A Comparison of SOM Based Document Categorization Systems

X. Luo, A. Nur Zincir-Heywood
Dalhousie University,
Faculty of Computer Science,
6050 University Avenue, Halifax, Nova Scotia. B3H 1W5

***Abstract* - This paper describes the development and evaluation of two unsupervised learning mechanisms for solving the automatic document categorization problem. Both mechanisms are based on a hierarchical structure of Self-Organizing Feature Maps. Specifically, one architecture is based on the vector space model whereas the other one is based on a code-books model. Results show that the latter architecture (90%) outperforms the first (56%) as based on the quality of the returned clusters.**

I. INTRODUCTION

One of the significant tasks in document categorization systems is clustering because it offers a solution to the problems of information overload and vocabulary differences. The term information overload is used to describe the constant influx of new information [2, 3], which causes users to be overwhelmed by the subject and system knowledge required to access this information. Thus, clustering systems are used to separate a very large set of documents into groups, where grouped documents share similar content. Hence, a natural extension of such a system would be the ability to automatically provide initial document categorization in much the same way that fingerprint categorization is used to sort prints into types before the computationally expensive process of classification takes place [4]. These functions should be performed in real-time, and therefore, are of significant use to many existing on-line applications, such as document browsing, nearest neighbor search or directory structures like Yahoo [3].

A common approach among existing systems is to cluster documents based upon their word distributions, while word clustering is determined by document co-occurrence. However, the distribution of words in most real document collections can vary significantly from one document to another. Moreover, vocabulary differences mean that automatic selection and weighing of keywords in text documents may well bias the nature of the clusters found at later stages. Thus, data representations enabling accurate measurement of semantic similarities between documents would greatly facilitate such tools in terms of computational costs as well as accuracy.

In information retrieval, a typical data representation phase uses the Vector Space Model (VSM), where the frequency of occurrence of each word in each document is recorded. These values are then generally weighted using the Term Frequency (TF) multiplied by the Inverse Document Frequency (IDF), following Shannon's information theory. Note that this approach considers only term co-occurrences in documents and therefore ignoring the significance of the order in which they occur.

Thus, this work details the construction, and testing of two unsupervised clustering systems based on Self-Organizing Feature Maps (SOM) that encompass representation of word features such as word orders as well as co-occurrences. By doing so, we aim to minimize any *a priori* assumption regarding suitable word features and to discriminate higher-level concepts. Specifically, the first clustering system built is based on the VSM, which is representative of typical information retrieval approaches, and makes use of topological ordering property of SOMs. On the other hand, the second system makes use of the SOM based architecture as an encoder for data representation - by finding a smaller set of prototypes from a large input space – without using the typical information retrieval pre-processing. Our results show that the quality of clustering of the second architecture (90%) outperforms the first one (56%).

The remainder of the paper is organized as follows. Section II introduces the problem addressed by this work and makes a case for solving this problem using an entirely unsupervised learning algorithm. Section III presents the two architectures for constructing such a system. Section IV introduces the test bed on which experiments are performed, and presents the results. Finally, conclusions are drawn in section V.

II. DOCUMENT CLUSTERING WITH SELF-ORGANIZING MAPS

In this work, the aim is to investigate the potential for automating the process of document clustering as much as possible. In order to achieve this, the first step is the identification of an encoding of the original information (category) such that pertinent features (character probabilities) may be decoded most efficiently. This information is then used to measure the similarity between the characteristics of any given document to a category. Thus, the core of the approach is to automate the identification of typical category characteristics. To this end, an unsupervised learning system – Self Organizing Feature Map (SOM) – is employed to detect and visualize the characteristics of a document category.

The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada for the second author's Discovery Grant.

The SOM algorithm is a vector quantization algorithm. Thus, the efficient update scheme and ability to express topological relationships of an SOM makes it very convenient for expressing relationships between different groups of documents [7]. These properties have made the SOM a popular approach for document clustering, visualization or analysis [2, 7]. The hypothesis motivating such a scheme is that similar document characteristics will be emphasized – densely populated regions of the SOM – whereas different characteristics will appear in sparse regions of the SOM [7].

