Expert Systems with Applications 37 (2010) 2874–2881

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Content-based hierarchical document organization using multi-layer hybrid network and tree-structured features

M.K.M. Rahman [1], Tommy W.S. Chow *

Dept. of Electronic Engineering, City University of Hong Kong, G6409, Tat Che Avenue, Kowloon Tong, Hong Kong

ARTICLE INFO

Keywords:
Document classification
Hierarchical organization
Tree-structured features
Self-organizing map
Multi-layer hybrid network

ABSTRACT

Automatic organizing documents through a hierarchical tree is demanding in many real applications. In this work, we focus on the problem of content-based document organization through a hierarchical tree which can be viewed as a classification problem. We proposed a new document representation to enhance the classification accuracy. We developed a new hybrid neural network model to handle the new document representation. In our document representation, a document is represented by a tree-structure that has a superior capability of encoding document characteristics. Compared to traditional feature representation that encodes only global characteristics of a document, the proposed approach can encode both global and local characteristics of a document through a hierarchical tree. Unlike traditional representation, the tree representation reflects the spatial organizations of words through pages and paragraphs of a document that help to encode better semantics of a document. Processing hierarchical tree is another challenging task in terms of computational complexity. We developed a hybrid neural network model, composed of SOM and MLP, for this task. Experimental results corroborate that our approach is efficient and effective in registering documents into organized tree compared with other approach.

1. Introduction

In the current information technology world, we rely vastly on electronic documents. The way that we search for literature has changed immensely from the traditional way. A large part of books, journals, news articles, etc. that are on printed paper, have been converted to electronic version. Books and documents of a library are usually indexed in electronically using short descriptions such as 'abstract' and 'content', so that users can browse online before inspecting physically or to decide which book to pick. Massive growth of the Internet makes the access to these electronic documents easy. A large community of users requires an easy and effective access to various domains of electronic documents all over the world.

Documents can be organized in a flat structure, i.e., in a number of categories. For example, a system called STRETCH (Appiani et al., 2001) was developed for high-volume document processing where documents are read from scanned images, classified to their classes and indexed for further retrieval operation. However, it is deemed essential to organize the document database in a more efficient way to facilitate effective browsing and retrieval from document database. Organizing documents in hierarchical tree is such an approach. For example, a library database has main categories of Electronic Engineering, Law, etc. Each category has more sub-categories of its own, and so on. Thus, organizing documents through such hierarchical layout can help users to easily find his documents of relevant area. School libraries, online journals and many others web-based libraries are benefited by such approach of organizing documents. However, most of these implantations are done by manually categorizing each document into a specific node of the tree. Automation of this job can save a lot of human labor. A number of related applications can be found in Dumais and Chen (2000), Rauber, Dittenbach, and Merkl (2000), Merkl, He, Dittenbach, and Rauber (2003) and Freeman and Yin (2005).

We call the tree '*database-tree*' that organizes the documents into hierarchical structure. To associate a given document to the nodes of a database-tree, there exists two major types of classifier models, namely flat and hierarchical (Fall & Benzineb, 2002). First one is the flat model that employs the traditional classifiers (Fuhr, Hartmanna, Lustig, Schwantner, & Tzeras, 1991; Joachims, 1998; Schütze, Hull, & Pedersen, 1995; Yang, 1994). In flat-model approach, each leaf node of the database-tree is treated as independent class. Association of a given document take place in bottom-up manner, i.e., the document is firstly classified to a given

* Corresponding author. Tel.: +852 2788 7756; fax: +852 2788 7791.
E-mail addresses: masuk@eee.uic.ac.bd (M.K.M. Rahman), eetchow@cityu.edu.hk (T.W.S. Chow).
[1] Address: Dept. of Electrical and Electronic Engineering, United International University, Dhaka, Bangladesh. Tel.: +8801916201234.

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.09.002

leaf node, and then the document is further associated to the parents of the leaf node. The second one is the hierarchical model (Dumais & Chen, 2000; Koller & Sahami, 1997; Weigend, Wiener, & Pedersen, 1999) where classification take place in a top-down manner through the database-tree, i.e., the document is firstly classified into any of the first level nodes, and the process further goes down through the database-tree until a leaf node is reached. This approach involves more complexity because the system is composed a number of classification modules at different levels of the database-tree. All of the above mentioned models differ from each other by classifying algorithm, but they rely on same basic features of word frequency information which is discussed detailed as follows.

Feature representation plays a vital role on the classification and organization of document. Vector space model (VSM) has been most popular and widely used model for document-feature representation. In the VSM model, the frequency of each term of a *vocabulary* is counted for a given document, where *vocabulary* is a list of terms/words used for feature description. A term weight vector is then constructed for a document using this "term-frequency" together with "document-frequency", where document-frequency is the number of documents where the term appears. Similarity between two documents is then carried out using 'cosine' similarity measure or any other distance function (Zobel & Moffat, 1998). For the VSM model, a lengthy vector is required for describing the frequency information of terms, because the number of words involved is usually huge. This causes a significant increase in computational burden making the VSM model impractical for handling large database. To overcome this limitation, latent semantic indexing (LSI) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) was developed to compress the feature vector of the VSM model into low dimension. In addition to feature compression, the LSI model has been shown to be a useful method in encoding the semantics of a document (Berry, Dumais, & O'Brien, 1995; Papadimitriou, Raghavan, Tamaki, & Vempala, 2000). Beside the LSI model, self-organizing map (SOM) was employed for document-feature projection and dimensionality reduction (Ampazis & Perantonis, 2004; Honkela, Kaski, Lagus, & Kohonen, 1997). In Zhao and Grosky (2002), LSI was used to encode the semantic concept to some extent. In spite of the simplicity of the above conventional models, they are unable to provide a detail document description. Also, it has never been easy to develop an effective document model to coop with the immense growth if document collections in various domains. There is a strong need of developing more accurate feature representation that can improve the classification accuracy as well as scale up the classifier system to deal with huge database.

