Some most common Social Network

world.

The convenience and comfort of social network makes it be rated as one of the best development in the first decade of 21st century. Days by days, there are more and more people join in Social Network; Table 2.1 includes the 15 Most Popular Social Networking Sites as derived from our *eBizMBA Rank* which is a constantly updated average of each website's *Alexa* Global Traffic Rank, and U.S. Traffic Rank [5]:

Table 2.1. 15 Most Popular Social Networking Sites (eBizMBA Rank)

NO.	SOCIAL NETWORKING SITES	ESTIMATED UNIQUE MONTHLY VISITORS
1	Facebook	750,000,000
2	Twitter	250,000,000
3	LinkedIn	110,000,000
4	Pinterest	85,500,000
5	MySpace	70,500,000
6	Google +	65,000,000
7	DeviantArt	25,500,000
8	LiveJournal	20,500,000
9	Tagged	19,500,000
10	Orkut	17,500,000
11	CafeMom	12,500,000
12	Ning	12,000,000
13	Meetup	7,500,000
14	myLife	5,400,000
15	Ask.fm	4,300,000

Generally, social network is an environment that provides services for connecting members with the same interests, point of view or other purposes, regardless of age, gender, space and time. Besides of connection, information exchange, communication, entertainment, etc. in the virtual world, social network is also an environment for people who work in online business, advertisement or politics, criminal investigation, etc.

Social network is mostly presented by a graph G = (V, E), where:

- V: nodes in the graph, which describe object in network.

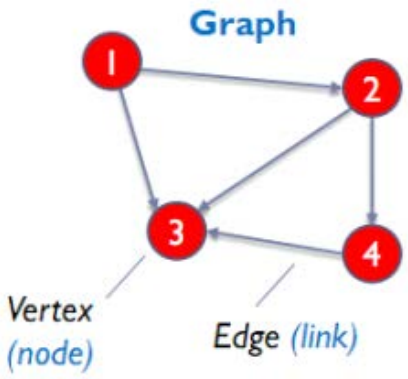


Figure 2.2. Graph sample

- E: edges in the graph, which ties nodes, define node's relation.

In the most basic form, social network is represented as an indirect graph.

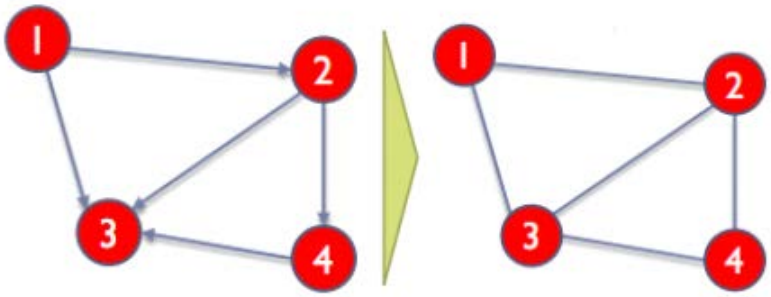


Figure 2.3. Change from direct to indirect graph

2.2 Identifying Interested Topics of Actor in Social network by ART Model

Document creating is a process of identifying a topic, building a paragraph, using grammars and words to make ideas more noticeable. Moreover, topic is a set of concepts (can be seen as process, object, event, property, etc.) those are identified and dominated process of documents.

For example: Topic 1 is “university education”, author build the document by using relevant words such as:

- Bachelor’s degree
- Scientific research
- Graduation thesis
- Etc.

Before introducing about Author Recipient Topic model (ART), we describe Author Topic model. AT model is a generative model that simultaneously models the document content and the interest of author. The selection of distinct topic distribution is identified by taking author from the list of document’s author. Each author has a multinomial distribution over topic and each topic has distribution over word. By calculating the distribution, we can easily to determine which topics are related to which author.

The ART model is a Bayesian network that contemporaneously simulates message content of author and recipient in the directed social network [6]. This model are based on the Latent Dirichlet Allocation (LDA) and the Author – Topic model and added the important attribute that discover topics based on the relationship between the senders and recipients. ART model describes the interaction of each node by analyzing transferring information of each node in the network; a topic relates to author, recipient and discovers role of author and recipient in transferring information process. Hence, the identification of topics in ART model depends on the social network in which messages are sent and received. Each pair of sender and receiver has a distribution over topic and each topic has a distribution over word. That is so similar with the AT model if we ignore a recipient who does not reply the messages. Thus, the output of the ART model is a pair of distribution of sender and recipient over the topics. Finally, we can easily calculate this output in order to figure out the interested topic that they most likely talk about.

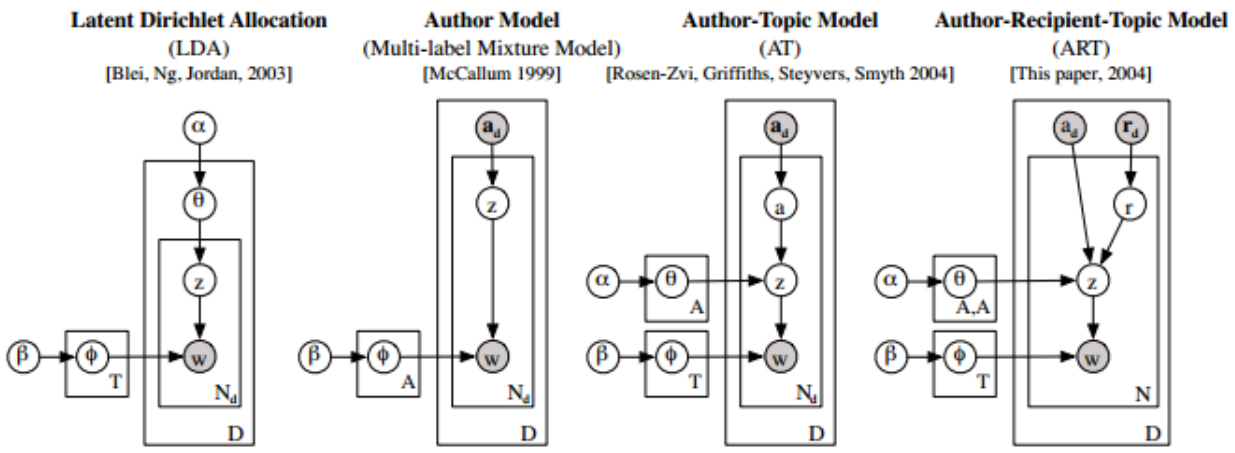


Figure 2.4. The development of social network analyzing models

Where: d: document, a_d: set of author, $r_{d:}$ set of recipient, a: an author, z: a topic

With each message from set of email messages D, set of author a_d and set of recipient r_d are observed. To generate each word, a recipient r are chosen from a set of recipient r_d and then topic z is chosen from the topic distributed function $\theta_{ad,x.}$ Finally, a word w is generated from the topic specific distributed function $ø_z$.

From the ART model, based on the Enron email dataset which composed message of actors on social network; the topic distribution of actors is determined. Each topic distribution of actor will correspond to a vector that has K dimensions (Section 3.1); K is a weight number of topic K^{th} such as sport, education, politic, etc., each part of vector reflects actor attention on topic. Finally, we use effectively the topics distribution to measure and cluster the actors.

2.3 Kohonen Network Model

In different type of neuron networks, Kohonen network is similar to biological neuron network. This artificial neuron network was first introduced in 1982 by Tuevo Kohonen, a professor emeritus of the Academy of Finland [7]. It is known as self-organizing map (SOM), one of the relatively simple models of neuron networks.

Kohonen network (or self-organizing map) models the behavior of the human brain, although it is quite simple. Information processing of other neuron networks is only interested in value of input data, not interested in exploring structure of relationships surrounding the sample data or whole data.

Self-organizing is one of the attractive topics in neural network which can be trained to find out the rules, relationships and predicts the next results. Through the competitive training process, neurons identify the equivalent input data. The main purpose of training Kohonen network is to identify a group of input vectors which have same type.

The implementation of Kohonen network can be replaced by a corresponding algorithm which has always been used in application of Kohonen network. We can call this algorithm: self-organizing neural network (Kohonen, 1988) or self-organizing map (SOM). We use “Self-Organizing” because no supervision is required. SOM learns on its own through unsupervised competitive learning. SOM automatically arranges high dimensional statistical data and map inputs close to each other [7]. In Kohonen network, input signal vectors will be mapped into neighboring neuron in the network.

Kohonen network only includes an input vector and an output layer of neurons. Data input vectors for the Kohonen network are training patterns with size = n. The output layer includes nodes (neurons) which are arranged in grid (map). Each neuron contains weight vector with dimensions equal to dimensions of input vector.

SOM technique has been successfully applied in some fields such as recognition, data clustering, prediction and data mining. Input data can be identified as images, audios or text, etc.

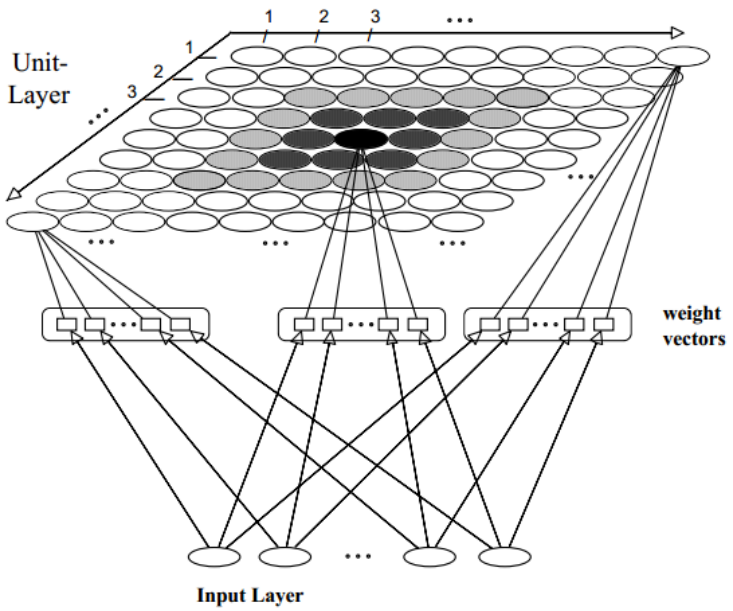


Figure 2.5. SOM structure

Training algorithm:

Initialize time parameter t: t = 0

Step 1: Initialization. Choose random value for weight vector w_{ij} for each "neuron i" in Kohonen network.

Step 2: Sampling. Draw a sample training input vector x from input space.

Step 3: Matching. Find the best match unit (winning neuron) that has weight vector closet to the input vector, i.e.

- Traverse each node in the map.
- Use the Euclidean distance formula to find the similarity between the input vector and the map's node's weight vector.

  Euclidean distance formula:

$$D = \sqrt{\sum_{i=0}^{n}(x_i - W_{ij})^2}$$

  $x_{i:}$ input vector.

W_{ij}: weight vector.

- Track the node that produces the smallest distance (this node is the best matching unit, BMU)

Step 4: Updating. Apply the weight updating equation

$$W_{ij}\,(t+1) = W_{ij}(t) + g(i, j, i_c, j_c, t)(v.\, W_{ij}(t))$$

Step 5: Continuation. Keep returning to step 2 until the feature map stops changing.

Chapter 3: Clustering Actors in Social network by Kohonen Network

In this chapter, we focus on the benefit of clustering actors in social network and how to use Kohonen network as a clustering method. Hence, this chapter contains two parts:

- Fundamental of Clustering Actors in Social network
- Clustering by Kohonen network

3.1 Fundamental of Clustering Actors in Social Network

In this section, we discuss the ways to cluster actors in social network based on their interested topics. At first, we need to investigate the clustering problem in clustering actors. In details, the input data is a vast number of actors which are senders or receivers and their messages and the outcome we get is a set of clusters, each cluster is a group of actors have relevant interested topics. However, the main problem is the input data. For instance, there are some greeting messages such as “hello”, “hi” or “good morning”, these messages is difficult to distinguish they are the same topic or different topics. Since clustering messages and actors in direct is not a good solution due to the complex of data, a process to normalize the input data is required. In this case, this is the encoding process described in this figure:

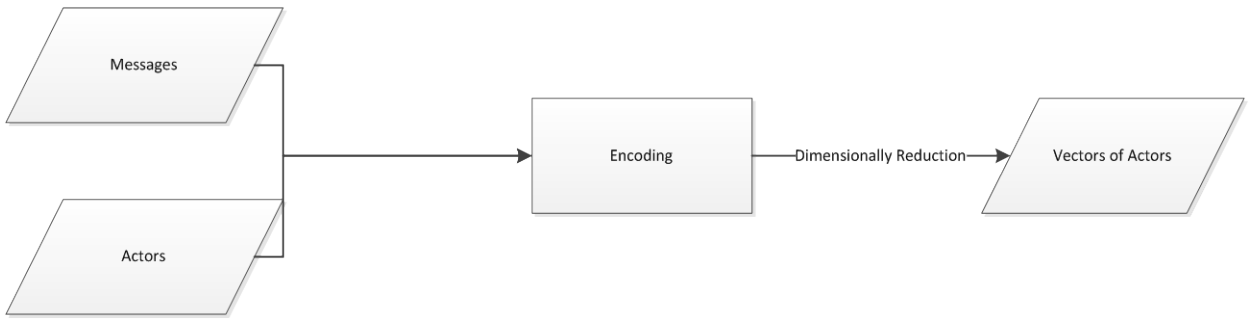


Figure 3.1. The Encoding Process

- In left hand side of the figure, the input of this process is messages and its actors (senders and recipients) extracted from social network.

- The encoding process uses this data as input in order to manipulate the output - a set of actor-vectors based on interested topics in right hand side of the figure.
- Dimensionally reduction is an additional process to reduce dimensional space of the output vectors such as removing redundant topics.
- Each actor-vector based on interested topics is a vector representing the distribution of the topics of each actor in a social network. Hence, the dimensionality of the vector is the number of interested topics. (Table 3.1)

Table 3.1. Example of actor-vectors

Actor	Topic 0	Topic 1	Topic 2
bill.rapp@enron.com	0.5006	0.4475	0.3872
danny.mccarty@enron.com	0.1298	0.5921	0.0327
drew.fossum@enron.com	0.4593	0.2391	0.0098
joe.stepenovitch@enron.com	0.7894	0.0852	0.2983
kevin.hyatt@enron.com	0.3348	0.6683	0.4439

According to the table, we have five vectors:

- *bill.rapp@enron.com:* $\overrightarrow{v_1} = (0.5006, 0.4475, 0.3872)$
- *danny.mccarty@enron.com:* $\overrightarrow{v_2} = (0.1298, 0.5921, 0.0327)$
- *drew.fossum@enron.com:* $\overrightarrow{v_3} = (0.4593, 0.2391, 0.0098)$
- *joe.stepenovitch@enron.com:* $\overrightarrow{v_4} = (0.7894, 0.0852, 0.2983)$
- *kevin.hyatt@enron.com:* $\overrightarrow{v_5} = (0.3348, 0.6683, 0.4439)$

We denote these vectors by $\overrightarrow{v_1}, \overrightarrow{v_2}, \overrightarrow{v_3}$ *… for making it is simple to recall in the next section.*

What is the algorithm for encoding process? We go back the ART (Author- Recipient-Topic) model and AT (Author- Topic) model which have been discussed in chapter 2 to find the answers. As we know, these models are not only used for discovering relevant topics but also predicting the role of actors [8]. Hence, with the input of messages and

actors, we can use them as an algorithm to normalize this plain data to actor-vectors like the previous example. In addition, messages and actors must be extracted as a matrix of frequency of words for all messages and a matrix of several sender-recipient pairs of the whole corpus of messages must be an input data of ART Model.

After having a set of actor-vectors which are extracted from messages and actors in social network, the next process is clustering these vectors which is called as mapping process and described in this figure:

Figure 3.2. The Mapping Process

- In the left hand side, the input data is the output of the previous encoding process: a set of actor-vectors based on interested topics.
- The Mapping Function is the method to classify these vectors as several clusters. In details, this function identifies the similarity of each vectors based on its dimensional value.
- Map of Cluster of Actors is the output of this process. This is a matrix map which represents the result of the mapping function. In this map, each cluster of vectors is its element which contains the relevant actors and their interested topics.

Suppose that the set of vectors in Table 3.1 is the input vectors of this process so that the mapping function can detect the similarity and then grouping them into several clusters. For example, $\overrightarrow{v_1}$ and $\overrightarrow{v_3}$ are similar so that $\overrightarrow{v_1}$ and $\overrightarrow{v_3}$ belong to a cluster. As a result, $\overrightarrow{v_1}$ and $\overrightarrow{v_3}$ also belong to an element in the output map.

There are several algorithms for mapping function. These algorithms are Bayesian Methods, Kohonen network Model discussed in Chapter 2 and so forth. In this thesis, we choose Kohonen network Model as mapping function for several benefits which are:

- Easy to implement due to it based on neural network theory. It means that the implementation of Kohonen network is general so that it is not necessary to modify for different applications or input data, instead, only training set for training process need to be changed.
- Reusable. After the training process, we store this outcome and reuse for the upcoming clustering which has the similar input data to the current input data. For instance, if we need to cluster a set of vectors which is similar to example in Table 3.1, we can use the result of training process of vectors in Table 3.1.
- Simple to improve the performance because of its ability to reuse.

In summary, to cluster actors in social network, two processes are required. The first process is encoding process based on Author- Recipient- Topic Model (ART Model) or Author- Topic Model in order to normalize the plain input data which are a set of actors and messages to a set of actor-vectors based on the interested topics. The second process is mapping process based on Kohonen network to classify these actor-vectors as clusters and represent them as a self- organizing map (SOM). In the SOM, each element is a set of actors which has similarity in interested topics.

So the new trouble is how to use Kohonen network to cluster the actor-vectors? This is the main topic of the next section we are going to discuss.