On the other hand, in this work, SOMs are of particular interest not only for their topological ordering property, feature selection and density matching but also, on account of their ability to provide approximations of the input in a lower dimensional space [5]. In other words, the SOM acts as an encoder to represent a large input space by finding a smaller set of prototypes.

In this case, the SOM represented by the set of weight vectors $\{w_j\}$ in the output space – O – is to provide a good approximation of the original input space – I. To achieve this, the basic aim of the SOM algorithm is to store a large set of input vectors, $x \in I$, by finding a smaller set of prototypes, $w_j \in O$. Hence, the feature map can be considered as an Encoding-Decoding model, figure 1.

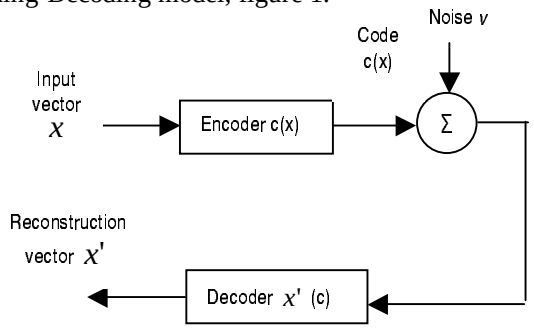

Fig.1. Encoding-Decoding Model [5]

In this model, $c(x)$ acts as an encoder of the input vector x and $x'(c)$ acts as a decoder of $c(x)$. The vector x is selected at random from a training sample, I, subject to an underlying probability density function $f_x(x)$. The optimum encoding-decoding scheme is determined by varying the functions $c(x)$ and $x'(c)$, so as to minimize the expected distortion. However, in order to overcome the local-minimum problem a signal independent noise process, v, is introduced following the encoder $c(x)$, i.e. out of category document character distributions. The noise v is associated with a fictitious communication channel between the encoder and the decoder. On the basis of this model, the best-matching neuron, $i(x)$, of the SOM algorithm corresponds to the encoder, $c(x)$; weight vector, w_j, corresponds to the reconstruction vector, $x'(c)$; and the neighborhood function, $h_{j,i(x)}$, corresponds to the probability density function, $\Pi(c-c(x))$, associated with the noise, v [5]. The details of the SOM algorithm are given in sub-section III-C.

This viewpoint forms the theoretical justification of using the SOMs as code-books, which corresponds to the motivation behind this work. As discussed in section I, the two main problems in information retrieval are information overload, i.e. large input space, and vocabulary differences, i.e. the nature of the signal. Hence, in this work, the first SOM architecture is making use of the property to map the (large) input space to a (smaller) set of prototypes independent from the signal (noise properties). On the other hand, the second architecture not only uses the above property but also aims to make use of the encoding-decoding model by building up an understanding of documents by first considering the relationships between characters, then words and finally word co-occurrences.

III. ARCHITECTURE OVERVIEW

To achieve the aforementioned objectives, the framework of figure 2 is followed, for both of the SOM architectures considered.

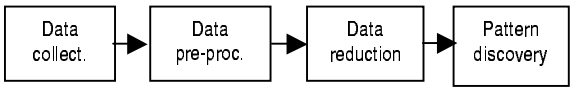

Fig.2. System flowchart

In document clustering, a typical pre-processing phase uses the Vector Space Model. There are two main parts to the vector space model: Parsing and Indexing [1, 7, 9, 10]. Parsing converts documents into a succession of words. The words are filtered using a basic Stop-List of common English words, which do not significantly contribute to discrimination between documents (such as "the", "it", etc.). A stemming algorithm is then applied to the remaining words. In the indexing part, each document is represented in the Vector Space Model where the frequency of occurrence of each word in each document is recorded. These values are then generally weighted using the Term Frequency (TF) multiplied by the Inverse Document Frequency (IDF) as per Shannon's information theory, (1). Once the set of document vectors has been created, techniques from pattern discovery are employed to form the overall classification.

$$P_{ij} = tf_{ij} \,.\, log\,(N/df_i) \qquad (1)$$

P_{ij} – Weight of term t_j in document d_i
tf_{ij} – Frequency of term t_j in document d_i
N – Total number of documents in collection
df_i – Number of documents containing term t_j

A. Architecture-1: Emphasizing Density Matching Property

In this architecture, the objective is to form illustrative 2-D projections of distributions of items in high dimensional data spaces, such as vast document collections. With classical methods these mappings are computationally expensive [16]. Thus, the SOM as a neural network method can be employed to approximate an unlimited number of input data items by a finite set of models (neurons) [11]. In this method,

relationships between documents can be viewed using spatial organizations on a 2-D grid structure.