In the above described models, they all use *flat feature* representation that is a function of term-frequency of a document. These flat features are a crude representation of a document. For example, two documents containing similar term-frequencies may be of different contextually when the spatial distribution of terms are very different, i.e., *school, computer*, and *science* means very different when they appear in different parts of a page compared to *school of computer science* that appear together. Thus it is important to account contextual similarity based on the ways how words are used throughout the document, i.e., sections, and paragraphs. But, until now, most document models incorporate only term-frequency and do not include such spatial information of words. As a result, classification accuracy can drops in tacking bigger database and larger sizes of documents.

Self-organizing map (SOM) is a versatile unsupervised neural network used for generating a topologically ordered map and clustering. It is able to organize documents in a topologically ordered map, facilitating users to find similar documents that are close to each other on the SOM map. Other than feature projection SOM can be used to perform classification, clustering and retrieval of document (Ampazis & Perantonis, 2004; Freeman & Yin, 2005; Honkela et al., 1997; Merkl et al., 2003; Rahman, Yang, Chow, & Wu, 2007). An application of SOM for hierarchical document organization and browsing can be found in (Freeman & Yin, 2005; Merkl et al., 2003). The method in (Freeman & Yin, 2005), called Topological Organization of Content (TOC), hierarchically spans a set of one-dimensional growing SOMs. This can be seen as hierarchical clustering, while maintaining the topology. The method is computationally effective and scalable for large dataset. However, all these approaches are still dependent upon typical term frequency information, and are unable to describe a document in details. In order to efficiently encode a document in a better way, it is necessary to use structured representation of document contents. Tree-structured representation is a powerful approach that has been successfully applied to a number of applications of pattern recognition (Dumais & Chen, 2000; Hagenbuchner, 2002; Salembier & Garrido, 2000). In the domain of tree-structured representation, applications of neural network have been proposed using supervised models (Cho, Chi, Siu, & Tsoi, 2003; Cho & Chi, 2005) and unsupervised models (Hagenbuchner, 2002; Hagenbuchner & Tsoi, 2003; Rahman et al., 2007; Wang, Hagenbuchner, Tsoi, Cho, & Chi, 2002). While supervised models are usually applicable to a classification task, unsupervised models exhibit good potential for performing image and document retrieval. In Rahman et al. (2007), we developed a new SOM model, called MLSOM, for tree-structured data that was successfully applied to image retrieval application (Chow, Rahman, & Wu, 2006) and classification problem in the domain of document and image (Rahman et al., 2007).

In this paper, we propose a tree-structured document representation together with a hybrid neural network to organize the documents into a hierarchical database-tree. The tree-structured representation helps to encode superior description of the document contents compared with the tradition flat features. In the proposed approach a document is partition into pages and pages are portioned into paragraphs; forming a hierarchical tree 'document → pages → paragraph'. Using such three-level tree representation we are able to encode spatial distribution of words throughout the pages and paragraphs. Such spatial information, which is not present in flat feature, is necessary in understanding the underlying semantics of a document. Thus tree-structured data can encode richer information resulting in enhanced classification accuracy. To handle tree-structured data, we employed an extended MLSOM model (Rahman et al., 2007) called multi-layer hybrid network (MLHN). Basically MLHN is build by transforming MLSOM into a supervised model where we include MLP on its top layer. In short, to handle three-level document tree data, our MLHN architecture consists of three layers: two SOM layers and an MLP layer. The two SOM layers help to compress the feature space of the document tree into a fixed length vector and thus facilitate the classification job for MLP. The use of tree-structured representation and MLHN architecture shows promising results compared with other conventional approaches.

The rest of this paper is organized as follows. Section 2 presents the details of document feature representation and feature extraction procedures. Section 3 provides the details of the MLHN. Experimental results are presented and analyzed in Section 4. The conclusion is finally drawn in Section 5.

2. Tree-structured document contents

We propose a hierarchical tree-structured representation of documents that consist of text content only. To extract the tree-structure, a document is partitioned into pages that are further partitioned into paragraphs. We have developed a Java code to

perform such segmentation, and have considered only 'html' documents at this stage. In 'html' format document, paragraphs can be easily identified using html tags. To begin the process, a document is firstly segmented into a number of paragraphs. The paragraphs are then merged into pages using a minimum threshold value for the number of words of a page. It is noted that any text appearing within the html tags, which is used for formatting, are not accounted for word-count or document feature extraction.

The document partitioning process can be summarized as follows:

1. Partition the document into paragraphs blocks using the html paragraph tag "<p>" and new line tag "
".
2. Merge the subsequent paragraph blocks to form a new page until the total number of words of the merged blocks exceeds a page threshold value 1000. There is no minimum threshold for the last page. The page blocks are formed.
3. Now each page is split into a smaller blocks using more html tags: "<p>", "
", "<il>", "<td>", etc. Merge these subsequent blocks in the same fashion of Step 2 to form a new paragraph until total number of words of the merged blocks exceeds a page threshold value 100. The minimum threshold for the last paragraph of a page is kept 40; otherwise the paragraph is merged with the previous paragraph.

For html documents, there is neither any definite page boundary/break, nor any rule for minimum/maximum number of words for paragraphs/pages. But the use of a threshold of word-count still enables us to form a hierarchical structure containing the characteristics from global to local. Thus, the document contents are structured in a 'document → pages → paragraph' tree. This is a simple way of generating a tree-structure, and it can be further improved by using a form of 'document → sections → paragraph'. But it demands more complex algorithm for partitioning a document.

Fig. 1 illustrates the tree-structured representation of a document. The root node at the first level represents the whole document, the second level nodes represent different pages, and the third level nodes represent the paragraphs of the pages. Thus, a three-level tree hierarchically represents the document content. Nodes contain compressed features describing frequency distribution of different words. It should be noted that nodes at different levels contains the same word-frequency features, but they are extracted from different contents of the document. Two documents having similar word-histograms at root nodes can be completely different in terms of semantics/context, because different spatial distributions of the same set of words can result in different meanings/context. This is what is reflected by the discriminative lower parts of the tree data (second/third level nodes). Thus, tree-structured features can provide better analysis of documents.