3.2 Clustering Actors by Kohonen Network

3.2.1 Initializing Procedure

As we know in Chapter 2, Kohonen network is a special kind of neural network. The special of this network is that there are only input and output layers in this network without any hidden layers like other kind of neural network. However, its initialization is approximately the same as the other type of neural network which are:

- Initializing the input and output layers
- Initializing the weights
- Initializing parameters, thresholds and function for training process

Let us find out what is the similarity and also the distinction of each initialization. At first, the overview of the Kohonen network structure for clustering vector described in this figure:

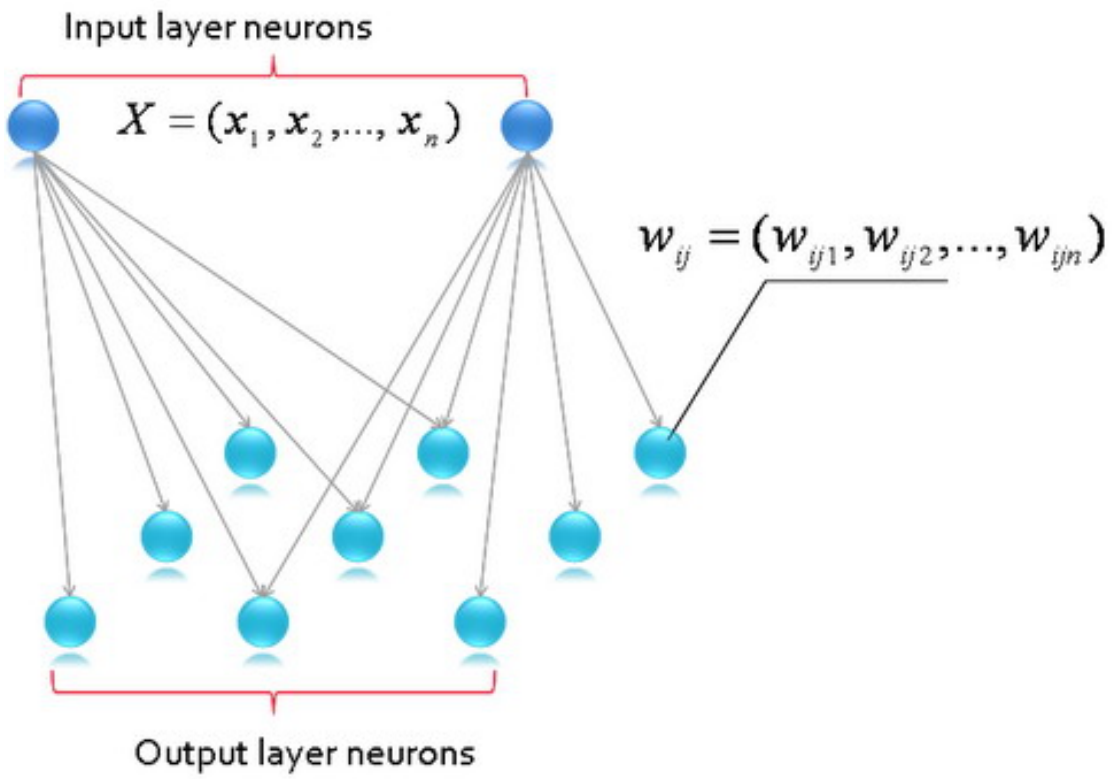


Figure 3.3. The Kohonen neuron network structure for clustering vectors [9]

In initializing the input and output layers, according to the figure 3.3, the input layer is a unique vector X .Each dimensional value of X such as x_1, x_2 or x_n is represented as a certain input layer neuron in the figure 3.5. On the other hand, the output layers of Kohonen network is a three-dimensional matrix of neurons. The self-organizing map is described as a square matrix since each output layer neuron is a group of one-dimensional matrix or a vector of weights with the number of its element is the number of input layer neurons or the number of dimensions of input vector - n. Therefore, the data we need for initializing the input and output layers will be:

- Let n be the number of dimensions of the input vector or the number of interested topics.

– And m be the number of elements for the output layer or the self-organizing map.

Assume that we use vectors in table 3.1 as input, in this case, n is equal to 3 because these vectors have 3 dimensions or interested topics and m is depend on how many output neurons we want. As a result, the output neurons is a self-organizing map (SOM) with m element and each element has 3 weights or we have m vectors in the out neurons layer.

In initializing the weights, after prerequisite process - the initializing input layers is completed. This initializing process is required to create weights of the SOM. Due to the characteristics of neural network such as self-learning of Kohonen network, the initializing weights is not essential to consider since they will be updated to an appropriate value during the training procedure. Of course, the initializing weights affect the learning process time, if the initializing weights is closed to the training set patterns, the number and also the time needed for learning steps are reduced as well.

However, in this case of clustering actor-vectors, it is difficult and ambitious to find an algorithm for finding values based on the ambiguous input vectors. This is a reason why in this process, we assign the weights of the output randomly in range $[0, 1)$ like several other kinds of neural networks.

Back to the previous example, suppose we choose m is 4 so that we have 2 x 2 SOM and the value of each element assigned by a random value:

$$\begin{bmatrix} \vec{w}_{0,0} & \vec{w}_{0,1} \\ \vec{w}_{1,0} & \vec{w}_{1,1} \end{bmatrix}$$

$\vec{w}_{0,0}$= random(0,1)=$(0.3241, 0.9804, 0.2319)$

$\vec{w}_{0,1}$= random(0,1)=$(0.1673, 0.2922, 0.1137)$

$\vec{w}_{1,0}$= random(0,1)=$(0.5985, 0.0101, 0.8763)$

$\vec{w}_{1,1}$= random(0,1)=$(0.4312, 0.0032, 0.0021)$

This outcome is what we desire to get from this procedure.

In initializing the parameters, thresholds and function for training process, these values must be identified:

- Number of iterations
- Epsilon (ε)
- Learning rate (α)
- Threshold of interested topic

Number of iterations is a maximum number of learning steps in training procedure. It means that the learning process still repeat until the number of loops is equal the number of iterations we desire. For example, the number of iterations is 100 then the learning process repeats 100 times.

Epsilon is a value which defines the lower bound or the minimum value of the modification value for updating neuron's value after a learning epoch. In additional, if the modification is not extent due to the modification value is less than the epsilon, the learning procedure should end. For instance, let epsilon be equal to 0.01, after several learning iterations, the modification value to update neuron's value is 0.008 which is less than 0.01 then we stop the learning because the learning phase becomes no significance or the out neuron has been approached to the desire value.

Learning rate is a value which indicates the acceleration of learning procedure. It let the modification value be high or low. During the procedure, the learning rate is reduced by a learning rate function. This function is used once of each learning epoch to update the learning rate value. For example, at the first epoch, learning rate is 0.5 then in the next epoch it must be less than 0.5 such as 0.2. As a result, the modification value at the first epoch is higher than the next epochs.

Threshold of interested topics is a threshold we use to identify the interested topics of an output element in SOM. Each of weights of an output neuron is compared to the threshold to ensure the interested topics of this neuron if the weight is equal or greater than the threshold. For example, we have the trained output neuron is $\vec{w}_{0,0} =$

$(0.1572, 0.9281, 0.5332)$ and the threshold is equal to 0.5, thus, the interested topics of this neuron is topic 1 (0.9281) and topic 2 (0.5332).

In order to make more clear, these figures below are the pseudocodes of the entire procedure.

```
Void function SOM_Initializing(Number of iterations, ε, α, threshold of interested
topics, Number of interested topics, Number of output neuron)
{
        //integer value
        this.Number of iterations ← Number of iterations;
        this.n ← Number of considerable topics;
        this.m ← Number of output neuron;
        //double value
        this.ε ← ε;
```

Figure 3.4. Pseudocode of Initializing Procedure

After the initializing procedure, learning procedure is the next process to self-train and update the weights of SOM.

```
        this.α ← α;
        this.threshold of interested topics ← threshold of considerable topics;
        //initialize the SOM
        SOM ← new double[√m][√m];
        //initialize weights of each neuron by random function
        for (int i = 0; i < √m; i++)
            for (int j = 0; j < √m; j++)
                for (int k = 0; k < n; k++)
                    SOM[i][j].weights[k] = random(0,1);
}
```

Figure 3.5. Pseudocode of Initializing Procedure (cont.)

3.2.2 Learning Procedure

This section indicates briefly about the learning procedure of SOM for clustering actor-vectors, especially the learning set and the learning rule. Let find out what is the learning set first.

According to the previous chapters and also our knowledge of neural network and Kohonen network, the learning set is an important ingredient for the learning procedure since neural network learns by using this learning set to re-update its weights. So what is the appropriate learning set for this situation which is clustering a set of actor-vectors? Is there any other set of vectors for using as learning set in this case?

The solution of this situation is to use the input actor-vectors as the learning set for SOM. The purpose is we desire to cluster and classify these vectors so there are no learning set more suitable than them. For example, the input actor-vectors is a set of vectors in Table 3.1 then the learning set is also this set of $\overrightarrow{v_1}, \overrightarrow{v_2}, \overrightarrow{v_3}\ldots$

The reason for this outcome is in the learning process from each vector of learning set, we need to find the winning output neuron then we updates the value for relevant neurons which depends on the winning output neuron and the current input vector. Finally, the trained SOM has an ability to cluster these vectors which is the prerequisite for the clustering procedure.

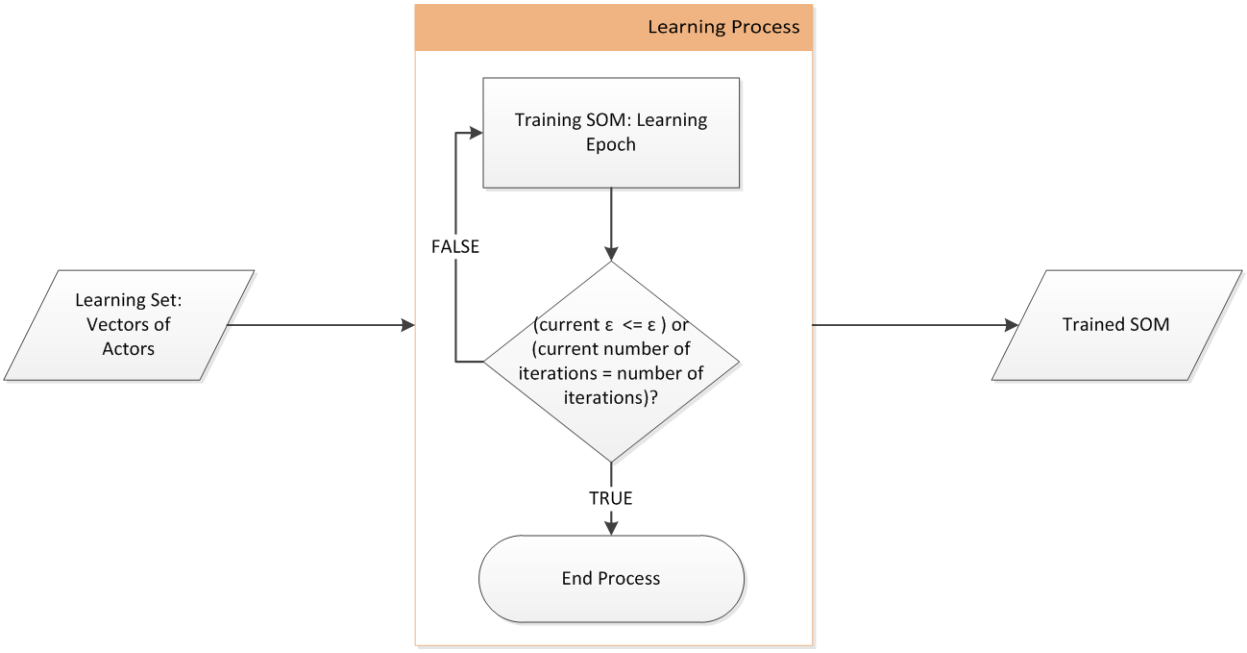


Figure 3.6. Basic of Learning Procedure

The input learning set of actor-vectors is a prerequisite for learning process to get the output of a trained SOM. From the figure, the learning process can be described that it uses learning set to train itself called as learning epoch then the new epoch start unless the outcome SOM matches the threshold (ε and number of iterations).

What is inside a learning epoch? As we know, this iteration is simple. First of all, we pick up a vector from the learning set. Then, we just apply the learning rule to renew the SOM such as these tasks we know which are to find the winning neuron and its neighborhood, update weights and change parameters. Finally, we finish the epoch.

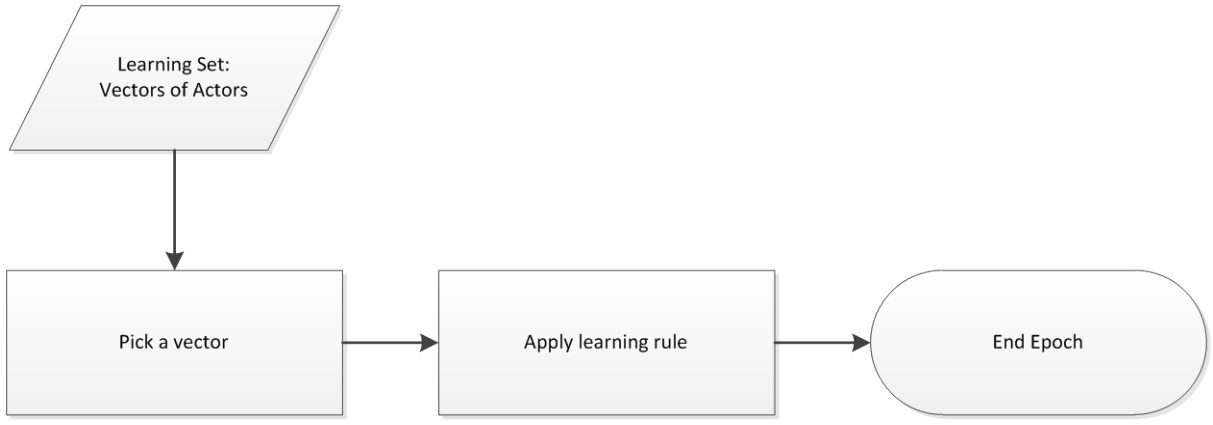


Figure 3.7. The Learning Epoch

On the other hand, what is learning rule? It is a rule for identifying the winning neuron and its neighborhood, updating weights, parameters such as learning rate, neighborhood function. Let find out the learning rule in details to comprehend it exactly.

Firstly, how can we identify the winning neuron? To know the answer let us start with the familiar term called the Euclidean distance. Imaging that we have the n-dimensional space with m is equal to the number of interested topics or the number of dimensions of vector of a training set and the neural vector of SOM, the Euclidean distance between the training vector and the neural vector can be computed by the formula as follows:

$$D_{i,j} = \left|\vec{v}_x - \vec{w}_{i,j}\right| = \sqrt{\sum_{k=0}^{n-1} \left(v_{x_k} - w_{(i,j)_k}\right)^2}$$

(Formula 3.1. Euclidean distance for learning procedure)

With:

- $D_{i,j}$ is the distance between the current training vector and the current neural vector at row i, column j in the SOM
- $\vec{v}_x$ is the current training vector
- $\vec{w}_{i,j}$ is the current neural vector at row i, column j in SOM
- k is the index of dimension of interested topics

- v_{x_k} is the dimensional value at dimension k
- $w_{(i,j)_k}$ is the weight of $\vec{w}_{i,j}$ at dimension k

For instance, suppose that the current learning vector is $\overrightarrow{v_1} = (0.5006, 0.4475, 0.3872)$ and the example SOM is the 2 x 2 SOM which we have discussed previously then:

$$D_{0,0} = \left|\vec{v}_1 - \vec{w}_{0,0}\right| = \sqrt{\sum_{k=0}^{2}\left(v_{1_k} - w_{(0,0)_k}\right)^2}$$

$$= \sqrt{(0.5006 - 0.3241)^2 + (0.4475 - 0.9804)^2 + (0.3872 - 0.2319)^2}$$

$$\approx 0.5824$$

$$D_{0,1} = \left|\vec{v}_1 - \vec{w}_{0,1}\right| = \sqrt{\sum_{k=0}^{2}\left(v_{1_k} - w_{(0,1)_k}\right)^2}$$

$$= \sqrt{(0.5006 - 0.1673)^2 + (0.4475 - 0.2922)^2 + (0.3872 - 0.1137)^2}$$

$$\approx 0.4582$$

$$D_{1,0} = \left|\vec{v}_1 - \vec{w}_{1,0}\right| = \sqrt{\sum_{k=0}^{2}\left(v_{1_k} - w_{(1,0)_k}\right)^2}$$

$$= \sqrt{(0.5006 - 0.5985)^2 + (0.4475 - 0.0101)^2 + (0.3872 - 0.8763)^2}$$

$$\approx 0.6634$$

$$D_{1,1} = \left|\vec{v}_1 - \vec{w}_{1,1}\right| = \sqrt{\sum_{k=0}^{2}\left(v_{1_k} - w_{(1,1)_k}\right)^2}$$

$$= \sqrt{(0.5006 - 0.4312)^2 + (0.4475 - 0.0032)^2 + (0.3872 - 0.0021)^2}$$

$$\approx 0.5920$$

Or we have:

Table 3.2. Example of Calculating Euclidean Distance

	Column 0	Column 1
Row 0	0.5834	0.4582
Row 1	0.6634	0.5920

Then after having all of these values, the winning neuron is identified by finding the neuron which has the shortest distance in the result set. In this case, the winning neuron is $w_{0,1} = w_{x,y}$. (Since $D_{0,1} = D_{min}$)

After that, the next step is to determine the neighborhood of the winning neuron or the winning region. As we know, this region is determined by a neighborhood function which is a kind of topological functions. According to the previous sections, we know that the function occasionally has two arguments which are the index of winning neuron and the current iterations. Then, the function returns a value for updating weights function. If this neuron is not in winning region it return 0, else, there is a particular value for the neuron to update weights. Because there are several topological functions can be used for this purpose, let get an overview of some of them.

Assume that:

- $w_{x,y}$: The current winning neuron
- $w_{i,j}$: The current neuron
- $w_{x,y}.row$: The current winning neuron's row
- $w_{x,y}.column$: The current winning neuron's column
- $w_{i,j}.row$: The current neuron's row
- $w_{i,j}.column$: The current neuron's column
- r : The distance between w_x and w_i
- t : The current iteration
- h : The neighborhood function

- $\sigma(t)$: The function used for identifying the space of the neighborhood. In the beginning of the function, it involves almost the whole space of the grid, but with time, the value of σ decreases [9]:
    - $\sigma(t) = \sigma_0 e^{-\frac{t}{\tau_1}}$ ($\tau_1$is a constant, $\sigma_0 = \sqrt{m}$, output neural layer dimension)

The first example of topological function is Mexican Hat:

- With:

$$r = \sqrt{\left(w_{x,y}.row - w_{i,j}.row\right)^2 + \left(w_{x,y}.column - w_{i,j}.column\right)^2}$$

- The neighborhood function is represented as:

$$h(r,t) = \left(1 - \frac{2}{\sigma^2(t)} r^2\right) e^{-\frac{r^2}{\sigma^2(t)}}$$

The second example is Gauss function as follows:

- With:

$$r = \sqrt{\left(w_{x,y}.row - w_{i,j}.row\right)^2 + \left(w_{x,y}.column - w_{i,j}.column\right)^2}$$

- The neighborhood function is represented as:

$$h(r,t) = e^{-\frac{r^2}{\sigma^2(t)}}$$

The third example is French Hat:

- With:

$$r = \left|w_{x,y}.row - w_{i,j}.row\right| + \left|w_{x,y}.column - w_{i,j}.column\right|$$

- The neighborhood function is represented as:

$$h(r) = \begin{cases} 1, & r \leq a \\ -\frac{1}{3}, & a < r \leq 3a \\ 0, & r < 3a \end{cases}$$

The last example is Discrete:

- With:

$$r = \left|w_{x,y}.row - w_{i,j}.row\right| + \left|w_{x,y}.column - w_{i,j}.column\right|$$

- The neighborhood function is represented as:

$$h(r) = \begin{cases} 1, & r = 0 \\ \frac{1}{2}, & r = 1 \\ \frac{1}{4}, & r = 2 \\ \frac{1}{8}, & r = 3 \end{cases}$$

Therefore, the testing method is required to decide which one is the most appropriate function for our learning process (See chapter 4).