To the best of our knowledge, the SOM method was first employed to small data sets (~100 documents) in [8, 15] and more recently to a large data set (~7 million documents) in WEBSOM project [7]. In all three cases, a vector space model is used at the pre-processing stage. Word histograms are formed as input vectors to represent documents to the SOM. Early examples had limited document vocabularies, providing for direct entry to the SOM. However, data sets met in practice require a random mapping to decouple the SOM from the size of the vocabulary space [6, 7].

In architecture-1, the classical vector space model method is utilized (1), Figure 3. Note that the random mapping method is used to reduce the dimensionality before presenting a document to the SOM.

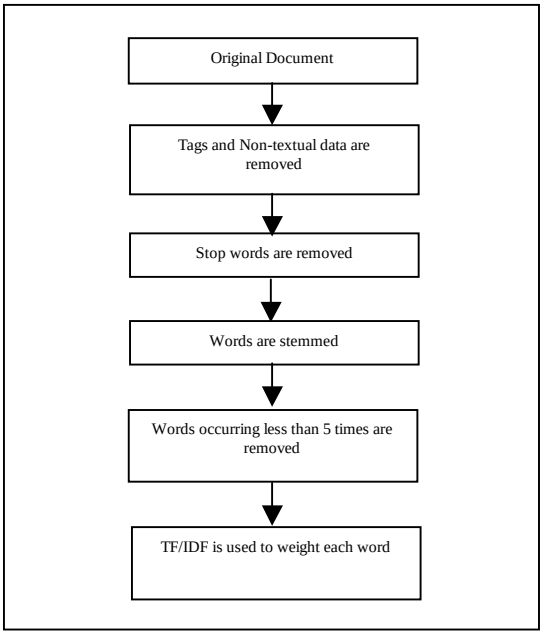

Fig.3. An overview of data pre-processing for the first architecture

In the random mapping method [6], the original data vector is reduced by replacing, each of its dimensions by a randomly selected non-orthogonal direction. The mapping [6] is as follows:

$$x' = R\,x \qquad (2)$$

x – The original data vector, where $x \in \mathbb{R}^N$

R – A matrix consisting of random values where the Euclidean length of each column has been normalized to unity

x'– Reduced-dimensional or quantized vector, where $x' \in \mathbb{R}^d$

Once the dimension of the input space is reduced, it is presented to a two level hierarchical SOM architecture. Hence, the significance of effective pre-processing and dimensionality reduction before presentation to the SOM cannot be over-emphasized. The hierarchical nature of the architecture is shown in Figure 4.

1) *Input for the First-Level SOMs:* The overall pre-processing process for the first-level SOM is:

- Generate vector space model to represent each document (1).
- Reduce the dimension of the vector space model for each document by random mapping (2).
- Present the new vector of each document to the SOM.

Thus, the first-level SOM is trained on the corresponding word list of documents. The assumption at this level is that similar document characteristics, i.e. words, will appear in the densely populated regions of the SOM for the patterns of words that occur in a specific document category.