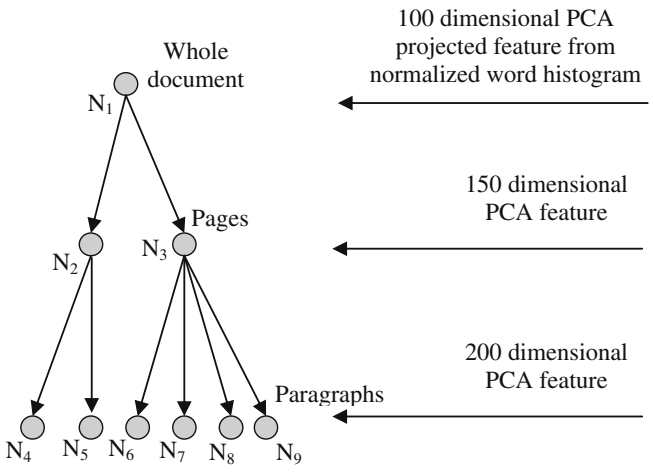

Fig. 1. Tree-structured feature representation of a document.

To extract the features, word histograms are computed for the nodes at different levels. We then apply PCA, a well known tool to project higher dimensional data into lower dimensional feature without losing much statistical information, to the word histogram vector. The overall procedure of extracting tree-structured feature can be summarized as follows:

(1) Extract words from all the documents in a database and apply stemming to each word. Porter stemming algorithm (Porter, 1980) is used to extract stem of each word, and stems are used for feature extraction instead of original words. In this way, 'work', 'works' and 'worked' are all considered being the same word. Remove the Stop words (set of common words like "the", "is", etc.). Store the stemmed words together with the information of term-frequency f_T (the frequency of a word in all documents) and the document-frequency f_d (the number of documents where a word appeared).

(2) *Vocabulary construction*: Using term-frequency f_T and document-frequency f_d information, calculate the weight of each word, which is very similar to the term-weighting in the vector space model (Salton & Buckley, 1996).

$$W_{term} = idf \times \sqrt{f_T}, \tag{1}$$

where the inverse-document-frequency $idf = \log_2\left(\frac{N}{f_d}\right)$, and N is the total number of documents in the whole database. We sort the words by using weight value in descending order and select the first N_s words. Alternatively, one can select words having a weight value above a threshold. The selected words construct the *vocabulary*. The choice of N_s or selection depends on the database.

(3) Partition the document into pages, and pages into paragraphs. Construct the document tree. The root node contains the histogram of a whole document, the second level nodes are used for pages, and the third level nodes are used for paragraphs.

(4) Calculate word histograms for documents, pages and paragraphs that represent the features of nodes. Each element of the histogram indicates the number of times the corresponding word appears in a document, a page, or a paragraph. Finally, we normalize the histogram using the follows:

$$H = [h_1\ h_2\ h_3 \ldots h_T], \quad h_t = \frac{Tn_t}{\sum_{t=1\ldots T} n_t}, \tag{2}$$

where T is the total number of words in the constructed *vocabulary*, and n_t is the frequency of tth word in the *vocabulary*.

(5) Use the normalized histogram to construct the PCA projection matrix B. To save the computational burden, we construct the matrix B only at the root nodes. We have used the MATLAB tool (Rasmussen, 2005) to compute the PCA projection matrix.

(6) Project the node features (normalized histogram) into the lower dimensional PCA feature by using PCA projection matrix. Using the projection matrix, the PCA features are computed.

$$F_h = H^* B, \tag{3}$$

where B is the projection matrix of dimension $T \times n_F$, and n_F is the dimension of the projected feature. Details of the PCA

projection can be referred to (Rasmussen, 2005). The projected features in F_h are ordered according to their statistical importance. In our application, n_F was set to 200, but we further reduced the number of features used for the root node and second level nodes. Thus, the new dimensions of the PCA feature for the first, second and third level nodes are considered to be 100, 150 and 200, respectively.

(7) Save the *vocabulary* base and the projection matrix that are later used to make the features of a new query document. The tree structured features of a query document are extracted in the same way but excluding Steps 1, 2 and 5 because these are required to compute only once over the database.

3. Network architecture and training

We developed a hybrid network model called MLHN by combining MLP and SOM. MLP has been a very popular supervised model in classifying patterns. However traditional MLP only supports flat feature representation of data. The SOM is well-known unsupervised that possess a number of abilities such as data clustering, topological ordering and dimensionality reduction. These abilities helped us to develop MLSOM (Rahman et al., 2007) to handle tree-structured data, where nodes at different levels of the tree are processed within different layers of SOM. In MLSOM, the top layer organizes all the tree data, whereas other layers help to compress huge dimensional features of a tree data. In our proposed MLHN architecture, we replace the top SOM layer of MLSOM with an MLP. Thus, the MLP serves the basic classification task, whereas other SOM layers help to encode the large dimension tree features into a short and fixed vector representation. After introducing the basic mechanism of SOM, the details of the MLHN will be presented.