In this chapter, suppose that we choose Gaussian function as a topological neighborhood function.

Next, the learning rate or α must be considered. As we know, like the winning region, the learning rate is not a constant, alternatively, it reduces by iteration and determine as the following function which is approximately the same as σ function:

$$\alpha(t) = \alpha_0 e^{-\frac{t}{\tau_2}}$$

With:

- $\alpha(t)$: The learning rate at the iteration t
- α_0: The initializing value of learning rate
- t: The current number of iterations

- τ_2: Constant

The next step after finding the winning region is to update weights of the winning neurons. In this step, weights of all of elements in the whole SOM are updated since the update only affects in winning region caused by the return value of neighborhood function. To be more understandable and comprehensible, the formula for updating weight showed as follows:

$$w'_{(i,j)_k} = w_{(i,j)_k} + \alpha(t)h(r,t)\left(v_{x_k} - w_{(i,j)_k}\right) \qquad \forall k \in \mathbb{N}, 0 \leq k \leq n$$

With:

- k: The dimension of neuron weights
- n: The number of interested topics
- $w'_{(i,j)_k}$: The new (post-update) value of k[th] weight of the neuron at row i, column j
- $w_{(i,j)_k}$: The current (pre-update) value of k[th] weight of the neuron at row i, column j
- $\alpha(t)$: The learning rate at the current number of iterations
- $h(r,t)$: The result of topological neighborhood function with t is the current number of iterations, r is the distance between the current neuron and the winning neuron
- v_{x_k}: The value of k[th] weight of the current learning vector v_x

Finally, the current ε and the current number of iterations are updated as follows in order to compare with their inputs to make sure the SOM is trained:

- For the current number of iterations, it increased by 1.
- For the current ε or ε', it is equal to the overall of the gained value of all weights in the traversing neuron $w_{i,j}$ as the formula follows:

$$\varepsilon' = \frac{\sum_{i=0}^{\sqrt{m}-1}\sum_{j=0}^{\sqrt{m}-1}\frac{\sum_{k=0}^{n-1}\left(w'_{(i,j)_k} - w_{(i,j)_k}\right)}{n}}{m}$$

$$= \frac{\sum_{i=0}^{\sqrt{m}-1}\sum_{j=0}^{\sqrt{m}-1}\frac{\sum_{k=0}^{n-1}\alpha(t)h(r,t)\left(v_{x_k} - w_{(i,j)_k}\right)}{n}}{m}$$

Let us go back with the example, after finding the winning neuron is neuron (0, 1), suppose that:

- $\sigma_0 = \sqrt{m} = 2$
- $\tau_1 = \frac{1000}{\ln \sigma_0}$
- $\alpha_0 = 0.1$
- $\tau_2 = 1000$

Since $t = 0, x = 1$: (the current training vector is $\overrightarrow{v_1}$)

$$w'_{(i,j)_k} = w_{(i,j)_k} + \alpha(t)h(r,t)\left(v_{x_k} - w_{(i,j)_k}\right) = w_{(i,j)_k} + \alpha_0 h(r,t)\left(v_{1_k} - w_{(i,j)_k}\right)$$

Updating neuron (0, 0):

$$r = \sqrt{\left(w_{0,1}.row - w_{0,0}.row\right)^2 + \left(w_{0,1}.column - w_{0,0}.column\right)^2} = 1$$

$$h(r,t) = \left(1 - \frac{2}{\sigma^2(t)}r^2\right)e^{-\frac{r^2}{\sigma^2(t)}} = \left(1 - \frac{2}{{\sigma_0}^2}r^2\right)e^{-\frac{r^2}{{\sigma_0}^2}} = \frac{1}{2}e^{-\frac{1}{4}}$$

$$w'_{(0,0)_0} = w_{(0,0)_0} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(0,0)_0}\right)$$

$$= 0.3241 + 0.1 \times \frac{1}{2}e^{-\frac{1}{4}} \times (0.5006 - 0.3241) \approx 0.3241 + 0.0067$$

$$\approx 0.3308$$

$$w'_{(0,0)_1} = w_{(0,0)_1} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(0,0)_1}\right)$$

$$= 0.9804 + 0.1 \times \frac{1}{2}e^{-\frac{1}{4}} \times (0.4475 - 0.9804) \approx 0.9804 + (-0.0207)$$

$$\approx 0.9597$$

$$w'_{(0,0)_2} = w_{(0,0)_2} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(0,0)_2}\right)$$
$$= 0.2319 + 0.1 \times \frac{1}{2} e^{-\frac{1}{4}} \times (0.3872 - 0.2391) \approx 0.2319 + 0.0057$$
$$\approx 0.2376$$

Thus,

$$\vec{w}_{0,0} = (0.3308, 0.9597, 0.2376)$$

Updating neuron (0, 1):

$$r = \sqrt{\left(w_{0,1}.row - w_{0,1}.row\right)^2 + \left(w_{0,1}.column - w_{0,1}.column\right)^2} = 0$$

$$h(r,t) = \left(1 - \frac{2}{{\sigma_0}^2} r^2\right) e^{-\frac{r^2}{{\sigma_0}^2}} = 0$$

$$w'_{(0,1)_0} = w_{(0,1)_0} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(0,1)_0}\right) = w_{(0,1)_0}$$

$$w'_{(0,1)_1} = w_{(0,1)_1} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(0,1)_1}\right) = w_{(0,1)_1}$$

$$w'_{(0,1)_2} = w_{(0,1)_2} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(0,1)_2}\right) = w_{(0,1)_2}$$

Therefore,

$$\vec{w}_{0,1} = (0.1673, 0.2922, 0.1137)$$

Updating neuron (1, 0):

$$r = \sqrt{\left(w_{0,1}.row - w_{1,0}.row\right)^2 + \left(w_{0,1}.column - w_{1,0}.column\right)^2} = \sqrt{2}$$

$$h(r,t) = \left(1 - \frac{2}{{\sigma_0}^2} r^2\right) e^{-\frac{r^2}{{\sigma_0}^2}} = 0$$

$$w'_{(1,0)_0} = w_{(1,0)_0} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(1,0)_0}\right) = w_{(1,0)_0}$$

$$w'_{(1,0)_1} = w_{(1,0)_1} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(1,0)_1}\right) = w_{(1,0)_1}$$

$$w'_{(1,0)_2} = w_{(1,0)_2} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(1,0)_2}\right) = w_{(1,0)_2}$$

Hence,

$$\vec{w}_{1,0} = (0.5985, 0.0101, 0.8763)$$

Updating neuron (1, 1):

$$r = \sqrt{\left(w_{0,1}.row - w_{1,1}.row\right)^2 + \left(w_{0,1}.column - w_{1,1}.column\right)^2} = 1$$

$$h(r,t) = \left(1 - \frac{2}{{\sigma_0}^2} r^2\right) e^{-\frac{r^2}{{\sigma_0}^2}} = \frac{1}{2} e^{-\frac{1}{4}}$$

$$\begin{aligned} w'_{(1,1)_0} &= w_{(1,1)_0} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(1,1)_0}\right) \\ &= 0.4312 + 0.1 \times \frac{1}{2} e^{-\frac{1}{4}} \times (0.5006 - 0.4312) \approx 0.4312 + 0.0027 \\ &\approx 0.4339 \end{aligned}$$

$$\begin{aligned} w'_{(1,1)_1} &= w_{(1,1)_1} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(1,1)_1}\right) \\ &= 0.0032 + 0.1 \times \frac{1}{2} e^{-\frac{1}{4}} \times (0.4475 - 0.0032) \approx 0.0032 + 0.0173 \\ &\approx 0.0205 \end{aligned}$$

$$\begin{aligned} w'_{(1,1)_2} &= w_{(1,1)_2} + \alpha_0 h(r,t)\left(v_{1_0} - w_{(1,1)_2}\right) \\ &= 0.0021 + 0.1 \times \frac{1}{2} e^{-\frac{1}{4}} \times (0.3872 - 0.0021) \approx 0.0021 + 0.0149 \\ &\approx 0.0170 \end{aligned}$$

So that

$$\vec{w}_{1,1} = (0.4339, 0.0205, 0.0170)$$

Then update the current number of iterations and the current ε:

$$number\ of\ iterations = number\ of\ iterations + 1 = 0 + 1 = 1$$

$$\varepsilon' = \frac{\sum_{i=0}^{\sqrt{m}-1}\sum_{j=0}^{\sqrt{m}-1}\frac{\sum_{k=0}^{n-1}\alpha(t)h(r,t)\left(v_{x_k} - w_{(i,j)_k}\right)}{n}}{m}$$

$$= \frac{\left(\frac{0.0067 + (-0.0207) + 0.0057}{3}\right) + \left(\frac{0}{3}\right) + \left(\frac{0}{3}\right) + \left(\frac{0.0027 + 0.0173 + 0.0149}{3}\right)}{4}$$

$$\approx 0.0022$$

The final step is to check the stop condition, assume in this case, threshold of number of iterations is 0, as a result, the learning process ends because the current number of iterations is greater than the threshold. Hence, this is our trained SOM:

$$\begin{bmatrix} \vec{w}_{0,0} & \vec{w}_{0,1} \\ \vec{w}_{1,0} & \vec{w}_{1,1} \end{bmatrix}$$

$$\vec{w}_{0,0} = (0.3308, 0.9597, 0.2376)$$

$$\vec{w}_{0,1} = (0.1673, 0.2922, 0.1137)$$

$$\vec{w}_{1,0} = (0.5985, 0.0101, 0.8763)$$

$$\vec{w}_{1,1} = (0.4339, 0.0205, 0.0170)$$

In summary, the pseudocode for the entire learning procedure is stated as these figures below.

```
Function d (v, w) //Euclidean distance
{
    s = 0;
    for (k = 0; k < n; k++)
        s+= sqr(v.weights[k] − w.weights[k]);
    return sqrt(s);
}
Function find_winningneuron(v, SOM) //find wining neuron, v is training vector
{
    min = d(v, SOM[0,0]);
    wneuron = SOM[0,0];
    for (i = 0; i < sqrt(m); i++)
        for (j = 0; j < sqrt(m); j++)
            if (min > d(v, SOM[i, j]))
            {
                min = d(v, SOM[i, j]);
                wneuron = SOM[i, j];
            }
    return wneuron;
}
Function sigma (t) //sigma function
{
    sigma0 = sqrt(m);
    tau1 = 1000/Math.log(sigma0);
    return (sigma0 ∗ Math.log(−t/tau1));
}
```

Figure 3.8. Pseudocode of Learning Procedure

```
//Guassian topological neighborhood function w is the winning neuron, c is the current neuron
Function h (c, w, t)
{
    r = sqrt(sqr(w.row − c.row) + sqr(w.column − c.column));
    return Math.log(−sqr(r/sigma(t)));
}
Function alpha (t) //alpha function
{
    alpha0 = 0.1;
    tau2 = 1000;
    return (alpha0 ∗ Math.log(−t/tau2));
}
Function modify_weights (c, w, t) //update a specific neuron
{
    s = 0; //use to identify the current epsilon
    for (k = 0; k < n; k + +)
    {
        l = alpha(t) ∗ h(w, c, t) ∗ (c.weights[k] − w.weights[k]);
        c.weights[k]+= l;
        s+= l;
    }
    return (s/n);
```

Figure 3.9. Pseudocode of Learning Procedure (cont.1)

```
Function learning_epoch (v, t, SOM) //a epoch of learning procedure
{
    epsilon = 0;
    w = find_winningneuron(v, SOM);
    for (i = 0; i < sqrt(m); i++)
        for (j = 0; j < sqrt(m); j++)
            epsilon+= modify_weights(SOM[i][j], w, t);
    return epsilon/m;
}
//learning procedure with v[ ] is training set, epsilon0, iterations0 is thresholds for stop condtion
Void function learning_procdure(v[ ], SOM, epsilon0, iterations0)
{
    epsilon = 0.1;
    iterations = 0;
    while (epsilon > epsilon0 && iterations ≤ iterations0)
    {
        v = v[random(0, v[ ].size − 1)];
        epsilon = learning_epoch(v, iterations, SOM);
        iterations++;
    }
}
```

Figure 3.10. Pseudocode of Learning Procedure (cont.2)

3.2.3 Clustering Procedure

After we have a trained SOM, clustering procedure is the next procedure which has the complexity is not as high as the previous procedure. Nevertheless, it is one of the most important procedures since it is a procedure for getting the result or clustering vector-actors. So how it works? Let have a look as the figure follows.

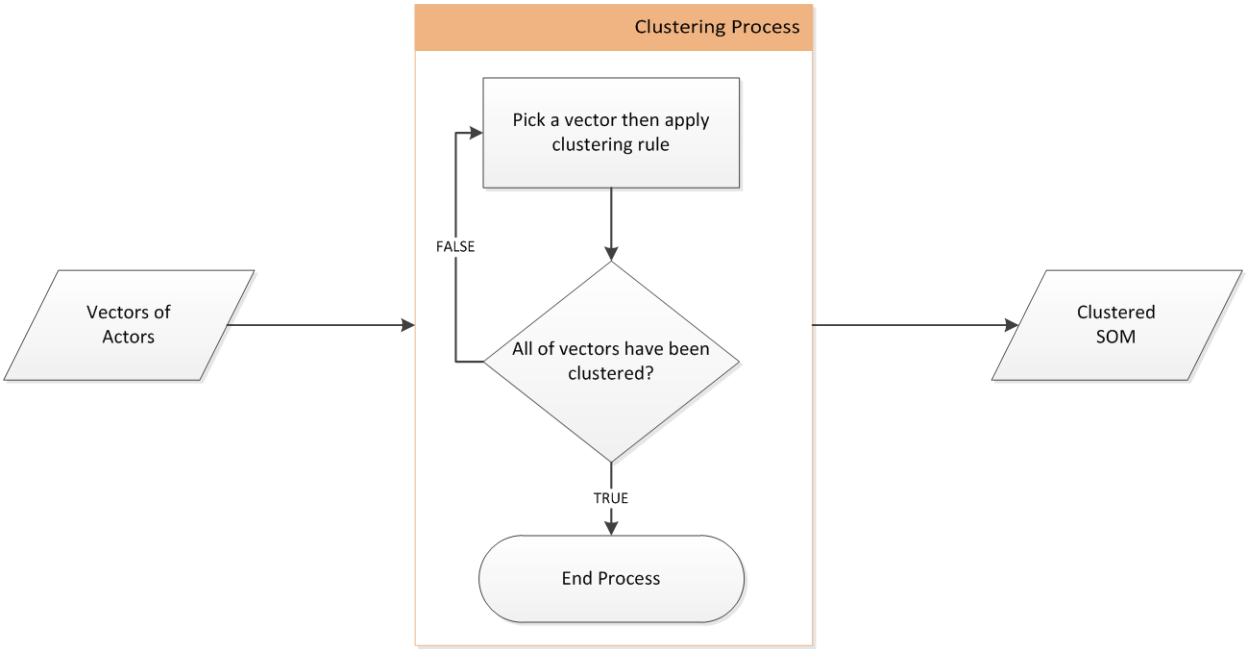


Figure 3.11. Basic of Clustering Procedure

From the figure, the process just a simple loop that applying clustering rule for each vector in a set of actor-vectors we desire to classify. Moreover, this set of vectors is actually the training set used in learning procedure and the outcome is the clustered SOM. Hence, there are two terms we need to consider which are the clustering rule and the clustered SOM. What are they? It is an interesting question.

First of all, we come to the term of clustered SOM. Clustered SOM is SOM which represents map of cluster of actors. It means that the clustered SOM has each its output neuron is a cluster which indicates the number of actors, their description and what are the interested topics of it. For describing the number of actors, we can base on the color levels. The darker the neuron is, the more number of actors it has. On the other hand, for

representing interested topics, it is based on comparing n of weights of the neural element with the threshold.

Another term is the clustering rule. This is a manner to match each vector to a certain neural output element of the SOM. It means that the rule helps us know the most acceptable output neuron in the SOM for a particular vector. Therefore, obviously, the clustering rule is just another type of finding winning neuron process in learning procedure. If there are several vectors which have the similar winning neuron, this neuron is their cluster. In fact, the difference between clustering rule and finding winning neuron that is the SOM has been trained in clustering rule and this rule supports to form the clustered SOM.

Therefore, the clustering rule can be described as these steps:

- Find the winning neuron by comparing the Euclidean distance which is computed by Formula 3.1
- Store the result as a list of actor for each neuron

Let go back with the example, with the trained SOM computed in the end of learning procedure we have:

For $\vec{v}_1$:

$$D_{0,0} = \left|\vec{v}_1 - \vec{w}_{0,0}\right| \approx 0.5599$$

$$D_{0,1} = \left|\vec{v}_1 - \vec{w}_{0,1}\right| \approx 0.4582$$

$$D_{1,0} = \left|\vec{v}_1 - \vec{w}_{1,0}\right| \approx 0.6634$$

$$D_{1,1} = \left|\vec{v}_1 - \vec{w}_{1,1}\right| \approx 0.5690$$

Hence, $\vec{v}_1$ belongs to neuron (0, 1).

For $\vec{v}_2$:

$$D_{0,0} = \left|\vec{v}_2 - \vec{w}_{0,0}\right| \approx 0.4663$$

$$D_{0,1} = \left|\vec{v}_2 - \vec{w}_{0,1}\right| \approx 0.3129$$

$$D_{1,0} = \left|\vec{v}_2 - \vec{w}_{1,0}\right| \approx 1.1269$$

$$D_{1,1} = \left|\vec{v}_2 - \vec{w}_{1,1}\right| \approx 0.6476$$

Hence, $\vec{v}_2$ belongs to neuron (0, 1).