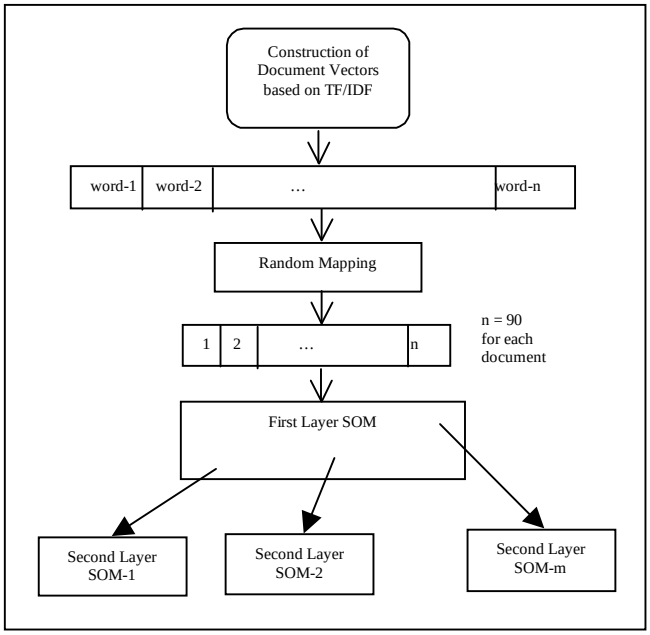

Fig.4. An overview of the first architecture

2) *Input for the Second-Level SOMs*: When a quantized vector representing a document is presented to a trained first-level SOM, the location of each neuron is expressed in terms of an Euclidean space with neuron topology roughly reflecting the density of the feature(s) in the document training set. However, due to the self-organizing characteristics of the map, some of the neurons may become more crowded with words (representing documents) as compared to others. Enlarging the size of the SOM solves the problem to some degree, but after a certain size the number of the crowded ones remains the same [4]. Moreover, as the size of a map increases, also the computational cost increases. Therefore, instead of increasing the size of the SOM within the level, associating additional SOMs with crowded neurons may solve the problem by a divide-and-conquer method [4]. Hence, each additional SOM network in the second-level is

trained only by those samples represented by the crowded neurons in the first-level. This process is applied whenever a region in the first-level SOM was excited by more than fifty documents from 2 or more categories.

B. Architecture-2: Emphasizing Encoding-Decoding Property

In this case, the core of the approach is to automate the identification of typical category characteristics, i.e. good approximation of the input space – code-book model. Steps to achieve data pre-processing and reduction are driven by the needs of the pattern discovery component. As mentioned before, pattern discovery employs a three level hierarchical SOM architecture (characters, words, and word co-occurrences).

In this architecture, the classical parsing method (section III) is applied to remove stop words etc., after which, a different indexing scheme is used as shown in Figure 5. In this scheme, a document is summarized by its words and their frequencies (TF) in descending order, but the information theory framework is *not* used, (1). Once such a list is formed, the 15 most frequent words are retained from each document. Moreover, these words are pre-processed to provide a numerical representation for each character. An SOM may now be used to identify a suitable character encoding, then word encoding, and finally, word co-occurrence encoding. By doing so, the authors of this paper aim to minimize the amount of *a priori* knowledge required to overcome the vocabulary differences. The hierarchical nature of the architecture is shown in Figure 6.

1) *Input for the First-Level SOMs:* In order to train an SOM to recognize patterns in characters, the document data must be formatted in such a way as to distinguish characters and highlight the relationships between them. Characters can easily be represented by their ASCII representations. However, for simplicity, we enumerated them by the numbers 1 to 26, i.e. no differentiation between upper and lower case. The relationships between characters are represented by a character's position, or time index, in a word. For example, in the word *"news"*: *"n"* appears at time index 1, *"e"* appears at time index 2, "*w*" appears at time index 3, and *"s"* appears at time index 4. It should be noted that it is important to repeat these words as many times as they occur in the documents. In other words, for a particular document, the 15 most frequently occurring words may occur a combined total of 50 times. Therefore, a list of 50 words is formed with the words remaining in the same order as they appear in the document. The overall pre-processing process for the first-level SOM is therefore:

- Convert the word's characters to numerical representations between 1 and 26.
- Find the time indices of the characters.
- Linearly normalize the indices so that the first character in a word is represented by one, and the last by 26. Those in between are equally spaced between 1 and 26.

The indices of the characters are altered in this way so that when the list is input to an SOM, both data features (enumerated characters and indices) are spread out over a close range. The assumption at this level is that the SOM forms a code-book for the patterns in characters that occur in a specific document category.