3.1. Self-organizing map

A basic SOM consists of M neurons located on a regular low dimensional grid that is usually in two-dimensional. The lattice of a two-dimensional grid is either hexagonal or rectangular. Each neuron i has a d-dimensional feature vector $w_i = [w_{i1}, \ldots, w_{id}]$. The SOM algorithm is iterative. During the training process the neurons are updated in a way that their feature vectors finally become representative of different data clusters in an orderly fashion. The iterative SOM training algorithm can be stated as follows:

Step 1. Set iteration $t = 0$.
Step 2. Randomly select a sample data vector $x(t)$ from a training set.
Step 3. Compute the distances between $x(t)$ and all feature vectors. The winning neuron, denoted by c, is the neuron with the feature vector closest to $x(t)$

$$c = \arg\max_i(S(x(t), w_i)), \quad i \in \{1, \ldots, M\}. \tag{4}$$

S is a similarity function to compute similarity between $x(t)$ and w_i.
Step 4. Update the winner neuron and its neighbor neurons. The weight-updating rule in the sequential SOM algorithm can be written as

$$w_i(t+1) = \begin{cases} w_i(t) + \eta(t)h_{ic}(t)(x(t) - w_i(t)), & \forall i \in N_c \\ w_i(t), & \text{otherwise} \end{cases}, \tag{5}$$

$\eta(t)$is the learning rate which decreases monotonically with iteration t.

$$\eta(t) = \eta_0 \cdot \exp\left(-\alpha \cdot \frac{t}{\tau}\right), \tag{6}$$

where $\eta_0$is the initial learning rate, and α is an exponential decay constant set to 3 throughout our experiments. N_c is a set of neighboring neurons of the winning neuron, and $h_{ic}(t)$ is the neighborhood kernel function that defines the closeness of a neighborhood neuron to the winning neuron c at position (x_c, y_c). The neighborhood function $h_{ic}(t)$ also decreases gradually during the learning process.

$$h_{ic}(t) = \exp\left(-\frac{[(x_i - x_c)^2 + (y_i - y_c)^2]}{2\sigma^2(t)}\right), \tag{7}$$

where $\sigma(t)$ is the width of the neighborhood function that decreases with iteration

$$\sigma(t) = \sigma_0 \cdot \exp\left(-\frac{t}{\tau} \cdot \log\sigma_0\right), \tag{8}$$

where σ_0 is the initial width, τ is a time constant set to the maximum number of iterations.
Step 5. Stop when the maximum iteration is reached, or set $t = t + 1$ and go to Step 2.

3.2. Multi-layer hybrid network (MLHN)

In general there are as many layers in MLHN as the number of levels in the tree data. The top layer is an MLP network and others are SOM. In this application, MLHN consists of three layers to deal with three-level tree of document. Nodes at different level are processed in level-by-level and bottom up fashion; i.e., third level nodes are processed in layer-3 SOM, then second level nodes are processed, and finally root nodes are processed in MLP. In MLHN architecture, input representation for a particular layer is dependant on its immediate lower layer. This is because a node contains features as well as child nodes' information. The child nodes' compressed information is only available after they are processed in the lower layer. This is illustrated in Fig. 2 where a three-level tree data is processed within a three-layer MLHN. First, the third level nodes, which are features of different paragraphs, are mapped to the layer-3 SOM output. The winner neurons of different nodes or paragraphs (B, C and D) are represented by their corresponding position vectors (p_B, p_C and p_D). Thus, the input vector at layer-2 is made up with the second level node (page) features and the position vectors from layer-3. However, a mapping procedure, which is described later in this section, is essential in combining position vector with the node's features. Using the mapping, the position vectors are mapped from the outputs of the layer-3 SOM into a structured format forming the input vector of the layer-2. In this way, we can form a compressed and fixed-dimensional representation of child nodes. Using the layer-2 input vectors, winner neurons correspond to the second level nodes are found on the layer-2 SOM. Thus, layer-1 inputs are made up with root node's feature and position vectors from layer-2. At last, layer-1 input corresponds to root node is feed to the MLP for classification of the tree datum. It should be noted that the third and second layer SOMs serve as a feature compression that compress the paragraphs and pages information, respectively. In this way, the whole tree can be compactly represented by the root node's input at layer-1.

3.2.1. Data representation and similarity function

In the tree data, a node X^k at the kth level is represented by $X^k = [f_{1x}, f_{2x}, \ldots, f_{mx}, p_{1x}, p_{2x}, \ldots, p_{c_{\max}x}]^T$, where f_i represents the ith feature of node X^k, $p_{ix}(= [x_i, y_i], x_i \in [0,1], y_i \in [0,1])$ is a normalized two-dimensional position vector of a neuron that compactly represents ith child node, and $c_{\max}$ is the maximum number of child nodes of the nodes at the kth level. The position vectors are mapped in the vector $[p_{1x}, p_{2x}, \ldots, p_{c_{\max}x}]$ according to spatial posi-

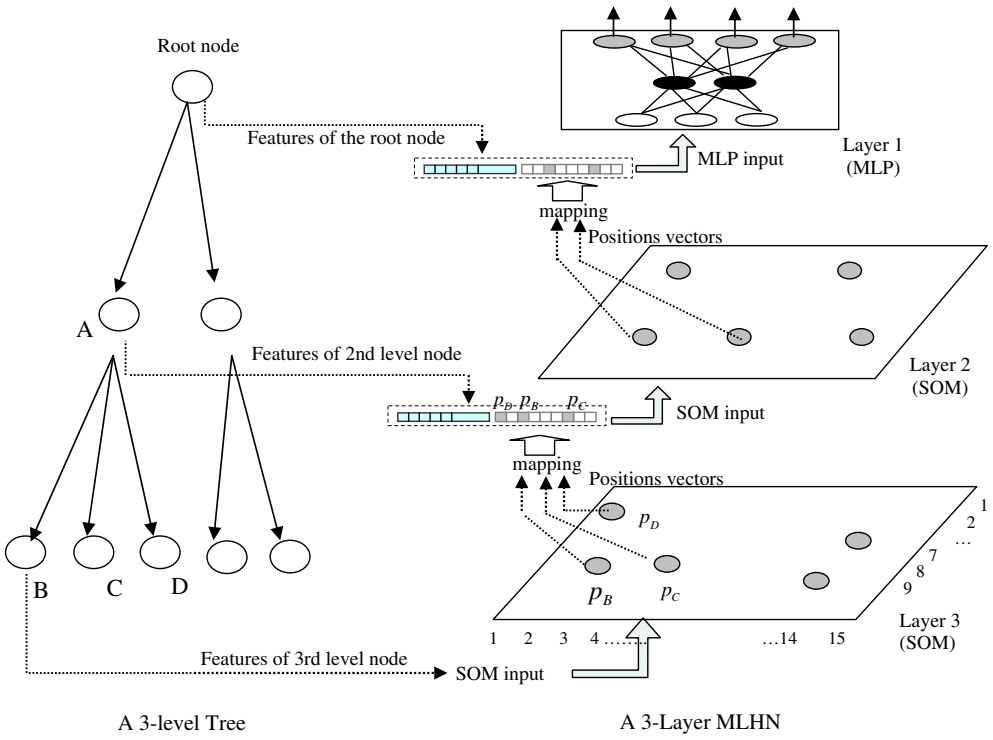

Fig. 2. Illustration of the mapping of a three-level tree-structured data into a three-layer MLHN.