For $\vec{v}_3$:

$$D_{0,0} = \left|\vec{v}_3 - \vec{w}_{0,0}\right| \approx 0.7665$$

$$D_{0,1} = \left|\vec{v}_3 - \vec{w}_{0,1}\right| \approx 0.3144$$

$$D_{1,0} = \left|\vec{v}_3 - \vec{w}_{1,0}\right| \approx 0.9069$$

$$D_{1,1} = \left|\vec{v}_3 - \vec{w}_{1,1}\right| \approx 0.2201$$

Hence, $\vec{v}_3$ belongs to neuron (1, 1).

Using the same method for $\vec{v}_4, \vec{v}_5$, we have $\vec{v}_4$ belongs to neuron (1, 1) and $\vec{v}_5$ belongs to neuron (0, 0).

As a result, we have:

Table 3.3. Example Result of Clustering Procedure

	Column 0	Column 1
Row 0	$\{\vec{v}_5\}$	$\{\vec{v}_1, \vec{v}_2\}$
Row 1	$\{\vec{v}_3, \vec{v}_4\}$	$\emptyset$

The last step is to represent actors, its amount and finding the interested topics of each cluster. For representing actors and its amount, we can see it in the table that neuron (1, 1) does not contain any vector then its color is the brightest in the output neuron, besides, neuron (0, 0) has a vector then its color is darker than neuron (1, 1) and the other neurons have two vectors so as the bold of its color.

For finding the interested topics of each cluster, assume that the threshold for interested topic in this case is equal to 0.25, so the interested topics of each neuron is topics which have weight greater than the threshold. That is a reason why we have the table below after comparing the weights with the threshold:

Table 3.4. Example Result of Finding Interested topics

	Column 0	Column 1
Row 0	{Topic 0, Topic 1}	{Topic 1}
Row 1	{Topic 1, Topic 2}	∅

In conclusion, the pseudocode of the clustering procedure is showed as follows:

```
Void function clustering_procedure
(SOM, v[ ], threshold of considerable topics)
{
        for (k = 0; k < v[ ].size; k++)
        {
                w = find_winningneuron( v[k], SOM);
                w.vectors.add(v[k]); //add vector to the winning neuron
                //decrease color level means decrease the brightness
                w.colorlevel − −;
                //find interested topics
                for (i = 0; i < n; i++)
                {
                        if (w.weights[i] ≥ threshold of considerable topics)
                                w.considerabletopics.add(i);
                }
        }
}
```

Figure 3.12. Pseudocode of Clustering Procedure

Chapter 4: System Implementation and Testing

This chapter is an answer for the question how to develop the application to test and demonstrate these theories in the real computing system.

This chapter mainly indicates the two parts to implement clustering actors by SOM:

- System Implementation
- System Test

4.1 System Implementation[1]

4.1.1 Programming Framework

In this section, there are several issues need to be considered which are:

- Why Microsoft .NET Framework and C# programming language is chosen?
- How to use parallel programming in Microsoft .NET Framework version 4.0. C# programming language?
- Why Microsoft Windows Presentation Foundation (denoted by WPF) is chosen to design Graphic User Interface?
- What is Microsoft Expression Blend?

To begin with, let get an overview of Microsoft .NET Framework. As we know, Microsoft .NET Framework was introduced in 2002 by Microsoft. This is a viral framework for programming developer with main function is to help programming be simple by providing several build-in function or code such as API to creating windows form so the remaining steps for the developer is just call the framework function to active this build-in code. It means like Java that code of a program is not complied as native code for computer such as C programming language anymore, the code is interpreted to bytecode in order to run in any computing system platform with support this build-in framework.

[1] See Appendices for SOM implementation

So the problem is why we choose .NET Framework instead of Java which is more popular, open-source and can be able to port to Web? The main reason is .NET Framework has more ability to interact with Windows Operating System than Java such as the support of Windows Presentation Foundation for representing Graphic User Interface (GUI) to make the demo become clearly. Secondly, .NET Framework has several powerful tools in editing programming code or designing GUI such as Microsoft Visual Studio and Microsoft Expression Blend, thus, the time to building the demo application is reduced. The last reason is the .NET Framework's programming language. C# is approximately similar to Java so if we want to reuse the code in another computing platform which Java has more advantages to implement such as web platform, it is not so hard to convert and reuse the code.

Alternatively, similar to another framework like Java, such framework from version 4.0 supports basic functions of parallel programming used in the demo application. Here is its fundamental architecture:

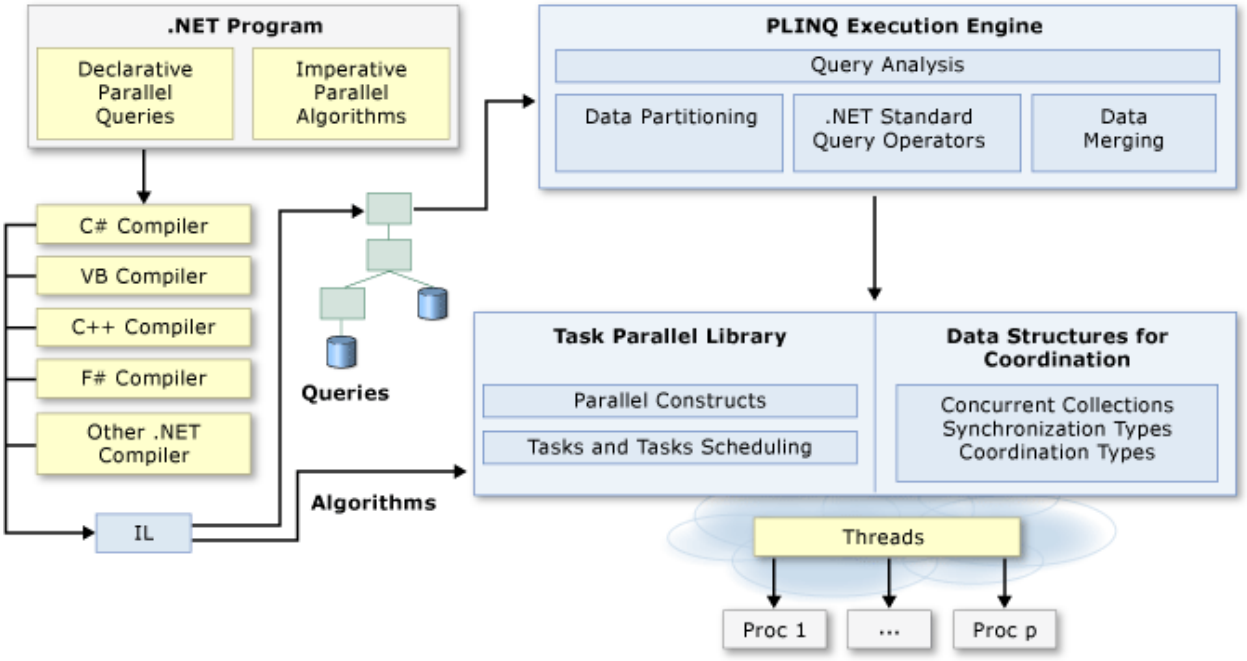


Figure 4.1. Parallel Programming Architecture in the .NET Framework 4.0 [10]

As we can see, the framework provides various kind of parallel programming for processing query in a database or the other algorithms. As simple as the usage of other library, we need to reference to the System.Threading.Tasks or parallel programming library by using keyword with the syntax like the first line of the figure. Then the syntax of calling a parallel function is showed from line 2 to line 5 with n is the greatest value of the increasing iteration variable i, according to the figure.

```
using System.Threading.Tasks; //for parallel processing

Parallel.For(0,n,delegate (int i) //line 2
{
       //your code here
});                                  //line 5
```

Figure 4.2. Parallel "For" Loop's Example in C#

Another simple one is parallel tasks. It means that we create a task contains several statements then we can run them at the same time with the statements outside the task. For instance, we create a task involved an assignment statement such as variable b assigned by 2 and outside the task we have another assignment that variable c assigned by 3, then these assignment are done concurrency during run time. This technique is definitely a suitable manner for solving an issue in the implementation of GUI or the upcoming section.

An example of parallel task is:

```
using System.Threading.Tasks; //for parallel processing
Task a=new Task(()=>{
                    //your code here
                    }));
a.Start();                    //start the task
Console.WriteLine("Ok task");//the outside statements of the task run contemporary
```

Figure 4.3. Parallel Task's Example in C#

On the other hand, the demo application's GUI is based on Microsoft WPF so what is the distinction between WPF and the traditional Windows Form? Let figure an overview of WPF to find the answer.

According to MSDN, WPF is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications [11]. Furthermore, it uses Extensible Application Markup Language (denoted by XAML) based on XML for implementing the appearance of the application. It means that it is more adaptive than the traditional presentation system because we can customize the GUI elements such as button, textbox and so on. In addition, WPF can be used for web application.

The example of WPF application and XAML is:

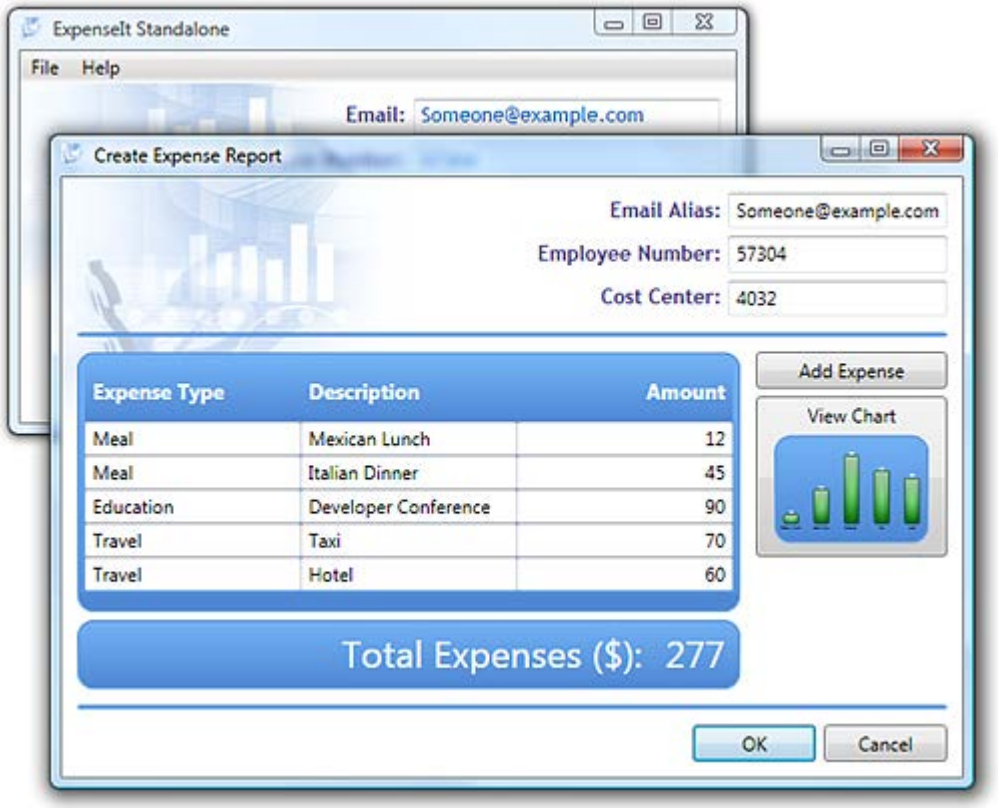


Figure 4.4. Example of WPF Application [11]

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Title="Window with Button"
    Width="250" Height="100">