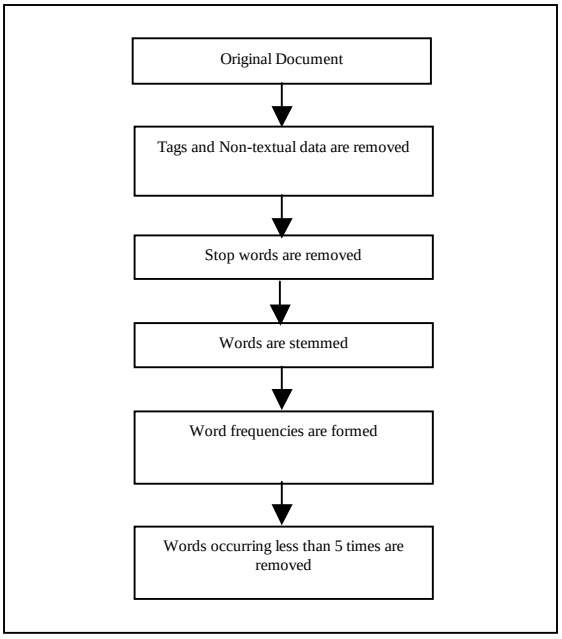

Fig.5. An overview of data pre-processing for the proposed approach

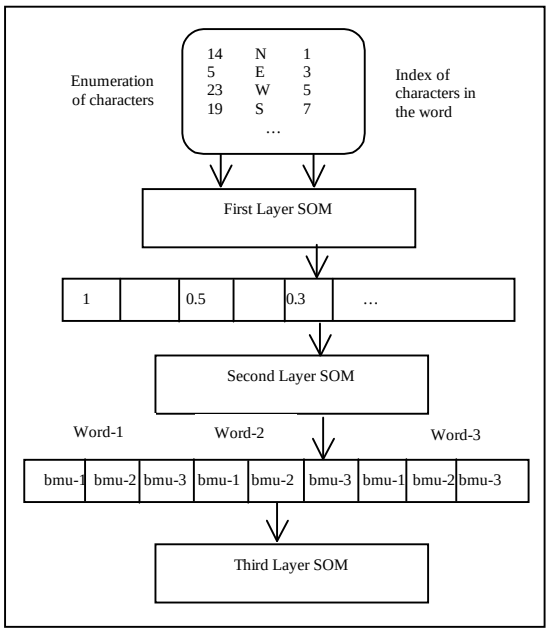

Fig.6. An overview of the second architecture

2) *Input for the Second-Level SOMs*: When a character and its index are run through a trained first-level SOM, the closest neurons (in the Euclidian sense), or *Best Matching Units* (BMUs), are used to represent the input space. The following inter-stage processing is therefore used for defining word level inputs to the second-level SOMs:

- For each word, *k*, that is input to the first-level SOM of each document,
  - Form a vector of size equal to the number of neurons (*r*) in the first-level SOM
  - For each character of *k*,
    - Observe which neurons n_1, n_2, ..., n_r are affected the most (the first *3* BMUs).
    - Increment entries in the vector corresponding to the first 3 BMUs by 1/j, $1 \leq j \leq 3$.

Hence, each vector represents a word through the sum of its characters.

3) *Input for the Third-Level SOMs*: In the context of this architecture, word co-occurrence is simply a group of consecutive words in a document. Thus, the third-level SOMs are used to identify such patterns in the input space. The input space of the third-level SOMs is formed in a similar manner to that in the second-level, except that the third-level input vectors are built using BMUs resulting from word vectors passed through the second-level SOMs. Furthermore, in order to form a more meaningful input space for the third-level SOMs, it should be noted that the input vectors represent consecutive words only from a single document with a sliding window of size three.

C. Training the SOMs

The algorithm responsible for the formation of the SOM involves three basic steps after initialization: sampling, similarity matching, and updating. These three steps are repeated until formation of the feature map has completed [5]. The algorithm is summarized as follows:

- Initialization: Choose random values for the initial weight vectors $w_j(0)$, $j=1,2,\ldots,l$, where l is the number of neurons in the map.
- Sampling: Draw a sample x from the input space with a uniform probability.
- Similarity Matching: Find the best matching neuron $i(x)$ at time step n by using the minimum distance Euclidean criterion:

$$i(x) = \arg\min_j \|x(n) - w_j\|, \qquad j=1,2,\ldots,l \qquad (3)$$

- Updating: Adjust the weight vectors of all neurons by using the update formula:

$$w_j(n+1) = w_j(n) + \eta(n) h_{j,i(x)}(n)(x(n) - w_j(n)) \qquad (4)$$

- Continuation: Continue with sampling until no noticeable changes in the feature map are observed.