tion that will be discussed later in this section. A node may have less than $c_{\max}$ number of child node, and some of p_{ix} may contain zero vector [0, 0]. The weight vector of a neuron at the third layer is represented by $W^3 = [f_{1w}, f_{2w}, \ldots, f_{m'w}]^T$, where f_{iw} is the ith weight. The following similarity function is used to find winner neuron for a given third level node.

$$S(X,W) = \sum_{i=1}^{m'} \{1 - |f_{ix} - f_{iw}|\}. \tag{9}$$

The weight vector of a neuron at the second layer is represented by $W^2 = [f_{1w}, f_{2w}, \ldots, f_{mw}, \quad p_{1w}, p_{2w}, \ldots, p_{c_{\max}w}]^T$, where $p_{ix}(= [x_i, y_i])$ is a two-dimensional vector. The following function computes the similarity between a second level node and a neuron to find the winner neuron.

$$S(X,W) = \frac{1}{m}\sum_{i=1}^{m}\{1 - |f_{ix} - f_{iw}|\} + \frac{1}{\sum_{j=1}^{c_{\max}} \mathbb{S}(p_j^x)} \sum_{j=1}^{c_{\max}} \mathbb{S}(p_{jx}) \cdot \{1 - d(p_{jx}, p_{jw})\}$$
$$\text{where,} \quad \mathbb{S}(p_{jx}) = \begin{cases} 1, & \text{if } p_{jx} \neq (0,0) \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

where d is Euclidean distance function. The first part of the expression computes the global similarity using the node feature, and the second part computes the local similarity using position vectors of child nodes.

3.2.2. Network training

Fig. 3 illustrates the block diagrams of the MLHN training. At first, the layer-3 SOM is trained with level-3 nodes, where nodes' features are directly used as data. In the iterative SOM training, a node is random selected, its compared to all neurons to find its winner neuron, and finally the winner and its neighborhood neurons are updated using Eqs. (4) and (5). After the layer-3 SOM is trained, the winner neurons for all level-3 nodes are found, and their corresponding position vectors are saved. Using all the two-dimensional position vectors a one-dimensional SOM is then trained, which is used for mapping of the position vectors into layer-2 inputs. Thus layer-2 SOM inputs are made up for corresponding level-2 nodes using the nodes' features and their child's position vectors. It should be noted that the number of child nodes of a node is not fixed. However, the input for a parent node is represented by a fixed vector, part of which represents the child nodes. Using a mapping procedure, position vectors are mapped into the fixed structure of the input vector. Using the above SOM inputs, layer-2 SOM is then trained in the same way of layer-1 but using similarity function in Eq. (10). The layer-1 inputs corresponding to root nodes are made up in the same way. And finally an MLP network is trained using the layer-1 inputs. For the MLP classifier we hired a standard MLP tool but we do not include the details of the MLP here. Readers are referred to any standard book such as (Haykin, 1994).

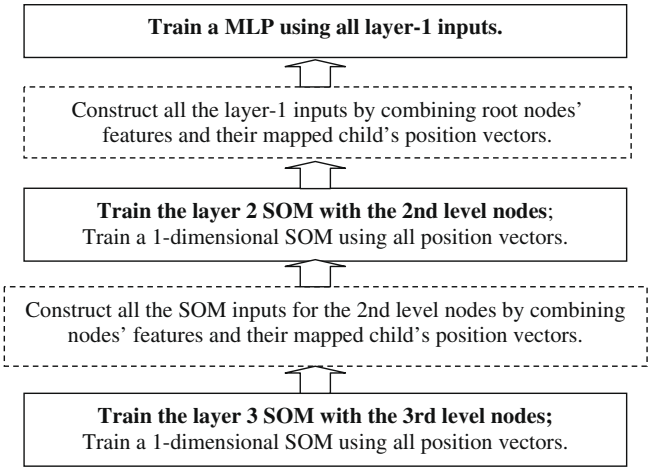

Fig. 3. The training sequences of MLHN for the three-level tree structured data.

3.2.3. Mapping of position vectors

The mapping of position vectors is required in making the inputs at layer-3 and layer-2. After the mapping, similar type of position vectors are mapped into a particular position of input vector. As a result, when two sets of child nodes are compared by Eq. (10), we are able to compare a child node of the first set with only one similar child node of another set. Thus, we have a meaningful "one to one" matching that is much faster compared with "many to many" matching (Chow et al., 2006). In this way, uncertain number of child nodes can be represented by a fixed and compact vector which is essential for using any neural network such as SOM and MLP. For each level-3 and level-2, a one-dimensional SOM is trained by all the position vectors ($p = \{x, y\}$) from that level. After training the one-dimensional SOM, the following procedure is applied to map a set of position vectors.