  <!-- Add button to window -->
  <Button Name="button">Click Me!</Button>

</Window>
```

Figure 4.5. An Example of XAML [11]

In order to make GUI design more simple, Microsoft provides Microsoft Expression Blend for this purpose. It means that the designer do not need to know about XAML, they just need to know how to use Expression Blend and the program can generate the XAML code for them. Nevertheless, since Visual Studio 2012 was introduced, Expression Blend has been integrated in this development software that why Microsoft Expression Blend version 4 is the newest version at this time.

A screenshot of Microsoft Expression Blend 4 is

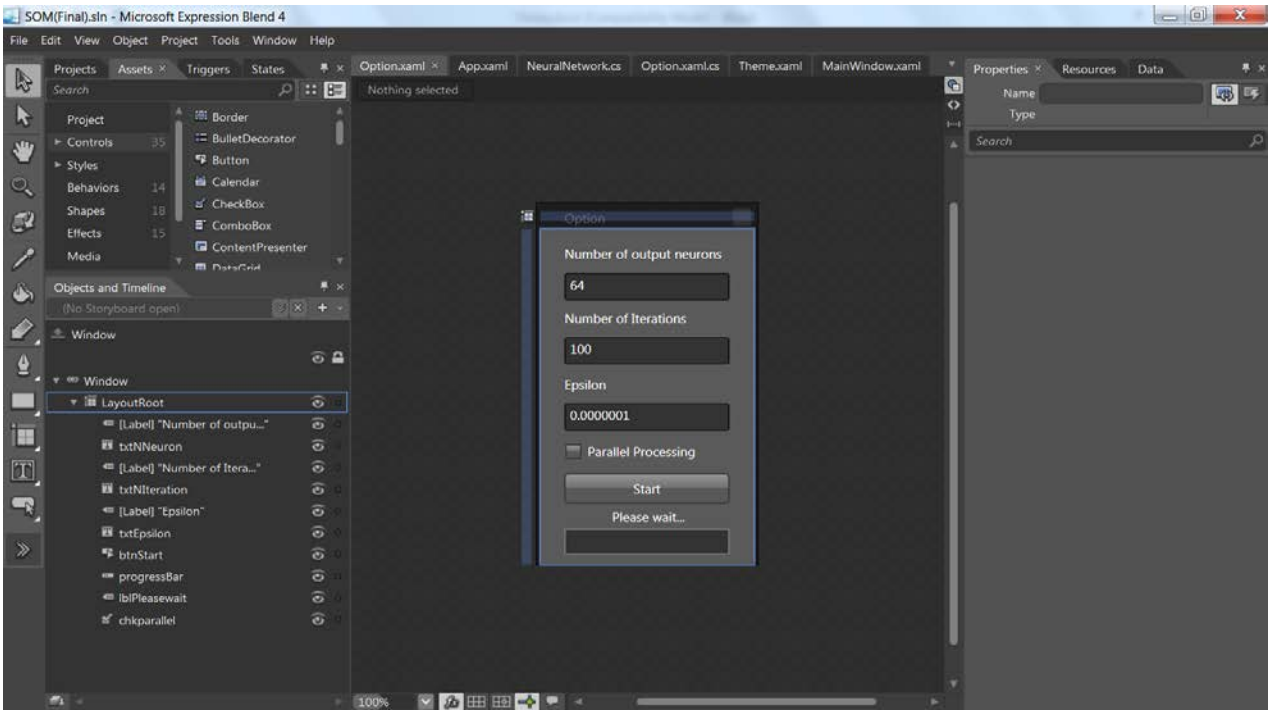


Figure 4.6. Microsoft Expression Blend 4

As we can see, it is approximately similar to the traditional designing tool for Windows Form - Visual Studio. However, it has some extensive function such as customize an element such as element shape or animation and create or apply a specific theme.

4.1.2 Graphic User Interface Implementation

To begin with, the basic functions of the demo application are:

- Load training set file to train the SOM with the changeable of parameters.
- Import and Export the trained SOM to/from a file.
- Represent the SOM with the brightness of color represents the number of actor in neural element.
- Show all actors in a neural element by clicking mouse left button, for right click, show the interested topics.

Thus, this is the main interface of the application:

1. Open button (on top menu): By clicking on this button, users can be able to open training set file by Open File Dialog, then the Option Dialog appears to help user configure arguments in order to start learning process and the clustering process.
2. Save button (on top menu): By clicking on this button, users can export the current trained SOM to a text file for future usage.
3. Self. Organized Maps (on detail pane): This is an array of buttons which is automatically generated after finishing the clustering process. In this case, each button represents an element of the SOM and its color shows the number of actors belongs to this element. The list of actors will be showed in Details Form if user clicks left mouse button, vice versa, on the other hand, the list of topics will be showed for right click. If users run the demo application in the first time, the detail pane is empty because there is no trained SOM.

The following figure is the main interface of the demo application with the number denoting each function which we have already discussed:

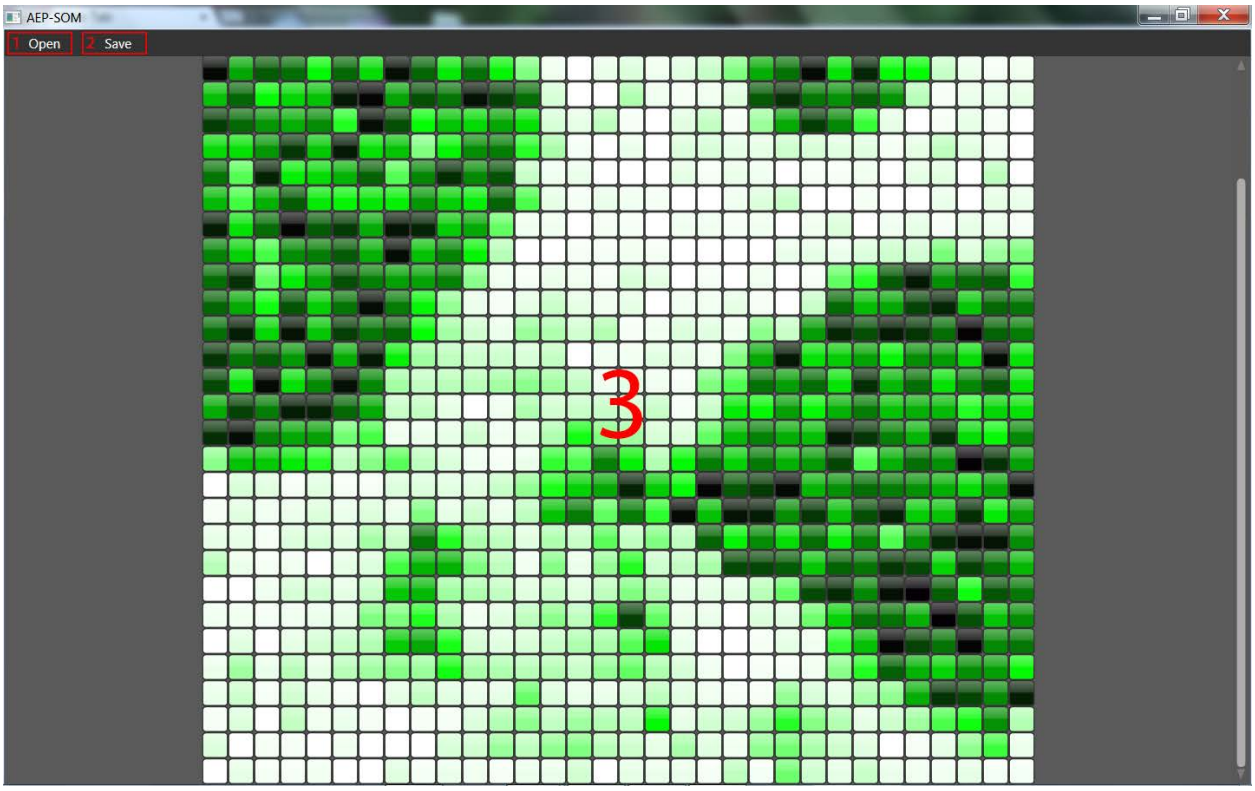


Figure 4.7. The main interface of the demo application

Note: We use the ExpressionDark theme to decorate the demo application. That's why all of the buttons, menus, scrollbars, textbox, progress bars and other components are in the dark theme.

Assume that users click the open button to load the input file, they should see Open File Dialog shown in figure 4.8:

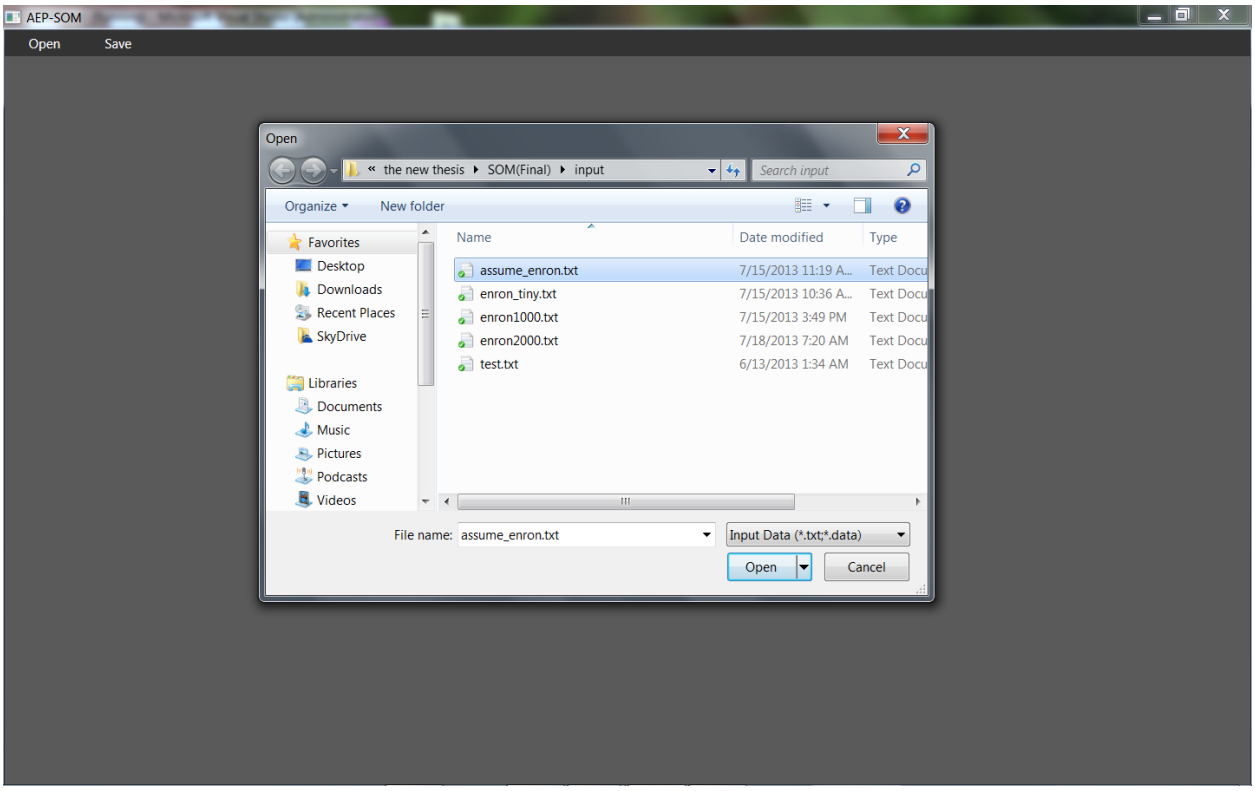


Figure 4.8. Open File Dialog for Training Set Input File

Then, Options Form appears:

1. By choosing option “Load Existing Network”, the browse button is available for users to load the trained SOM from a text file in order to skip the learning process.
2. By choosing option “New Network”, users can be able to modify the value of the number of neuron. The number of elements in the SOM, number of iterations. The maximum value of number of iterations in learning process and epsilon. The maximum value of epsilon or the average different weight value between the before updating weight value and the after updating weight value. As we know, both values are the stop condition for the learning process.
3. The options of clustering process which are always available because they are options for clustering process. It contains a threshold for determining interested topics, the color step value for coloring the SOM’s buttons, and whether users desire to use parallel processing in clustering process or not.

4. The start button to begin all processes.
5. The progress bar which is used to indicate the current status of the learning process.

To be more understandable, figure 4.9 describes all of these functions are denoted as number as follow:

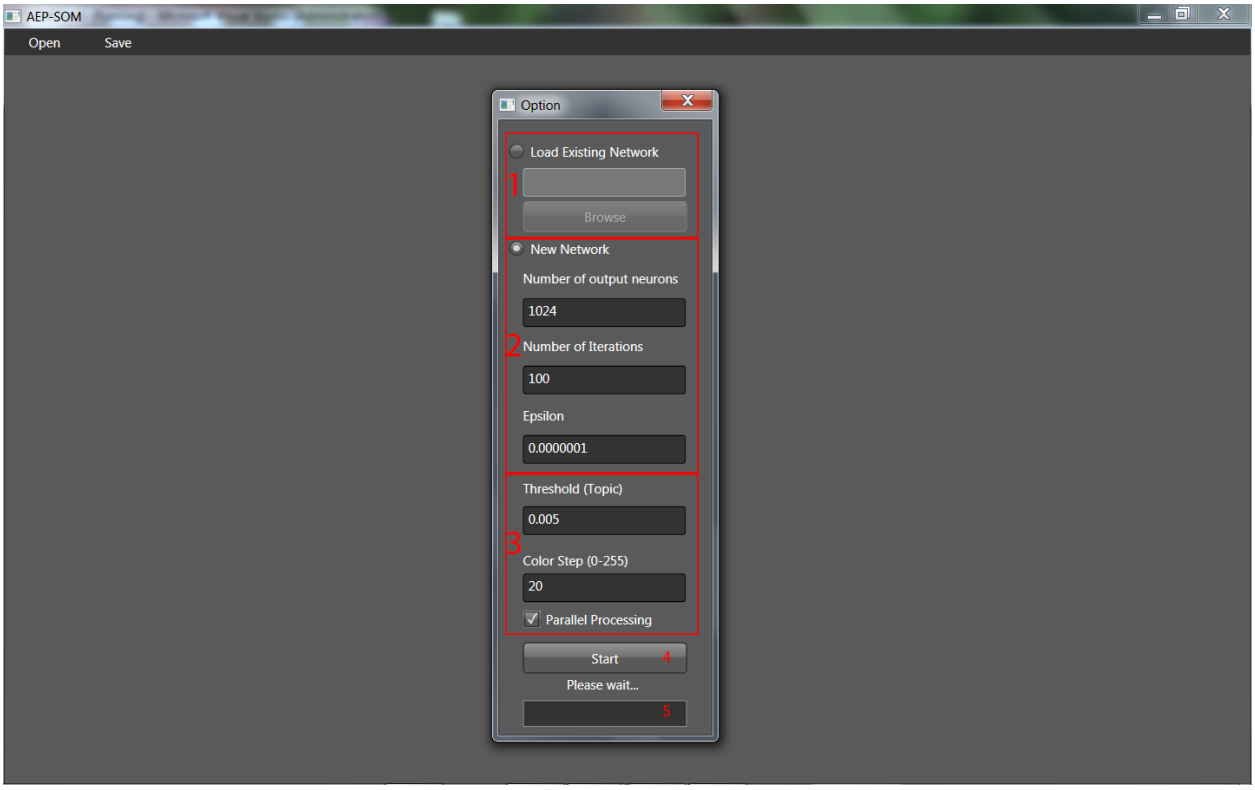


Figure 4.9. Option Form

After that, if all the processes have been succeeded, users can see this with the MessageBox cited the running time:

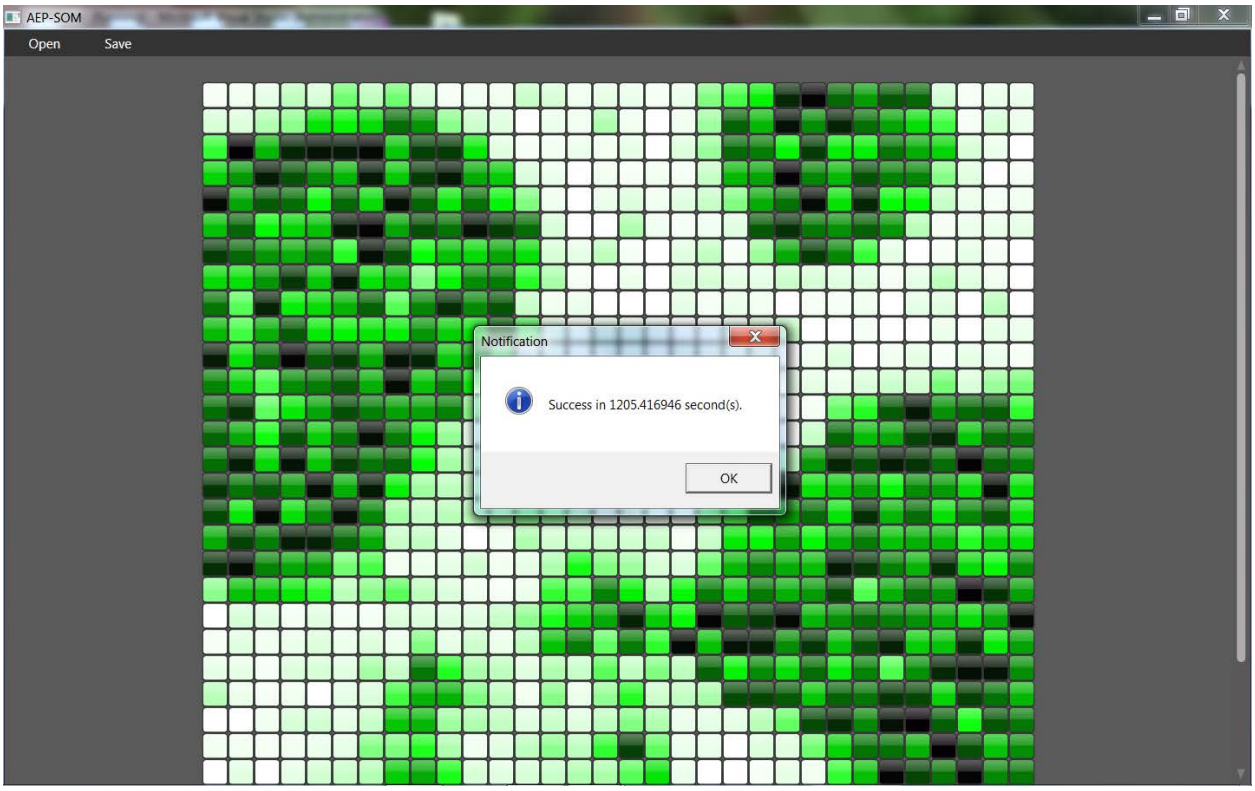
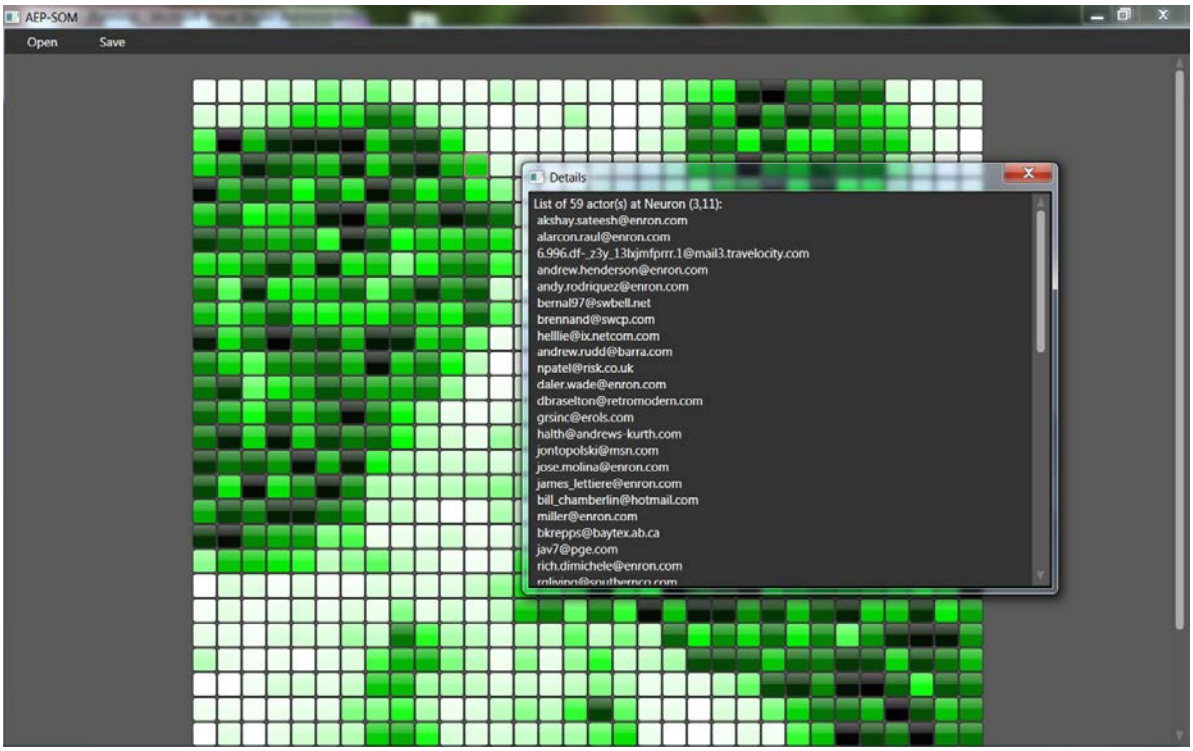


Figure 4.10. SOM has been trained, clustered and generated

Finally, users can be able to click left mouse button or right mouse button on the neural elements in order to get information of the actors in it or its interested topics as follows:



interested topic

Figure 4.11. Left click to view list of actors

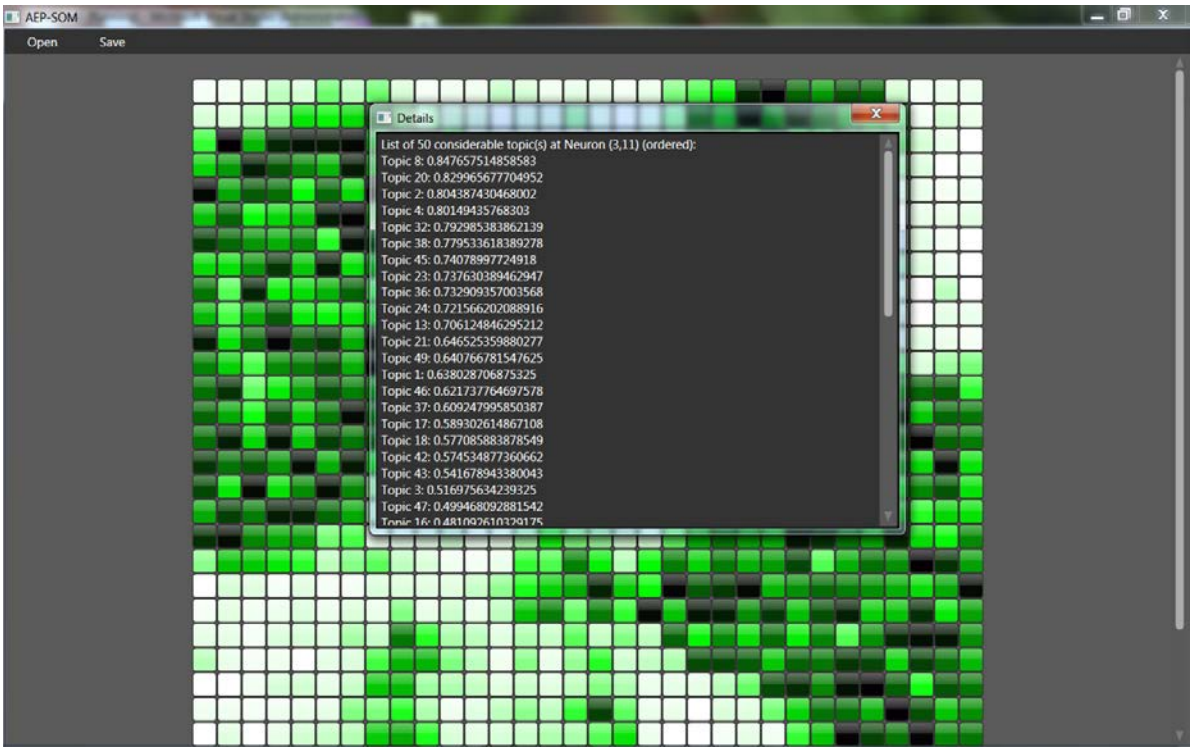


Figure 4.12. Right click to view interested topics

In addition, the trained SOM can be exported to a text file by click on save button, then the save file dialog appears:

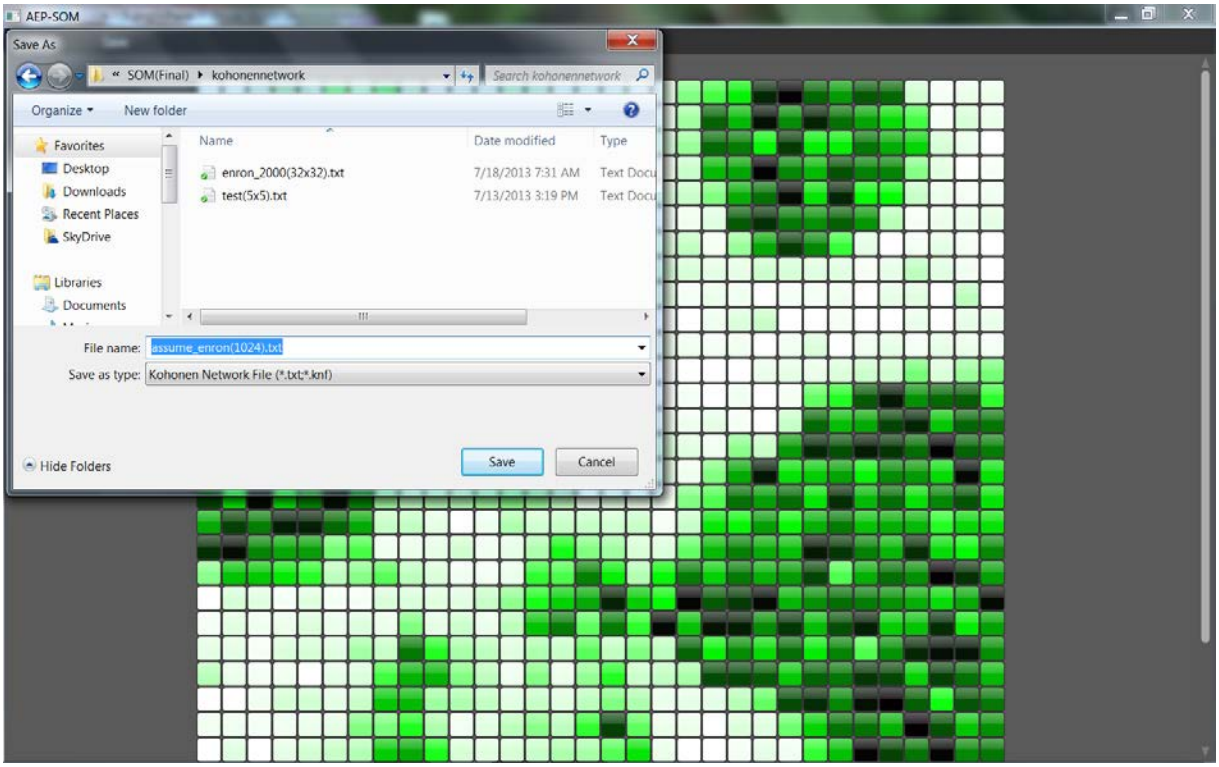


Figure 4.13. Save File Dialog

In addition, the structure of these forms is:

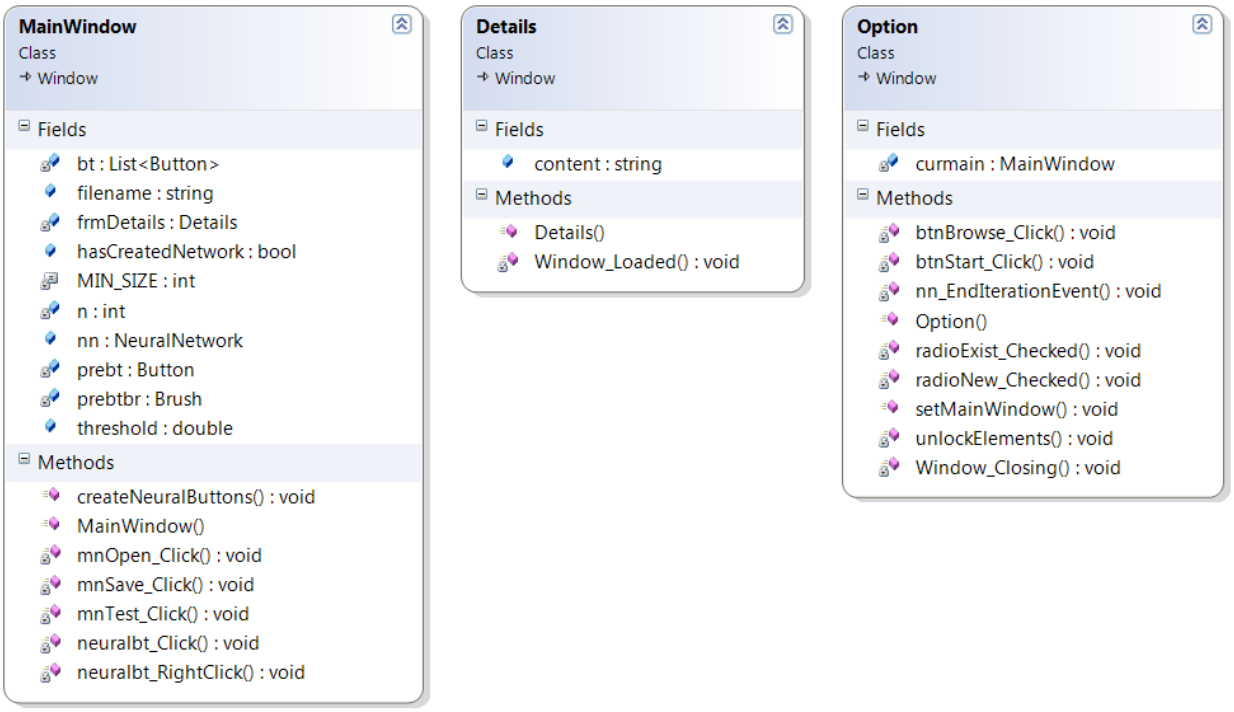


Figure 4.14. MainWindow, Details and Option Form Classes

The description of variables and functions are:

Table 4.1. GUI-Variable's Description

Variable (Field)	Description
bt	List of neural buttons.
filename	The training set input file name.
frmDetails	Details form used to show the number of actors, interested topics.
hsaCreatedNetwork	The value of epsilon at the recent epoch.
MIN_SIZE	The size of a neural element in the SOM.
n	The number of rows or columns of the SOM.
nn	An instance of Neural Network Class.
prebt	The selected neural buttons.
prebtbr	The current border brush of prebt.
threshold	The threshold to distinguish interested topics.

content	The content string of Detail Form.
curmain	Point to the MainWindow Form.

Table 4.2. GUI-Method's Description

Method	Description
createNeuralButtons()	Create neural buttons with colors.
MainWindow()	Constructor.
mnOpen_Click()	For event of clicking on open button, prompt the open file dialog then show Option Form.