The sizes of the maps shown in table 1 are chosen empirically according to the observed weight changes. Hence, we considered the balance between the computational cost and the weight change in choosing the size of a map.

TABLE I
SIZES AND TRAINING TIME FOR THE MAPS

	Size of the map	Training time
Architecture-1		
Level-1	25 by 20	~38 hours
Level-2	6 by 6 – 4 by 4	~5 - ~2 minutes
Architecture-2		
Level-1	7 by13	~ 4 hours
Level-2	8 by 8	~ 19 hours
Level-3	20 by 20	~ 25 hours

IV. PERFORMANCE EVALUATION

In this work, a subset of the Reuters-21578 document set [13] is used to evaluate the two architectures. There are a total of 12612 news stories in this collection. These stories are in English, where 9603 of them are in the training data set, and 3009 are in the test set. Moreover, among the training documents, which are labeled, approximately 333 of them have more than 3 labels, and only 25 categories have more than 50 documents. On the other hand, among the test documents, only 2733 of them are labeled, and in that set, 2526 documents belong to the 25 different categories. Hence, we selected these test cases to use for the experiments described here. Thus, our training set consists of 9603 documents, whereas the test set has 2526 documents.

Most often in information retrieval, clusters are embedded in an application and therefore evaluation takes the form of a comparison between a 'winning' labeled cluster and the class label. However, such approaches measure only the quality of the best cluster. On the other hand, the documents of the above data set belong to more than one class; hence there is no single *best cluster*, this is a multi-class multi-labeled problem. Therefore, an evaluation methodology [12] that keeps a balance between generality and goal-orientation is used to compare the performance of the two architectures. The performance measurement used is based on:

$$1 - \frac{dist(C, A)}{dist(B, A)} \qquad (5)$$

A– set of correct class labels (answer key)
B – baseline clusters where each document is one cluster
C – set of clusters ('winning' clustering results)
$dist(C, A)$ – The number of operations required to transform C into A.
$dist(B, A)$ – The number of operations required to transform B into A.

In this evaluation method, it is assumed that there are correct class labels (answer keys) that define how the elements (documents) are supposed to be clustered.

Moreover, three *operations* are allowed to support construction of answer keys from 'winning' clusters.

- Merge two clusters.
- Move an element from one cluster to another
- Copy an element from one cluster to another

Hence, the performance or quality of clustering is interpreted as the percentage of savings (in terms of *operations*) from using the 'winning' clustering results to construct the answer key versus using baseline clusters to construct the answer key. In this case, we applied it as follows:

1. If a document is in a class, which is not listed in the document's label set, then we move the document to the class it belongs.
2. If a document belongs to more than one target class, then we copy the document to the remaining classes listed in its label set.

The similarity between documents and the class is computed using the cosine coefficient [14] of their outputs on the corresponding architectures. Table 2 shows the performance scores (using function 5) of both architectures for three different threshold values for their cosine coefficient functions.

TABLE II
CLUSTER QUALITY OF THE TWO ARCHITECTURES

	ARCHITECTURE-1	ARCHITECTURE-2
Threshold=0.01	56.3%	90.5%
Threshold=0.05	10.3%	62.6%
Threshold=0.1	6.7%	35.2%

V. CONCLUSION

Two systems have been developed for unsupervised clustering of document collections that encompasses pre-processing for word features as well as word co-occurrences. By doing so, the authors aim to minimize any *a priori* assumptions regarding suitable word features as well as to discriminate between higher level concepts.

Unlike earlier approaches, both systems emphasize the use of SOMs on account of their ability to provide approximations of the input in a lower dimensional space. The first architecture emphasizes a layered approach to lower the computational cost of the training of the map and employs a random mapping to decrease the dimension of the input space. Although, these techniques have been used separately in previous works [4, 7], using them together on a well know benchmarking data set (Reuter's data set) has not been previously reported.