Mapping of position vectors $\{p_1, p_2, \ldots p_c\}$ ***using 1-d SOM of*** $c_{\max}$ ***neurons:***

Set $p_j^s \leftarrow [0,0], (N_j \leftarrow 0 j = 1 \ldots c_{\max}$

Loop for **each** $p_i, i = 1 \ldots c$

Find three most matched neurons j, k and l for p_i

If $N_j = 0, p_j^s \leftarrow p_i$ and $N_j \leftarrow N_j + 1$

else if $N_k = 0, p_k^s \leftarrow p_i$ and $N_j \leftarrow N_k + 1$

else if $N_l = 0, p_l^s \leftarrow p_i$ and $N_j \leftarrow N_k + 1$

else $p_j^s \leftarrow \frac{p_j^s \times N_j + p_i}{N_j + 1}$ and $N_j \leftarrow N_j + 1$

end if

End

Return $\{p_j^s\}$

The above procedures differs from (Chow et al., 2006) in a way that two or more position vectors in these procedures can be merged together by averaging. Thus, $c_{\max}$ is set to a value that is lower than the actual maximum number of child nodes. The length of the input vector that represents child-nodes is then reduced, which results in a reduction of the dimension of SOM weight vector. The decrease of $c_{\max}$ is useful when the maximum number of child nodes in the database is large. In our application, $c_{\max}$ is set to $\frac{2}{3}S_S$, where S_S is the length of square SOM at corresponding MLHN layer. The weight vectors of the one-dimensional SOMs are saved together with that of SOM and MLP, which will be later used to process a query document.

3.2.4. Classification of a query

To classify a given query document by MLHN, firstly the tree-structured features of the document is extracted. Then the nodes of the tree are mapped in MLHN in bottom-up fashion. First, level-3 nodes are mapped into layer-3 SOM and the position vectors are collected. Using the position vector, the level-2 nodes' input are generated and mapped on the layer-2 SOM. Layer-1 input is generated same the way which is then fed to MLP for classification of the document. The overall procedure can be summarized as follows:

Procedures of classifying a query document:

1. Extract tree-structured data of the query document.
2. Process level-3 nodes on layer-3 SOM, and get the position vectors.
3. Form the layer-2 inputs corresponding to the level-2 nodes. The previously saved one-dimensional SOM is used to map the position vector into the input.
4. Similarly, map the level-2 nodes into layer-2 SOM and form the layer-1 input.
5. Present the layer-1 input into the MLP for classification of the document.

4. Experimental results

We have used a document collection named "*Html_CityU1*", which is available at ' www.ee.cityu.edu.hk/~twschow/Html_CityU1.rar'. Our database consists of 5200 html documents organized through a three-level hierarchical *database-tree* as depicted in Fig. 4. At each level database-tree, the collection is divided into a number of sub-categories. The total number of sub-categories at levels 1, 2 and 3 are 7, 12 and 20, respectively. In total, there are 26 categories of documents, which is the number of leaf nodes (that have no child node) in the tree. Each category contains 200 documents. It should be noted that traditional classification involves a dataset where the training is usually larger or equal to testing set. However, due to the nature of the current application it is important that the system can train itself based on a training set, which is much smaller compared with test set. This is because the objective of this application is automation of the classification process, and save human labor such as labeling documents of the training set. Thus, unlike (Dumais & Chen, 2000) we used a training set that contains limited number of document, which is much smaller than the testing set. This makes the classification problem harder. Our training set consists of 1040 documents, i.e., 40 documents from each category, and rest 4160 documents made the testing set.

In this application, the hierarchical classification problem is implemented by flat classifier. It means we first classify a query into any leaf node of Fig. 4 (a class out of 26) and then use the tree to assign the query to hierarchical parent nodes at different level. For example, a document is classified into the leaf node '*Java-jsp-servlet*'. The document is then automatically assigned to its parent node '*IT-language*' and parent's parent node '*IT*'.

Node distribution of tree-structured document feature at different levels is listed in Table 1. According to the node distribution, the size of the layer-2 SOM and layer-3 SOM are set at 36×36 and 42×42, respectively. In training the MLHN, the initial learning rate was set to 0.3. The initial radius of the neighborhood function was set to half-length of the square grid at an SOM layer. The number of total training iterations was set to 4534 and 14,536 (which are the rounded multiple of the number of data-nodes in the corresponding level) for the layer-2 SOM and layer-3 SOM, respectively. To train the MLP the number of hidden neurons, initial learning rate and training iterations were set to 40, 0.1 and 3120 (which is the rounded multiple of the number of root nodes), respectively. The values of the above parameters were observed to be a good choice. All simulations were performed using MATLAB 7 on a PC with Intel 1.8 GHz and 2 GB memory.

We have compared the proposed approach with LSI approach that is implanted by using LSI features and MLP. We also include the results from VSM approach using VSM features and MLP. Here the VSM feature is word histogram (Eq. (2)) using VSM normalization($h_t = n_t \times \log(N/(f_d))$). Table 2 summarizes the results from the three approaches. Classification accuracy is listed separately for different levels as well as for leaf nodes of database tree. The results show that the proposed approach can deliver superior results for overall classification compared with other two approaches. The results of LSI and VSM approaches are comparatively inferior at level-3 and level-1, respectively. It should be noted that discriminating keyword feature among classes are narrow at level-3, whereas that at level-1 are wide. The LSI features are more global in nature compared with VSM features because LSI features are compressed projected information from VSM feature. As a results LSI perform better on level-1 compared with VSM, whereas VSM comparatively perform better on level-3 as some information are lost in LSI due to projection. In overall, considering 26 class/leaf nodes, VSM perform better than LSI. On the other hand, the results of the proposed ap-