mnSave_Click()	For event of clicking on save button, export the current SOM to a text file with the file name provided by Save File Dialog.
neuralbt_Click()	For event of clicking left mouse button on any neural button, Details Form showed with the content of actors belong to this SOM's element.
neuralbt_RightClick()	For event of clicking left mouse button on any neural button, Details Form showed with the content of interested topics of this SOM's element.
Details()	Constructor.
Window_Loaded()	It occurs when Details Form is loaded. The purpose is to change the content of textbox to content variable.
btnBrowse_Click()	For event of clicking on the Browser button, open the load file dialog to help user indicate the current SOM file.
btnStart_Click()	Execute all processes as configuration and represents the status by calling functions of instance nn.
nn_EndIterationEvent()	It occurs when the learning iteration has been completed. The value of progress bar increase by 1 o represents the status.
Option()	Constructor.

radioExist_Checked()	For event of selecting option of load existing network, it locks the unnecessary elements and unlocks the appropriate options.
radioNew_Checked()	In the same vein with the previous one, but in this case for the event of selecting option of create a new SOM.
setMainWindow()	Set the pointer curmain to a specific MainWindow form.
unlockElements()	Lock or unlock necessary elements in the Form. Ex: Lock elements while training and clustering, unlock when everything has been completed.
Window_Closing() (Option Form)	For event of closing the form, the manner is to prevent users close the form while it is running.

In order to express the status of the entire procedure by the progress bar in Option Form, we use another technique of parallel processing called task. In this case, we create a task for running the entire procedure then let it run separately with the Option Form. As a result, users can see the progress of training and whether the progress bar changes to be determinate, it means the clustering process and the creating neural buttons process is running.

4.2 System Test

4.2.1 Testing and Assessing Procedure

To begin with, we go back to the algorithms of clustering actors in social network which has been discussed in chapter 3. The prerequisite of our processes is the encoding process which is based on ART model or AT model in this case, thus, the testing and assessment procedure is these steps.

Firstly, we find a corpus of a large number (up to 100) of messages which are discussed in social network in English language. (Due the fact that the encoding application just

supports this language at this time and English source is easier to look for through the internet than other language)

The next step is pre-encoding this corpus to match the input of the encoding application which are two matrices. The first one is the matrix of emails, senders, and recipients and the last one is the matrix of documents and words.

Then, running the encoding application based on ART model or AT model to get the output which contains:

- The top n topics of all senders.
- The top n topics of all recipients.
- The distribution of senders in each topic.

After that, the post-encoding requires to create a list of vectors which their names are the senders name, the number of dimensions is the number of topics configured in the encoding process, the dimensional index is the topic number, and the dimensional value is the distribution in this topic.

Then we train the SOM and use it to cluster the input by running the demo application with the list of vectors created by the post-encoding step.

Finally, we compare the result with the man-made result. That is a reason why in testing the accuracy, we use 3 measurements: precision, recall and f-measure to compute the accuracy. [12]

In addition, in order to compare the accuracy of each topological neighborhood function, we need to compute the clustering results of several functions used in training procedure with 3 assumed tests. Then, we compare these results to find out which one is the most appropriate.

4.2.2 Working with Enron Corpus

According to Wikipedia, Enron Corpus is a large database of over 60000 emails generated by 158 employees of the Enron Cooperation and acquired by the Federal

Energy Regulatory Commission during its investigation after the company's collapse. A copy of the database was subsequently purchased by Andrew McCallum, a computer scientist at the University of Massachusetts Amherst. He released this copy to researchers, providing a trove of data that has been used for studies on social networking and computer analysis of language. The corpus is "unique" in that it is one of the only publicly available mass collections of "real" emails easily available for study, as such collections are typically bound by numerous privacy and legal restrictions which render them prohibitively difficult to access. [13]

This is a reason why we use Enron Corpus for testing and assessing the result of the thesis because this corpus satisfies our requirement.

On the other hand, there is a MySQL database version of Enron Corpus which is developed by two professors Jitesh Shetty and Jafar Adibi at University of Southern California. The duplicate and corrupt messages are cleaned in this version. [14]

This is its database schema:

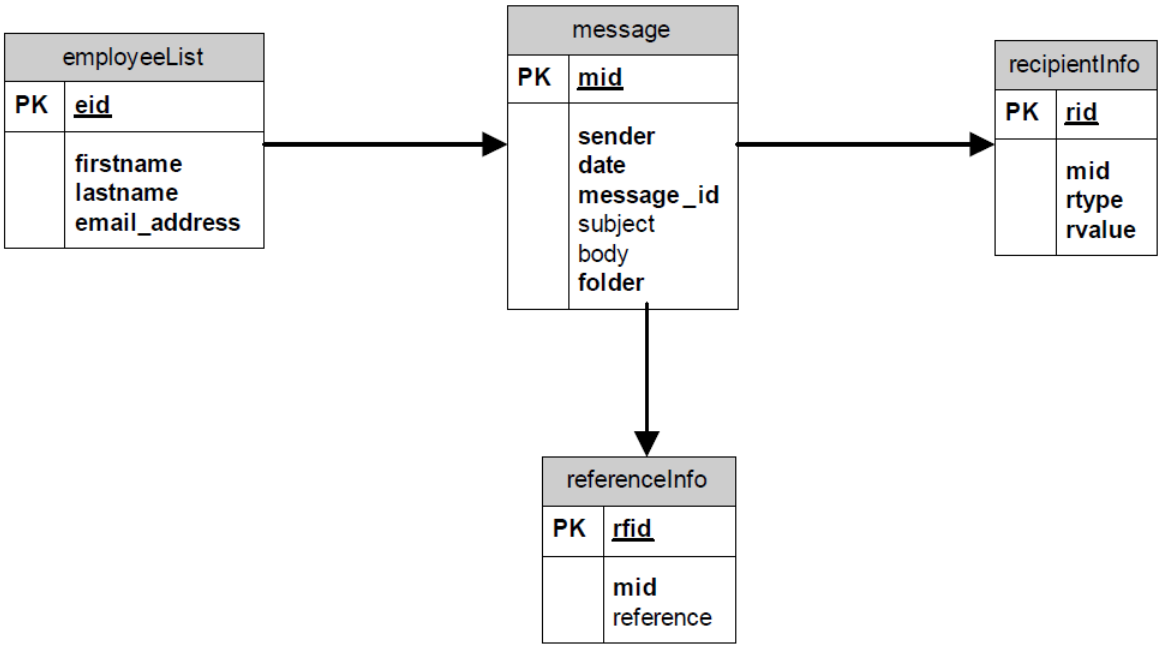


Figure 4.15. Database schema of Enron Database [15]

The additional information about this corpus such as the description of each field and statistical report appears in the reference document.

From the schema, to create a matrix of email-author -recipient, attribute mid, attribute sender (email address) of Message Entity and attribute rid, attribute rvalue (email address) of Recipientinfo Entity could be extracted from the database.

On the other hand, to create a matrix of word distribution, a couple of attributes subject and body of message entity need to be extracted then we use an algorithm to extract words from this data and compare with our build-in dictionary to count the frequencies of words in each documents. That is how we create a matrix of word distribution.