On the other hand, the second architecture is based on a new idea where the SOM acts as an encoder to represent a large input space by finding a smaller set of prototypes. In contrast to previous methods, such a scheme significantly reduces *a priori* assumptions regarding suitable word features. Hence, code-books for automatic clustering are formed by building up an understanding of documents by first considering the relationships between characters (the first-level SOMs), then words (the second-level SOMs), and finally, word co-occurrences (the third-level SOMs).

Training and testing results for 25 document categories (as described in section 4) of the Reuters data set to utilize the above two systems have been very promising. Architecture-2 outperformed architecture-1 on this data set based on the editing distance between output clusters and the correct labels of the data set. Future work will develop a classifier, which will work in conjunction to these clustering systems and apply the technique to a wider cross-section of benchmark data sets. In addition, we are also interested in the utilization of this system to perform document categorization and as an indexing mechanism for a document database so that it does not need retraining for each new document category added.

REFERENCES

[1] Boone G., Concept features in Re:Agent, and intelligent e-mail agent, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents '98)*, pages 141–148, New York, 9–13, 1998. ACM Press.

[2] Chen H., Schuffels C., Orwig R., Internet categorization and search: A self-organizing approach, *Journal of Visual Communication and Image Representation*, Vol. 7, No.1, pp. 88-102, March 1996.

[3] Freeman R., Yin H., Allinson N. M., Self-organizing maps for tree view based hierarchical document clustering, *Proceedings of the IEEE 2002 International Joint Conference on Neural Networks*.

[4] Halici, U., Ongun, G., Fingerprint classification through self-organizing feature maps modified to treat uncertainties, *Proceedings of the IEEE*, vol.84, no.10, 1996, pp. 1497-1512.

[5] Haykin S., *Neural Networks - A comprehensive foundation*, Chapter-9: Self-organizing maps, Second Edition, Prentice Hall, 1999, ISBN 0-13-273350-1.

[6] Kaski, S., Dimensionality reduction by random mapping: Fast similarity computation for clustering, *Proceedings of the IJCNN'98 Int. Joint Conf. Neural Networks*, 1998, pp. 413-418.

[7] Kohonen T., Kaski S., Lagus K., Salojrvi J., Honkela J., Paatero V., Saarela A., Self organization of a massive document collection, *IEEE Transactions on Neural Networks*, Vol.11, No.3., pp. 574-585, May 2000.

[8] Lin X., Soergel D., Marchionini G., A self-organizing semantic map for information retrieval, *Proceedings of the 14th Annu. Int. ACM/SIGIR Conf. Research and Development in Information Retrieval*, 1991, pp. 262-269.

[9] Merkl D., Exploration of document collections with self-organizing maps: A novel approach to similarity representation, *Principles of Data Mining and Knowledge Discovery*, pages 101–111, 1997.

[10] Merkl D., Lessons learned in text document classification, *Proceedings of WSOM'97, Workshop on Self-Organizing Maps*, 1997.

[11] Pal, S.K., Talwar V., Mitra, P., Web mining in soft computing framework: Relevance, state of the art and future directions, *IEEE Transactions on Neural Networks*, vol.13, no.5, 2002, pp. 1163-1177.

[12] P. Patrick, D. Lin, Document Clustering with Committees, Proceedings of SIGIR'02, pp.199-206

[13] Reuters data set, http://www.daviddlewis.com/resources/testcollections/reuters21578/

[14] Salton G., McGill M. J., Introduction to Modern Information Retrieval, McGraw Hill, 1983.

[15] Scholtes, J.C., Unsupervised learning and the information retrieval problem, *Proceedings of the IJCNN'91 Int. Joint Conf. Neural Networks*, vol.I, 1991, pp. 95-100.

[16] Sebastiani, F., Machine learning in automated text categorization, *ACM Computing Surveys*, vol.34, no.1, 2002, pp. 1-47.

[17] Vesanto J., Himberg J., Alhoniemi E., Parhankangas J., Self-organizing map in Matlab: The SOM toolbox, *Proceedings of the Matlab DSP Conference*, pages 35–40, 1999.