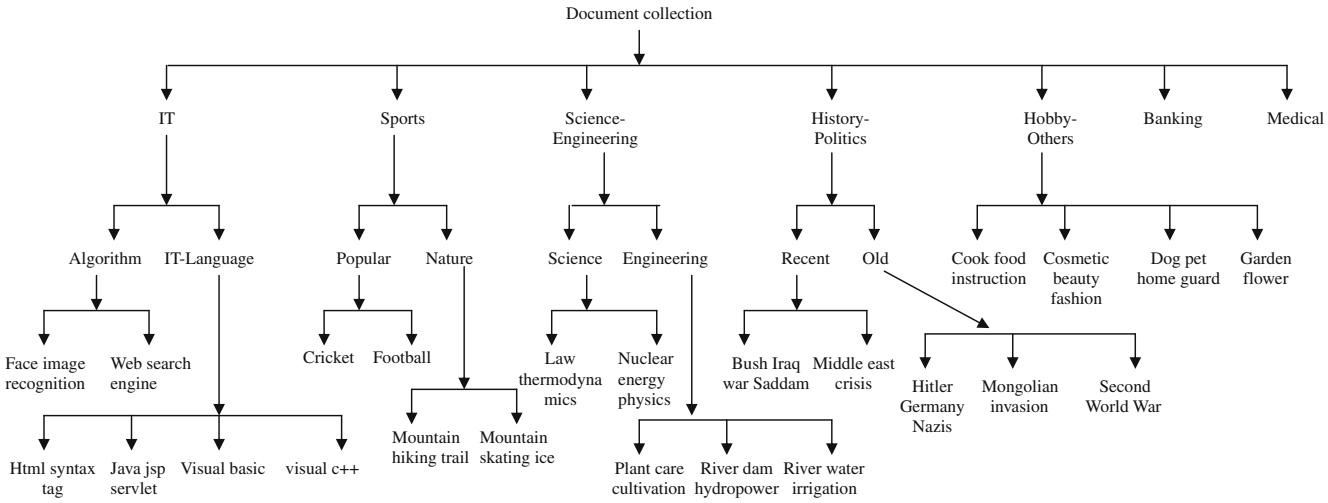

Fig. 4. Hierarchical organization of the document database.

Table 1
Distribution of nodes in data set

Level	1	2	3	All levels
Max. number of children nodes	277	87	0	277
Total number of nodes in training set	1040	2267	14,536	17,843
Total number of nodes in test set	4160	8865	53,684	66,709

Table 2
Comparative results: proposed approach, LSI, VSM

	Proposed approach	LSI–MLP	VSM–MLP
Level-1	98.1971	97.6202	93.1731
Level-2	95.4167	97.6042	92.7604
Level-3	87.1250	73.9375	85.2188
Overall (leaf nodes)	88.4135	78.4856	85.1923

proach are consistently good at all levels. The proposed tree-structured approach encodes the word-frequency information in a meaningful hierarchical way that encodes the spatial distribution of words along the document. Such spatial information is essential in encoding better semantics of the document. As a result, the tree-structure features, in spite of the use of compressed information like LSI, can deliver better results compared with both LSI and VSM.

To affirm the stability of the network, we included the results from different initialization of network weights before the training. Table 3 lists results from four different initializations that confirm that the performance of the proposed system is consistent. In Table 4 we included results from using different number of epochs for MLP, where one epoch consists of iterations equal to the number of data. We observed no noticeable decline of performance when the number of epochs is reduced from 5 to 1. Finally, classification performance is analyzed against hidden number of neurons. Fig. 5 summarizes the overall classification accuracy as well as that at different level when the number of hidden neurons is increased

Table 3
Stability of the results against different training instances.

Trials	1	2	3	4
Level-1	97.6683	97.1635	97.1394	97.3798
Level-2	95.3385	95.1563	94.9740	95.1302
Level-3	87.9375	88.7500	87.3125	88.7500
Overall (leaf nodes)	88.6298	88.8702	88.1971	89.1346

Table 4
Results against different the number of epochs.

Epochs	1	2	3	4	5
Level-1	97.4760	97.4519	98.1971	97.4519	97.4760
Level-2	95.1563	95.3906	95.4167	95.3125	95.4167
Level-3	87.5000	88.8125	87.1250	88.0938	88.4063
Overall (leaf nodes)	88.6058	89.3750	88.4135	89.0625	89.1587

from 10 to 50. It was noticed a descent performance is achieved by the system when the number of hidden neurons is equal or greater than the number of classes (leaf nodes in database-tree).

At last we investigated how the proposed tree-structured features perform compared with stand-alone global features or local features only. Global features are obtained by using only the root nodes' feature from our tree-structured representation, and nodes at the second and third levels are ignored. Thus, the format of global features is traditional flat vector type, and an MLP layer is used for the classification task. For local features, we exclude the features from the root nodes, and use the rest of tree-structured representation. Thus local features contain the information from pages and paragraphs that are organized in the hierarchical tree. The

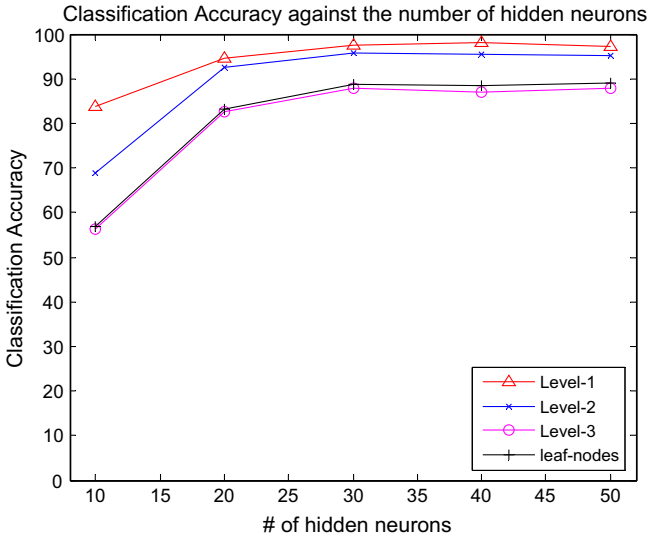

Fig. 5. Classification performance against the number of hidden neurons.

Table 5
Comparing feature types: global, local and hybrid tree.

Feature type	Global	Local	Hybrid-tree
Level-1	93.8221	91.1538	98.1971
Level-2	91.7708	86.1198	95.4167
Level-3	81.7188	75.3438	87.1250
Overall (leaf nodes)	82.9808	74.4231	88.4135

same MLHN model is used to process the local features. Table 5 summarizes the results from the three types of feature representations: global, local and hybrid-tree. Compared with the local and global, the hybrid-tree performs better overall classification as well as classifications in all levels. These results corroborate the superiority of the tree-structured features, which achieved by combining global features with local characteristics that contains hierarchical and spatial information of word-frequencies.