```
select
   enron.message.mid,
   enron.message.sender,
   enron.recipientinfo.rvalue
from
   enron.message
      inner join
   enron.recipientinfo ON enron.message.mid = enron.recipientinfo.mid
```

Figure 4.16. Query for creating a matrix of email-author-recipient

The outcome of the encoding application is the file contains all of the distribution of each actor in each topic. Thus, it is not tough for extracting these values to create a set of vectors file input for the demo application, we can extract by ourselves or create an extractor. However, in this case for large number of values, we develop an extractor in C# for this manner.

We have 3 training set to test the demo application:

- Enron Tiny: the actual outcome of the encoding application with 16 actors.

- Enron: the actual outcome of the encoding application with 147 actors.
- Assumed Enron Simple: the assumed set with 100 actual actors in Enron Corpus and the assumed dimensional values and 10 topics. In this case, we have 10 types of actors and each type contains 10 actors. The characteristic of each type is that the actors who belong to a type must have the same interested topic. For instance, Type 0 contains all actors which have the weight at least 0.5 for Topic 0 and 0 for the other topics.
- Assumed Enron Triple: The set has the same size as Assumed Enron Simple. The unique difference is that instead of having only one interested topic, each type is interested in 3 topics.
- Assumed Enron Complex: The set has 1000 actors, 10 topics and 10 types of actors. The complex of the set is that the number of actors in each type is not the same value, and each type has different number of dominant topics. For example, Type 0 we have 20 actors and the actors only consider Topic 4. On the other hand, Type 1 has 90 actors and the topics have weight greater than 0 are Topic 1, Topic 2 and Topic 7.
- AT Enron: The set extracted from the result of AT Model from University of California, Irvine.[2] In details, it has 1080 actors and each one has 200 topics.

4.2.3 Precision, Recall and F-Measure [12]

Assume that in set of actors we divide these actors into m clusters of actors by man-made. On the other hand, by using SOM, the set is split to k clusters.

Then at cluster m_i and cluster k_i we have:

- A: the set of actors belong to both clusters. ($m_i \cap k_i$)
- B: the set of actors only belong to m_i. ($m_i \setminus k_i$)
- C: the set of actors only belong to k_i. ($k_i \setminus m_i$)

Or: $|m_i| = |A| + |B| = a + b$ and $|k_i| = |A| + |C| = a + c$

[2] http://www.datalab.uci.edu/author-topic/

Thus, we have these measures:

Precision measure represents the ratio of the accuracy of a SOM cluster. If the ratio is 1, it means that all the actors in SOM cluster belong to cluster m_i, or $k_i \subset m_i$. We have:

$$P = \frac{a}{a + c}$$

Recall measure describe the ratio of the accuracy of a true positive actors clustered by SOM with the actual positive actors. If it is 1, $m_i \subset k_i$.We have:

$$R = \frac{a}{a + b}$$

According to Brew & Schulte im Walde (2002), F-Measure, which is the combination of Precision and Recall, is used to compute the accuracy of the system. For the clustering system, this is the equation:

$$F = \frac{2PR}{P + R}$$

The greater value F-Measure has, the more accurate the SOM is.

Example: (The result of Assumed Triple Enron with the discrete function, a cluster with all values NaN means empty)

Precision Table:

	m_0	**m_1**	**m_2**	**m_3**	**m_4**	**m_5**	**m_6**	**m_7**	**m_8**	**m_9**
k_0	1	0	0	0	0	0	0	0	0	0
k_1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
k_2	0	0	0	0	1/11	0	0	10/11	0	0
k_3	0	0	0	0	1	0	0	0	0	0
k_4	0	0	0	0	0	1	0	0	0	0
k_5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

$\mathbf{k_6}$	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
$\mathbf{k_7}$	0	0	0	0	0	0	0	0	0	1
$\mathbf{k_8}$	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
$\mathbf{k_9}$	0	9/19	10/19	0	0	0	0	0	0	0
$\mathbf{k_{10}}$	0	1	0	0	0	0	0	0	0	0
$\mathbf{k_{11}}$	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
$\mathbf{k_{12}}$	0	0	0	1	0	0	0	0	0	0
$\mathbf{k_{13}}$	0	0	0	0	0	0	0	0	1	0
$\mathbf{k_{14}}$	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
$\mathbf{k_{15}}$	0	0	0	0	0	0	1	0	0	0

Recall Table:

	$\mathbf{m_0}$	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$	$\mathbf{m_4}$	$\mathbf{m_5}$	$\mathbf{m_6}$	$\mathbf{m_7}$	$\mathbf{m_8}$	$\mathbf{m_9}$
$\mathbf{k_0}$	1	0	0	0	0	0	0	0	0	0
$\mathbf{k_1}$	0	0	0	0	0	0	0	0	0	0
$\mathbf{k_2}$	0	0	0	0	1/10	0	0	1	0	0
$\mathbf{k_3}$	0	0	0	0	9/10	0	0	0	0	0
$\mathbf{k_4}$	0	0	0	0	0	1	0	0	0	0
$\mathbf{k_5}$	0	0	0	0	0	0	0	0	0	0
$\mathbf{k_6}$	0	0	0	0	0	0	0	0	0	0
$\mathbf{k_7}$	0	0	0	0	0	0	0	0	0	1
$\mathbf{k_8}$	0	0	0	0	0	0	0	0	0	0
$\mathbf{k_9}$	0	9/10	1	0	0	0	0	0	0	0
$\mathbf{k_{10}}$	0	1/10	0	0	0	0	0	0	0	0
$\mathbf{k_{11}}$	0	0	0	0	0	0	0	0	0	0
$\mathbf{k_{12}}$	0	0	0	1	0	0	0	0	0	0
$\mathbf{k_{13}}$	0	0	0	0	0	0	0	0	1	0

k_{14}	0	0	0	0	0	0	0	0	0	0
k_{15}	0	0	0	0	0	0	1	0	0	0

F-Measure:

	m_0	**m_1**	**m_2**	**m_3**	**m_4**	**m_5**	**m_6**	**m_7**	**m_8**	**m_9**
k_0	1	0	0	0	0	0	0	0	0	0
k_1	0	0	0	0	0	0	0	0	0	0
k_2	0	0	0	0	2/21	0	0	20/21	0	0
k_3	0	0	0	0	18/19	0	0	0	0	0
k_4	0	0	0	0	0	1	0	0	0	0
k_5	0	0	0	0	0	0	0	0	0	0
k_6	0	0	0	0	0	0	0	0	0	0
k_7	0	0	0	0	0	0	0	0	0	1
k_8	0	0	0	0	0	0	0	0	0	0
k_9	0	18/29	20/29	0	0	0	0	0	0	0
k_{10}	0	2/11	0	0	0	0	0	0	0	0
k_{11}	0	0	0	0	0	0	0	0	0	0
k_{12}	0	0	0	1	0	0	0	0	0	0
k_{13}	0	0	0	0	0	0	0	0	1	0
k_{14}	0	0	0	0	0	0	0	0	0	0
k_{15}	0	0	0	0	0	0	1	0	0	0
MAX	**1**	**18/29**	**20/29**	**1**	**18/19**	**1**	**1**	**20/21**	**1**	**1**

Total value: $1 + \frac{18}{29} + \frac{20}{29} + 1 + \frac{18}{19} + 1 + 1 + \frac{20}{21} + 1 + 1 \approx 9.2100942$

4.2.4 Assessment of the Clustering Accuracy of Assumed Enron

In this section, we uses three assumed Enron sets (Simple, Triple and Complex) to test the accuracy of each topological functions. The number of output neuron is 16 for most cases except the additional situation we enlarge the number to 36 to test Complex set. The number of iteration is 1000 and $\varepsilon = 10^{-7}$.

Each time we change the neighborhood function, we compute the table of Precision, Recall, then manipulate the total F-measure. Finally, the chart is sketched to provide a visual view of the accuracy of each situation:

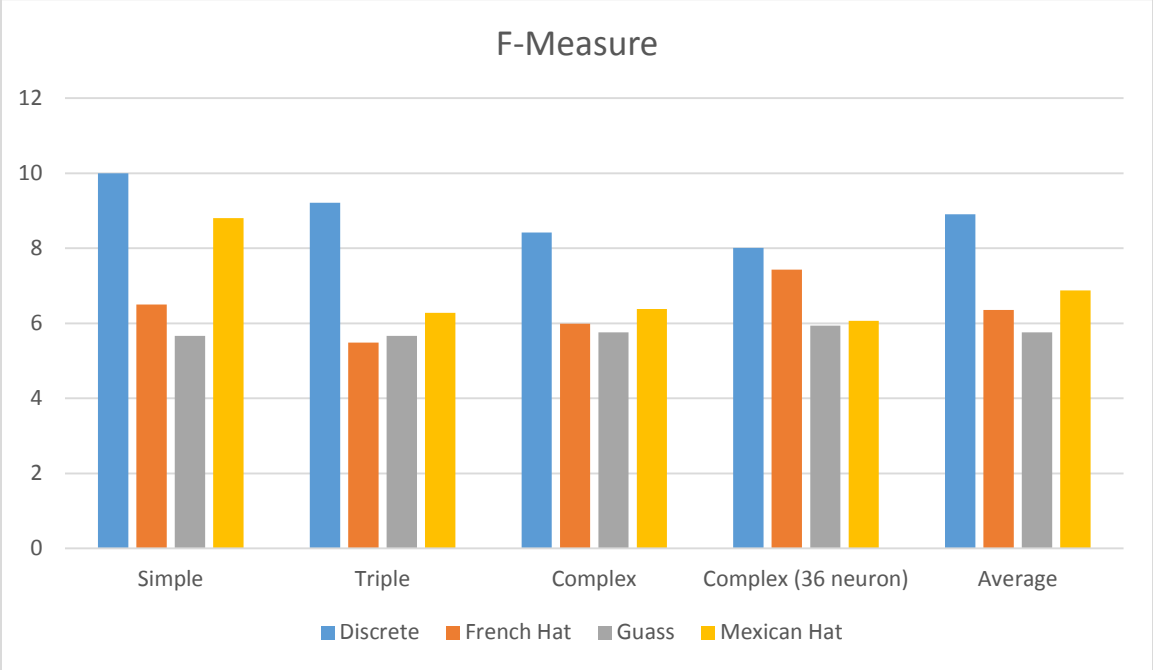


Figure 4.17. The F-Measure when using a specific topological function

As we can see, the discrete function has the best result with these sample tests with the total F-Measure is always greater than 8.[3] Therefore, this function is chosen for the upcoming testing.

[3] The result of these tests contains in the CD-ROM (Excel Files)

4.2.5 Result of Clustering Enron Sample Tests

First of all, we begin with Enron Tiny. With setting 64 output elements and color step is 30, this is a result after running the demo application:

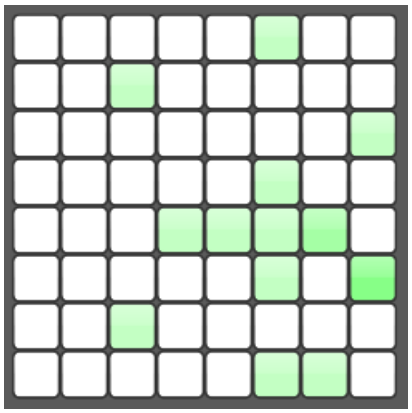


Figure 4.18. The SOM of Enron Tiny

By man-made clustering, we have 15 clusters. On the other hand, 13 clusters are created by the SOM. As a result, the F-Measure is 13.6333.

For Enron Set with 147 actors with 129 inactive actors who have all the dimensional values are 0:

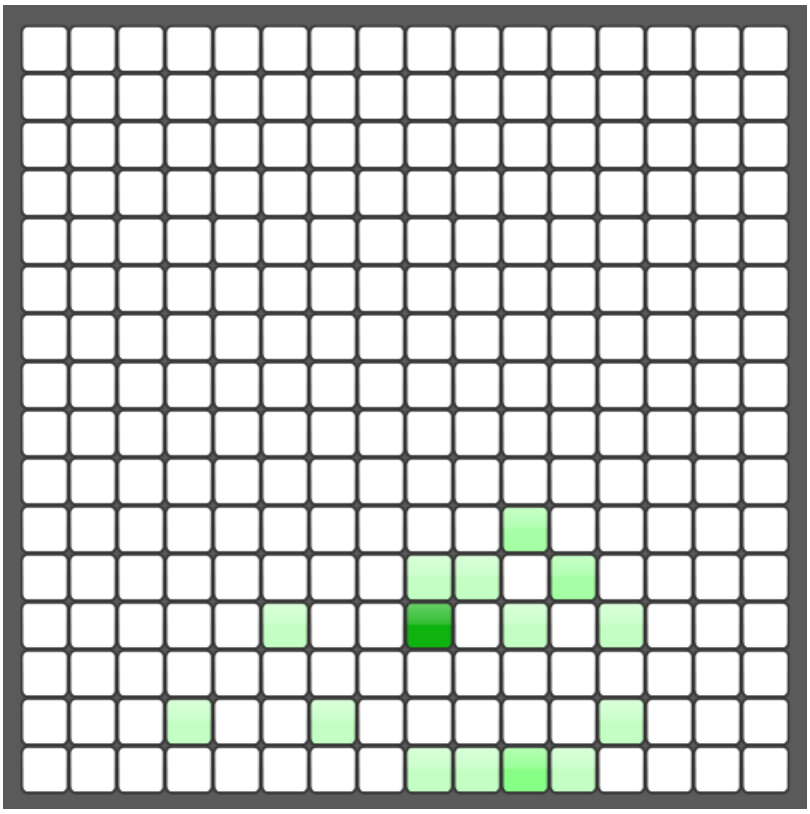


Figure 4.19. The Resulted SOM of 127 Actors Enron

In this case, with the number of output neuron is 256, the total F-measure is 16.166667. (We have 18 clusters of actors by man-clustering)

For AT Enron, by using the SOM to cluster, we have 262 clusters and the total F-meaure is 229.9254:

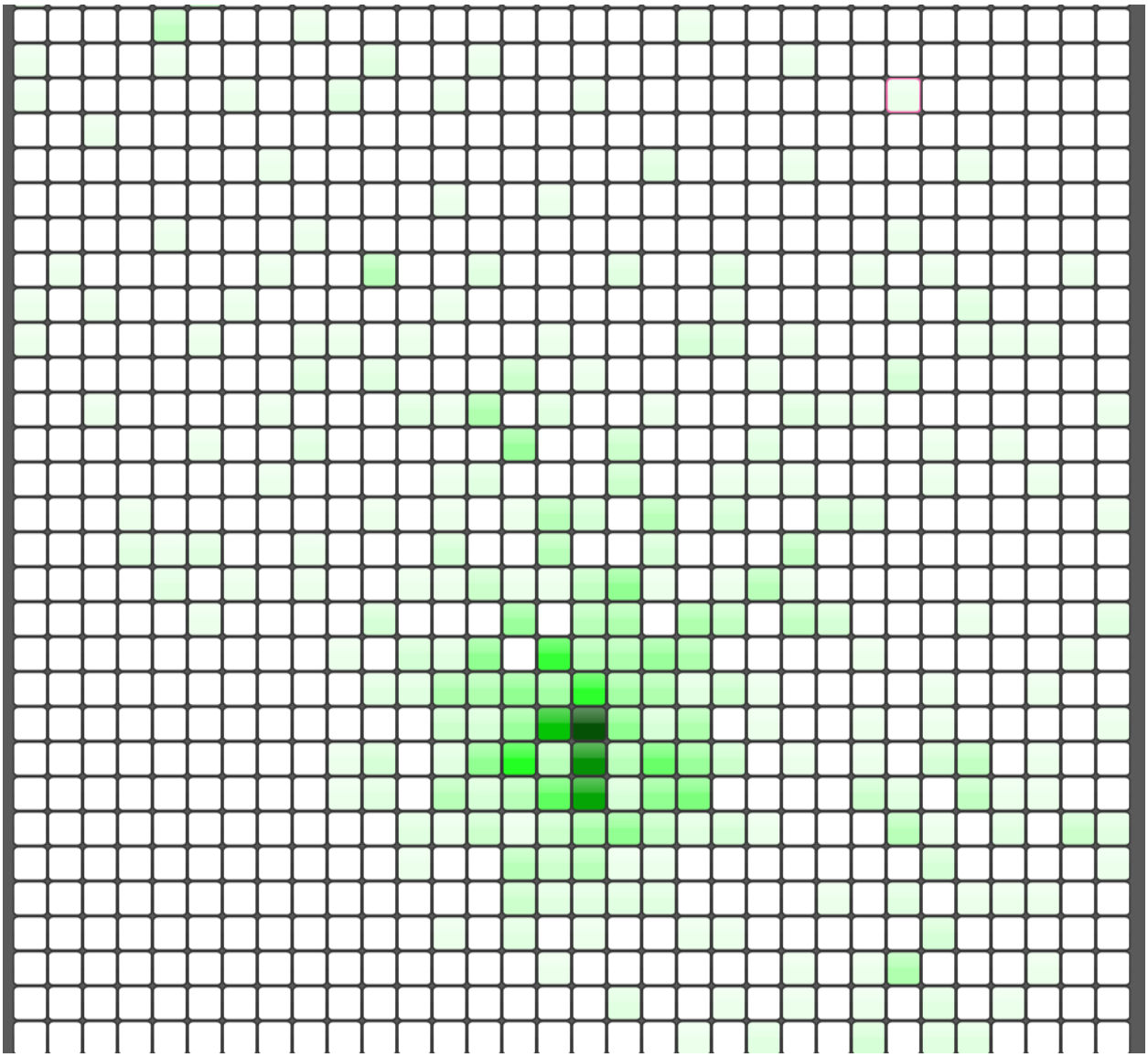


Figure 4.20. The Resulted SOM of AT Enron

In some cases, the coloring elements seem not perfect. The issue is our color bold value just has 500 levels in maximum with COLORSTEP is 1 which is the smallest. Hence, if the COLORSTEP is too small, the clusters contain few actors become hard to distinguish with the empty one. On the other hand, if the value is large, the clusters contain huge actors could be reduce bold level because the out of level issue.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

In summary, the thesis presents an overview of Social Network and its popularity in the modern society. As a result, it becomes a fascinating field for scientists and researchers. Alternatively, our problem has been introduced as well as the solution methodology, and the fundamental theories of basic terms such as Social Network, networking model and Kohonen Network.

In the main chapters of the thesis, it indicates in depth the encoding and clustering procedures also our enhancement by explaining the algorithms with example, pseudocode, and trying to compute by hand.

After that, the thesis states in details of the structure of the demo application which is the combination of the implementation of the SOM and the GUI and developed in C# programming language in .NET Framework 4.0. In the same vein, the testing and assessing procedure and the testing database also has been cited.

With the acceptable total F-measure value, the clustering actors' method by SOM has a potential to deploy in the actual social network system.

5.2 Future Work

The demo application is just for exhibiting the algorithms of clustering actors in social network based on the interested topics and based on self-training Kohonen Network or SOM. Therefore, to be a real system, there are several works need to be done.

Due to the fact that the web application is widespread for mainframe and Java is one of the most programming languages for building web application as well as supporting parallel programming. Firstly, it is necessary to do is to develop an web application instance for clustering actors based on the interested topics by Kohonen Network to use efficiently in mainframe computing system.

After that, the second thing we desire to work is integrate all document encoding procedures into this instance. As a result, the input of the system is a set of messages and the output is the SOM like the demo application but it represents via internet browsing application.

Then, the third thing to do is let the system be able to work with an actual Social Network's data. In this case, building a social network for a lab based on open source such as Mahara must be considered. Furthermore, the system must overcome the out of memory issue during computing this huge amount data.

Fourth, improving parallel processing and language support must be desirable. For improving parallel processing, the system should be able to process in parallel in training procedure. In addition, the improved algorithm should be able to solve the conflict of updating value in parallel. For improving language support, encoding process should be support several languages such as French, Chinese and so on. The solution for this issue is that we enlarge the encoding word dictionary for these languages.

Finally, developing a module to name the topics must be considered. The solution to develop this module is finding the distance of keywords with a specific measurement such as finding the least common hyponym of a couple of word in WordNet Dictionary.

REFERENCES

[1] N. Pathak, C. Delong, K. Erickson and A. Banerjee, "Social Topic Models for Community Extraction," Minneapolis.

[2] T. Kohonen and T. Honkela, "Kohonen network," [Online]. Available: http://www.scholarpedia.org/article/Kohonen_network.

[3] "Social Network," [Online]. Available: https://en.wikipedia.org/wiki/Social_network.

[4] "What is social network," Webopedia, [Online]. Available: http://www.webopedia.com/TERM/S/social_network.html.

[5] "Top 15 Most Popular Social Networking Sites," July 2013. [Online]. Available: http://www.ebizmba.com/articles/social-networking-websites.

[6] A. McCallum, A. Corrada-Emmanuel and X. Wang, "The Author-Recipient-Topic Model for Topic and Role Discovery in Social Networks: Experiments with Enron and Academic Email," Amherst, 2005.

[7] K. Lagus, T. Honkela, S. Kaski and T. Kohonen, "WEBSOM for Textual Data Mining," Helsinki.

[8] A. McCallum, A. Corrada-Emmanuel and X. Wang, "Topic and Role Discovery in Social networks," Amherst.

[9] B. Magomedov, "Self-Organizing Feature Maps (Kohonen maps)," 7 November 2006. [Online]. Available: http://www.codeproject.com/Articles/16273/Self-Organizing-Feature-Maps-Kohonen-maps.

[10] "Parallel Programming in the .NET Framework," [Online]. Available: http://msdn.microsoft.com/en-us/library/dd460693.aspx.

[11] "Introduction to WPF," [Online]. Available: http://msdn.microsoft.com/en-us/library/aa970268.aspx.

[12] M. X. Hùng, “GOM CỤM VĂN BẢN BẰNG MẠNG SOM DỰA TRÊN CÁC CỤM DANH TỪ TRONG VĂN BẢN TIẾNG VIỆT,” 2006.

[13] "Enron Corpus," [Online]. Available: http://en.wikipedia.org/wiki/Enron_Corpus.

[14] J. Adibi, "Enron Dataset," [Online]. Available: http://www.isi.edu/~adibi/Enron/Enron.htm.

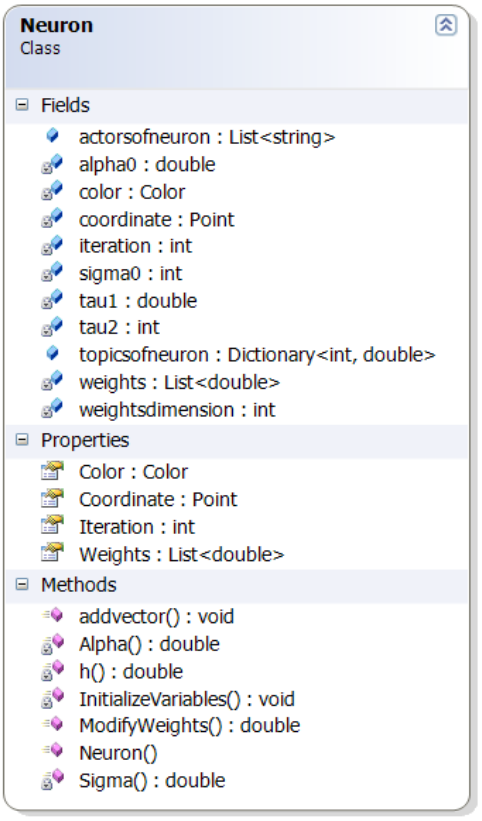
[15] J. Shetty and J. Adibi, "The Enron Email Dataset Database Schema and Brief Statistical Report".

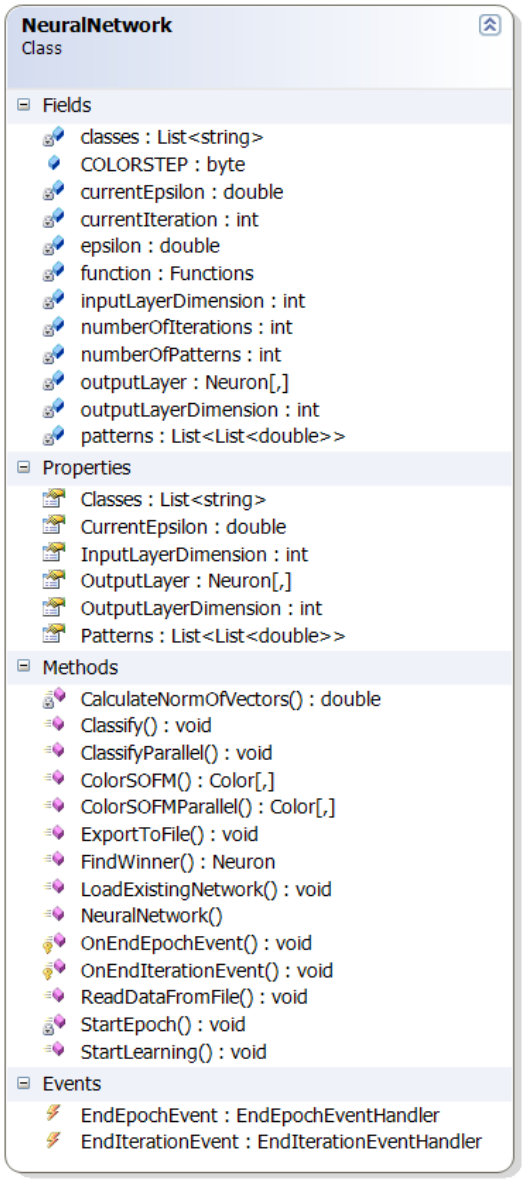
[16] J. Heaton, Introduction to Neural Network with Java, Chesterfield, Missouri: Heaton Research, Inc., 2005.

APPENDIX

SOM Implementation

In this implementation, we have two main class which are Neuron and NeuralNetwork. Neuron class contains methods and characteristics for a SOM node such as weights, colors and modifying function and NeuralNetwork class represents SOM or a matrix of Neuron. The figure below indicates in depth of these classes:





Let begin with the Neuron class first. These tables below explain the role of variables or fields, properties and methods:

Variables in Neuron Class

Variable (Field)	Description
actorsofneuron	List actors belong to the neuron.
alpha0	The initial value of alpha. (alpha0=0.1)
color	The color value of the neuron which has been modified by clustering process.
coordinate	The index of row and column output neuron in SOM. The value of index is integer. For example: (1, 1) or row 1, column 1.
iteration	The current iteration.
sigma0	The initial value of sigma. (sigma0=width or length of SOM)
tau1	The value of tau1 in sigma function. (tau1=1000/Log(sigma0))
tau2	The value of tau2 in alpha function. (tau2=1000)
topicsofneuron	The dictionary which index is the index of topic such as 0, 1, 2, etc. and the value is the weights of the neuron at each topics. (Use for sorting by weights)
weights	The weights of output neuron.
weightsdimension	The total number of topics or number of dimensions of each weight of output neuron.

Properties in Neuron Class

Property	Description
Color	Get or set value of variable color.
Coordinate	Get or set value of variable coordinate.
Iteration	Get or set value of variable iteration.

Weights	Get or set value of variable weights.

Methods in Neuron Class

Method	Description
addvector()	Add actors to variable actorsofneuron.
alpha()	Alpha function.
h()	Neighborhood function.
initializeVariables()	Assign initial value for iteration, sigma0 and tau1.
modifyWeights()	Update weights of neuron.
neuron()	Constructor.
sigma()	Sigma function.

