Tree2Vector: Learning a Vectorial Representation for Tree-Structured Data

Haijun Zhang, *Member, IEEE*, Shuang Wang, Xiaofei Xu, Tommy W. S. Chow, *Senior Member, IEEE*, and Q. M. Jonathan Wu, *Senior Member, IEEE*

***Abstract*— The tree structure is one of the most powerful structures for data organization. An efficient learning framework for transforming tree-structured data into vectorial representations is presented. First, in attempting to uncover the global discriminative information of child nodes hidden at the same level of all of the trees, a clustering technique can be adopted for allocating children into different clusters, which are used to formulate the components of a vector. Moreover, a locality-sensitive reconstruction method is introduced to model a process, in which each parent node is assumed to be reconstructed by its children. The resulting reconstruction coefficients are reversely transformed into complementary coefficients, which are utilized for locally weighting the components of the vector. A new vector is formulated by concatenating the original parent node vector and the learned vector from its children. This new vector for each parent node is inputted into the learning process of formulating vectorial representation at the upper level of the tree. This recursive process concludes when a vectorial representation is achieved for the entire tree. Our method is examined in two applications: book author recommendations and content-based image retrieval. Extensive experimental results demonstrate the effectiveness of the proposed method for transforming tree-structured data into vectors.**

***Index Terms*— Author recommendations, image retrieval, locality reconstruction, tree structure, vectorial representation.**

I. Introduction

THE tree structure, one of the most powerful tools for representation, is ubiquitous in nature. Many applications in database, data mining, and image processing are attributed to the use of tree structures. For example, a lengthy book can be represented in a hierarchical way of "book→sections→paragraphs→sentences" [1]. Traditional methods, however, usually use a "bag-of-words" model to transform this natural tree-structured representation into a single vector for book recommendations, classification, or clustering. As a result, the term spatial information in the hierarchy is overlooked. In practice, a vectorial representation for a tree is always desirable for calculating in-between similarity in many applications. Therefore, our objective lies in learning a vectorial representation for tree-structured data, which preserves discriminative information coming from different levels of the tree. Most current studies of tree-structured data in the fields of data mining primarily work on tree-structured data indexing, image tree for image representation, and document tree for textual representation.

Numerous modern database applications rely on the use of tree-structured data. Researchers usually focus on studying the structure similarity measure and similarity search on large trees in huge data sets [2]. Edit distance is commonly used to measure the dissimilarity between trees. Various tree edit distance algorithms mainly differ in the set of allowed edit operations [3]–[5]. However, reducing tree-edit computation is essential for making similarity search scalable on large trees in huge data sets. Guha *et al.* [6] proposed a pivot-based approximate similarity join algorithm for handling XML documents. Garofalakis and Kumar [7] proposed to embed tree-edit distance metrics into a numeric vector space. A constant lower bound, however, cannot be explicitly given by the method on the tree-edit distance. Subsequently, a set of filters was introduced to perform on structure and content-based information in trees [8]. Yang *et al.* [2] provided accurate lower bounds for the tree-edit distance by integrating the two sources of information. These above-mentioned methods mainly work on tree-structured data querying in large databases. They attempt to improve similarity search efficiency and solve the scalability problem by developing different structure similarity measures. In this paper, however, we focus on investigating tree-structured data from the node representation perspective. The resulting vectorial representation of the entire tree-structure datum obtained by our method can be easily stored and indexed in a database.