5. Conclusion

In this paper, we propose a tree-structured feature representation for document classification that is used for automatic organization/association of documents into the nodes of a database-tree. Tree-structured representation includes the spatial information of word distribution throughout the document which is vital in understanding the underlying semantics of the document content. Thus, compared with only global characteristics of traditional feature, tree-structured features hierarchically include both global and local characteristics of document that helps to enhance the classification accuracy. For classification of the tree-structure data, we developed the MLHN model which is an extension of our previously developed unsupervised MLSOM model. Comparative results corroborate the superiority of the proposed approach to other compared approaches. Robustness of the proposed system is also presented against network parameters.

Acknowledgement

This project was wholly supported by the Research Enhancement fund of the Department of Electronics Engineering, City University of Hong Kong.

References

Ampazis, N., & Perantonis, S. (2004). LSISOM, a latent semantic indexing approach to self-organizing maps of document collections. *Neural Processing Letters, 19*(2), 157–173.

Appiani, E., Cesarini, F., Colla, A. M., Diligenti, M., Gori, M., Marinai, S., et al. (2001). Automatic document classification and indexing in high-volume applications. *International Journal on Document Analysis and Recognition, 4*(2), 69–83.

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review, 37*(4), 573–595.

Cho, S. Y., & Chi, Z. (2005). Genetic evolution processing of data structures for image classification. *IEEE Transactions on Knowledge and Data Engineering, 17*(2), 216–231.

Cho, S. Y., Chi, Z., Siu, W. C., & Tsoi, A. C. (2003). An improved algorithm for learning long-term dependency problems in adaptive processing of data structures. *IEEE Transactions on Neural Networks, 14*(4), 781–793.

Chow, T. W. S., Rahman, M. K. M., & Wu, S. (2006). Content based image retrieval by using tree-structured features and multi-layer SOM. *Pattern Analysis and Applications, 9*(1), 1–20.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of American Society for Information Science, 41*, 391–407.

Dumais, S., & Chen, H. (2000). Hierarchical classification of web content. In *Proceedings of SIGIR-00, 23rd ACM international conference on research and development in information retrieval*.

Fall, C., & Benzineb, K. (2002). Literature survey: Issues to be considered in the automatic classification of patents. *World Intellectual Property Organization*. http://www.wipo.org/classifications/ipc/en/ITsupport/Categorization/wipo-categorizationsurvey.pdf.

Freeman, R. T., & Yin, H. (2005). Content management by self-organization. *IEEE Transactions on Neural Networks, 16*(5), 1256–1268.

Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M., & Tzeras, K. (1991). Air/X – A rule-based multi-stage indexing system for large subject fields. In *Proceedings of RIAO'91* (pp. 606–623).

Hagenbuchner, M. (2002). *Extension and evaluation of adaptive processing of structured information using artificial neural networks*. PhD dissertation, Faculty of Informatics, University of Wollongong.

Hagenbuchner, M., & Tsoi, A. C. (2003). A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks, 14*(3), 491–505.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. IEEE Press.

Honkela, T., Kaski, S., Lagus, K., & Kohonen, T. (1997). WEBSOM–self-organizing maps of document collections. In *Proceedings of WSOM'97, workshop on self-organizing maps* (pp. 310–315). Espoo, Finland, Helsinki University of Technology: Neural Networks Research Centre.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of European conference on machine learning (ECML'98)*.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the 14th international conference on machine learning (ICML'97)* (pp. 170–178).

Merkl, D., He, S. H., Dittenbach, M., & Rauber, A. (2003). Adaptive hierarchical incremental grid growing: An architecture for high-dimensional data visualization. In *Proceedings of the 4th workshop on self-organizing maps, advances in self-organizing maps, Kitakyushu, Japan* (pp. 293–298).

Papadimitriou, C. H., Raghavan, P., Tamaki, H., & Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences, 61*(2), 217–235.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130–137.

Rahman, M. K. M., Yang, W. P., Chow, T. W. S., & Wu, S. (2007). A flexible multi-layer self-organizing map for generic processing of tree-structured data. *Pattern Recognition, 40*(5), 1406–1424.

Rasmussen, E. H. (2005). BBTools – A Matlab toolbox for black-box computations. Neurobiology Research Unit, Copenhagen University Hospital. http://nru.dk/software/bbtools/.

Rauber, A., Dittenbach, M., & Merkl, D. (2000). Automatically detecting and organizing documents into topic hierarchies: A neural network based approach to bookshelf creation and arrangement. In *Proceedings of the 4th European conference on research and advanced technologies for digital libraries, Lisboa, Portugal* (pp. 18–20).

Salembier, P., & Garrido, L. (2000). Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing, 9*(4), 561–576.

Salton, G., & Buckley, C. (1996). Term weighting approaches in automatic text retrieval. *Information Processing and Management, 32*(4), 431–443.

Schütze, H., Hull, D., & Pedersen, J. O. (1995). A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'95)* (pp. 229–237).

Wang, Z., Hagenbuchner, M., Tsoi, A. C., Cho, S. Y., & Chi, Z. (2002). Image classification with structured self-organization map. In *Proceedings of the international joint conference on neural networks* (Vol. 2, pp. 1918–1923).

Weigend, A. S., Wiener, E. D., & Pedersen, J. O. (1999). Exploiting hierarchy in text categorization. *Information Retrieval, 1*(3), 193–216.

Yang, Y. (1994). Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'94)* (pp. 13–22).

Zhao, R., & Grosky, W. (2002). Narrowing the semantic gap – Improved text-based web document retrieval using visual features. *IEEE Transactions on Multimedia, 4*(2), 189–200.

Zobel, J., & Moffat, A. (1998). Exploring the similarity space. *ACM SIGIR Forum, 32*(1), 18–34.