```
public Neuron(int x, int y, int sigma0, byte COLORSTEP)
        {
            color = System.Windows.Media.Color.FromRgb((byte)(255 - COLORSTEP), 255,
(byte)(255 - COLORSTEP)); //lightgreen
            coordinate.X = x;
            coordinate.Y = y;
            InitializeVariables(sigma0);
        }

public void addvector(String actor)
        {
            actorsofneuron.Add(actor);
        }

private void InitializeVariables(int sigma0)
        {
            iteration = 1;
            this.sigma0 = sigma0;
            tau1 = 1000 / Math.Log(sigma0);
        }

private double Alpha(int t)
        {
            double value = alpha0 * Math.Exp(-t/tau2);
            return value;
        }

private double Sigma(int t)
        {
            double value = sigma0 * Math.Exp(-t/tau1);
            return value;
        }

public  double  ModifyWeights(List<double>  pattern,  Point  winnerCoordinate,  int
iteration)
        {
            double avgDelta = 0;
            double modificationValue = 0;
            for (int i = 0; i < weightsdimension; i++)
            {
                modificationValue = Alpha(iteration) *
h(winnerCoordinate,Functions.Discrete) * (pattern[i] - weights[i]);
                weights[i] += modificationValue;
                avgDelta += modificationValue;
            }
            avgDelta = avgDelta / weightsdimension;
            return avgDelta;
        }
```

Implementing Code of Neuron Class Functions

```
private double h(Point winnerCoordinate, Functions f)
        {
            double result = 0;
            double distance = 0;
            switch (f)
            {
                case Functions.Discrete:
                    {
                        distance = Math.Abs(this.Coordinate.X - winnerCoordinate.X) +
Math.Abs(this.Coordinate.Y - winnerCoordinate.Y);
                        switch ((int)distance)
                        {
                            case 0:
                                result = 1;
                                break;
                            case 1:
                                result = 0.5f; //0.5 (float)
                                break;
                            case 2:
                                result = 0.25f;
                                break;
                            case 3:
                                result = 0.125f;
                                break;
                        }
                        break;
                    }
                case Functions.Gaus:
                    {
                        distance = Math.Sqrt(Math.Pow((winnerCoordinate.X -
coordinate.X), 2) + Math.Pow((winnerCoordinate.Y - coordinate.Y), 2));
                        result = Math.Exp(-(distance * distance) /
(Math.Pow(Sigma(iteration), 2)));
                        break;
                    }
                case Functions.MexicanHat:
                    {
                        distance = Math.Sqrt(Math.Pow((winnerCoordinate.X -
coordinate.X), 2) + Math.Pow((winnerCoordinate.Y - coordinate.Y), 2));
                        result = Math.Exp(-(distance * distance) /
Math.Pow(Sigma(iteration), 2)) * (1 - (2 / Math.Pow(Sigma(iteration), 2)) * (distance
* distance));
                        break;
                    }
```

Implementing Code of Neuron Class Functions

```
                case Functions.FrenchHat:
                    {
                        int a = 2;
                        distance = Math.Abs(this.Coordinate.X - winnerCoordinate.X) +
Math.Abs(this.Coordinate.Y - winnerCoordinate.Y);
                        if (distance <= a) result = 1;
                        else
                            if (distance < a && distance <= 3 * a) result = -1 / 3;
                            else
                                if (distance > 3 * a) result = 0;
                        break;
                    }
            }
            return result;
        }
```

Implementing Code of Neuron Class Functions

Due to the fact that the demo application must be able to read training set from file, export the trained Kohonen Network to file and vice versa, import the trained network from file, in NeuralNetwork class, we can describe each element as follows:

Variables in Neural Network Class

Variable (Field)	Description
Classes	The name (email) of each actor - pattern in the training set (variable patterns).
COLORSTEP	The gaining value for changing color process in the classify procedure.
currentEpsilon	The value of epsilon at the recent epoch.
currentIteration	The value of iteration at recent.
Epsilon	The threshold value of epsilon.
Function	The function of topological neighborhood function.
inputLayerDimension	The number of dimension of inputlayer - training set.
numberOfIterations	The threshold value of iteration's number.
numberOfPatterns	The number of elements in training set.
outputLayer	The SOM - matrix of output neuron.
outputLayerDimension	The number of dimension of the SOM.

Patterns	The training set or the list of actors needed to cluster.

Properties in Neural Network Class

Property	Description
Classes	Get or set value of variable classes.
CurrentEpsilon	Get or set value of variable currentEpsilon.
InputLayerDimension	Get or set value of variable inputlayerdimension.
OutputLayer	Get or set value of variable outputlayer.
OutputLayerDimension	Get or set value of variable outputlayerdimension.
Patterns	Get or set value of variable patterns.

Methods in Neural Network Class

Method	Description
CalculateNormOfVectors()	Compute the Euclidean distance of a pair vectors: training set vector (actor) and the SOM weight.
Classify()	Classifying process procedure.
ClassifyParallel()	Classifying process procedure with parallel loop.
ColorSOFM()	Get a matrix of color which represents a color of each element in the SOM.
ColorSOFMParallel()	ColorSOFM() with parallel loop.
FindWinner()	Find the winning neuron with an input actor.
LoadExistingNetwork()	Load the trained SOM from file in order to skip the training process.
NeuralNetwork()	Constructor.
OnEndEpochEvent()	Call EndEpochEvent() function which is a delegate function and can be describe in another class.
OnEndIterationEvent()	Call EndIterationEvent() function which is a delegate

	function and can be describe in another class.
ReadDataFromFile()	Read the training set or patterns to train the SOM
StartEpoch()	An epoch of learning process procedure.
StartLearning()	Learning process procedure.

Those are the code for the functions:

```
public void Classify()
        {
           for (int i = 0; i < patterns.Count; i++)
            {
               Neuron winner=FindWinner(patterns[i]);
               if (winner.Color.R-COLORSTEP>0)
                   winner.Color =
System.Windows.Media.Color.FromRgb((byte)(winner.Color.R - COLORSTEP), winner.Color.G,
(byte)(winner.Color.B - COLORSTEP));
               else
                winner.Color = System.Windows.Media.Color.FromRgb(winner.Color.R,
(byte)(winner.Color.G - COLORSTEP), winner.Color.B);


               winner.addvector(classes[i]);
            }
        }
public void ClassifyParallel()
        {
            Parallel.For(0,patterns.Count,delegate (int i)
            {
                Neuron winner = FindWinner(patterns[i]);
                if (winner.Color.R - COLORSTEP > 0)
                    winner.Color =
System.Windows.Media.Color.FromRgb((byte)(winner.Color.R - COLORSTEP), winner.Color.G,
(byte)(winner.Color.B - COLORSTEP));
                else
                    winner.Color = System.Windows.Media.Color.FromRgb(winner.Color.R,
(byte)(winner.Color.G - COLORSTEP), winner.Color.B);
                winner.addvector(classes[i]);
            });
        }

public Color[,] ColorSOFM()
        {
            Color[,] colorMatrix = new Color[outputLayerDimension,
outputLayerDimension];
            Classify();
            for (int i = 0; i < outputLayerDimension; i++)
                for (int j = 0; j < outputLayerDimension; j++)

                    if (outputLayer[i,j].actorsofneuron.Count>0)
                        colorMatrix[i, j] = outputLayer[i, j].Color;
                    else
                    {

                        colorMatrix[i, j] = System.Windows.Media.Colors.White;
                    }
            return colorMatrix;
        }
```

Implementing Code of Neural Network Class Functions

```
public void ClassifyParallel()
        {
            Parallel.For(0,patterns.Count,delegate (int i)
            {
                Neuron winner = FindWinner(patterns[i]);
                if (winner.Color.R - COLORSTEP > 0)
                    winner.Color =
System.Windows.Media.Color.FromRgb((byte)(winner.Color.R - COLORSTEP), winner.Color.G,
(byte)(winner.Color.B - COLORSTEP));
                else
                    winner.Color = System.Windows.Media.Color.FromRgb(winner.Color.R,
(byte)(winner.Color.G - COLORSTEP), winner.Color.B);
                winner.addvector(classes[i]);
            });
        }


public Neuron FindWinner(List<double> pattern)
        {
            double D = 0;
            Neuron Winner = outputLayer[0, 0];
            double min = CalculateNormOfVectors(pattern, outputLayer[0, 0].Weights);
            for (int i = 0; i < outputLayerDimension; i++)
                for (int j = 0; j < outputLayerDimension; j++)
                {
                    D = CalculateNormOfVectors(pattern, outputLayer[i, j].Weights);
                    if (D < min)
                    {
                        min = D;
                        Winner = outputLayer[i, j];
                    }
                }
            return Winner;
        }


public NeuralNetwork(int m, int numberOfIterations, double epsilon, Functions f)
        {
            outputLayerDimension = m;
            currentIteration = 1;
            this.numberOfIterations = numberOfIterations;
            function = f;
            this.epsilon = epsilon;
            currentEpsilon = 100;
        }
```

Implementing Code of Neural Network Class Functions (cont.1)

```
public void LoadExistingNetwork(String ExistingNetworkFile)
        {
            StreamReader sr = new StreamReader(ExistingNetworkFile);
            outputLayerDimension = Int32.Parse(sr.ReadLine());
            outputLayer = new Neuron[outputLayerDimension, outputLayerDimension];
            int sigma0 = outputLayerDimension;
            for (int i=0;i<OutputLayerDimension;i++)
                for (int j = 0; j < OutputLayerDimension; j++)
                {
                    outputLayer[i, j] = new Neuron(i, j, sigma0, COLORSTEP);
                    outputLayer[i, j].Weights = new List<double>();
                    string[] weights = sr.ReadLine().Split(' ');
                    for (int k = 0; k < weights.Length; k++)
                        outputLayer[i, j].Weights.Add(Double.Parse(weights[k]));
                }
            sr.Close();
        }


private void StartEpoch(List<double> pattern) //for learning process
        {
            Neuron Winner = this.FindWinner(pattern); //find the winning neuron first.
            currentEpsilon = 0;
            for (int i = 0; i < outputLayerDimension; i++)
                for (int j = 0; j < outputLayerDimension; j++)
                {
                    currentEpsilon += outputLayer[i, j].ModifyWeights(pattern,
Winner.Coordinate, currentIteration);
                }
            currentIteration++;
            currentEpsilon = Math.Abs(currentEpsilon / (outputLayerDimension *
outputLayerDimension));
            EndEpochEventArgs e = new EndEpochEventArgs();
            OnEndEpochEvent(e);
        }


protected virtual void OnEndEpochEvent(EndEpochEventArgs e)
        {
            if (EndEpochEvent != null)
                EndEpochEvent(this, e);
        }

protected virtual void OnEndIterationEvent(EventArgs e)
        {
            if (EndIterationEvent != null)
                EndIterationEvent(this, e);
        }
```

Implementing Code of Neural Network Class Functions (cont.2)

```
public void ReadDataFromFile (string inputDataFileName,bool isNew)
        {
            StreamReader sr = new StreamReader(inputDataFileName);
            string line = sr.ReadLine();
            int k = 0;
            for (int i = 0; i < line.Length; i++)
            {
                if (line[i] == ' ') k++;
            }
            inputLayerDimension = k;
            if (isNew)
            {
                int sigma0 = outputLayerDimension;
                outputLayer = new Neuron[outputLayerDimension, outputLayerDimension];
                Random r = new Random();
                for (int i = 0; i < outputLayerDimension; i++)
                    for (int j = 0; j < outputLayerDimension; j++)
                    {
                        outputLayer[i, j] = new Neuron(i, j, sigma0, COLORSTEP);
                        outputLayer[i, j].Weights = new
List<double>(inputLayerDimension);
                        for (k = 0; k < inputLayerDimension; k++)
                        {
                            outputLayer[i, j].Weights.Add(r.NextDouble());
                        }
                    }
            }
            k = 0;
            while (line != null)
            {
                line = sr.ReadLine();
                k++;
            }
            patterns = new List<List<double>>(k);
            classes = new List<string>(k);
            numberOfPatterns = k;
            List<double> pattern;
            sr = new StreamReader(inputDataFileName);
            line = sr.ReadLine();
            while (line != null)
            {
                int startPos = 0;
                int endPos = 0;
                int j = 0;
                pattern = new List<double>(inputLayerDimension);
```

Implementing Code of Neural Network Class Functions (cont.3)

```
                for (int ind = 0; ind < line.Length; ind++)
                {
                    if (line[ind] == ' ' && j != inputLayerDimension)
                    {
                        endPos = ind;
                        pattern.Add(Convert.ToDouble(line.Substring(startPos, endPos -
startPos)));
                        startPos = ind + 1;
                        j++;
                    }
                    if (j > inputLayerDimension) throw new InvalidDataException("Wrong
file format. Check input data file, and try again");
                }
                patterns.Add(pattern);
                startPos = line.LastIndexOf(' ');

                classes.Add(line.Substring(startPos));
                line = sr.ReadLine();
            }
        }


public void ExportToFile(String FileName)
        {
            System.IO.StreamWriter writer = new System.IO.StreamWriter(FileName,
false);
            //write the number of neurons first
            writer.WriteLine(OutputLayerDimension);
            //then the weights of each output
            for (int i = 0; i < OutputLayerDimension; i++)
                for (int j = 0; j < OutputLayerDimension; j++)
                {
                    Neuron cur = OutputLayer[i, j];
                    for (int k = 0; k < cur.Weights.Count; k++)
                    {
                        writer.Write(cur.Weights[k]);
                        if (k != cur.Weights.Count - 1)
                            writer.Write(" ");
                    }
                    writer.WriteLine();
                }
            writer.Close();
        }
```

Implementing Code of Neural Network Class Functions (cont.4)

```
private double CalculateNormOfVectors(List<double> vector1, List<double> vector2)
//compute Euclidean distance
        {
            double value = 0;
            for(int i=0; i<vector1.Count; i++)
                value += Math.Pow((vector1[i] - vector2[i]), 2);
            value = Math.Sqrt(value);
            return value;
        }
```

Implementing Code of Neural Network Class Functions (cont.5)

To be more comprehensible with the input files, these are the structure of them:

In the training set input file, the structure of this text file is organized as a several of lines and each line represents a training vector with the dimensional values in order then the email. Separators between each values are white space.

This is an example of training set input file:

```
0.3 0.2 0.1 user1@test.com
0.9 0.3 0.4 user2@test.com
0.1 0.2 0.2 user3@test.com
```

In the existing Kohonen network input file, this text file is contains two part. The first part is the first line which represents the number of neural elements in width or length of the SOM. The second part contains various lines which the total number is the overall number of neural elements or equal to the square of the previous value in the first part. In the same vein with the training set input file, each line includes the dimensional values in order of the weights of each neuron and the separators in this case are also whitespace characters. This is an example:

2

0.3 0.2 0.3

0.05 0.1 0.4

0.2 0.2 0.2

0.15 0.11 0.3