Tree-structured representation has shown potential for image analysis [9]. There is evidence suggesting that region-based image representation [10], [11] can be better encoded by a tree representation [12]–[15]. A binary space partition (BSP) tree was used for image representation [13], [14]. Later studies [15], [16] suggested that the BSP-tree-based region-oriented image representation is effective for image classification and retrieval. Despite efficiency of BSP for image processing, dividing a region into two subregions in BSP cannot assure a meaningful segmentation of objects in real-life images. For example, the height of the BSP tree increases

Manuscript received November 13, 2016; revised May 5, 2017 and December 18, 2017; accepted January 16, 2018. This work was supported in part by the Natural Science Foundation of China under Grant 61572156 and in part by the Shenzhen Science and Technology Program under Grant JCYJ20170413105929681. *(Corresponding author: Haijun Zhang.)*

H. Zhang, S. Wang, and X. Xu are with the Shenzhen Graduate School of Harbin Institute of Technology, Shenzhen 518055, China (e-mail: hjzhang@hit.edu.cn; wangshuang616@gmail.com; xiaofei@hit.edu.cn).

T. W. S. Chow is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: eetchow@cityu.edu.hk).

Q. M. J. Wu is with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: jwu@uwindsor.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2797060

linearly with the number of objects, when subregions lack strong ability to represent the image contents. To overcome these limitations, a new tree-based image representation was introduced [17]. Moreover, an extended self-organizing map (SOM)-based system was reported for efficient content-based image retrieval (CBIR) [17]. Kohonen [18] also extended the traditional SOM algorithm into a multilayer version for processing tree-structured image data. Apart from these applications of image retrieval using tree-structured representation, recent research has shown the potential of a spatial pyramid image representation (a kind of quad-tree structure) for recognizing natural scene categories and objects [19]–[21]. However, these methods devote much efforts to developing pyramid matching schemes using an aggregated vector representation while preserving the spatial information of the pyramid.

Tree-structured representations have also shown effectiveness for document modeling. Observing that the traditional bag-of-words models [22]–[25] discard the term spatial information hidden in a document, Chow and Rahman [26] proposed a tree structure in a way of "document→pages→paragraphs" for document retrieval and plagiarism detection. In particular, they applied a multilayer self-organizing map (MLSOM) to handle the tree-structured representation and speed up the document indexing process attributed to the clustering capability of SOM. Instead of SOM, it is possible to design multilayer versions of multidimensional scaling [51], multivariate analysis [52], and t-distributed stochastic neighbor embedding [53] for addressing tree-structured data. Zhang and Chow [27], [28] used a similar tree structure for representing each document and employed earth mover distance for tree matching. A triple wing harmonium model that projects text metadata into a low-dimensional semantic space was then applied to content-based movie recommendations [29]. Recently, a four-layer tree-based representation, "author→books→pages→paragraphs," was used for book and author recommendations [1]. MLSOM employed in [1] has also shown efficiency for this task.

Despite promising results achieved by MLSOM, the resulting image and document tree-based representations depend heavily on the clustering capability of MLSOM. It is difficult for MLSOM to formulate a "global" noninvariant vectorial representation of each tree-structured datum based on aggregating statistics of local features over children nodes. Motivated by this, in this paper, we introduce a new learning framework of formulating vectorial representations for general tree-structured data (image or text), whereby the resulting vectorial representation can facilitate the querying process in various database applications without calculating the tree edit distance. In fact, there exist several models of graph neural networks (GNNs) [46], which extend multilayer perceptron into the domain of graphs (cyclic and acyclic), and recursive neural networks (RNNs) [47], with an input domain consisting of directed acyclic graphs, which are mapped into vectors of reals. Moreover, in the field of natural language processing, researchers have designed different kernels that used a method of counting the number of common substructures of two trees [48]–[50]. The main difference between these methods and our method is that GNNs, RNNs, and kernel methods often fall into the supervised machine learning category, whereas our method is an unsupervised one. Specifically, in order to uncover the global discriminative information of child nodes over the same level of all of the trees, a node allocation process favored by a clustering technique, k-means, is developed for assigning child nodes into different clusters. This node allocation process is utilized to formulate the corresponding components of a vector. Moreover, a locality-sensitive reconstruction (LSR) method is introduced to model the reconstruction process, in which each parent node is assumed to be reconstructed by its children. The resulting reconstruction coefficients are used for locally weighting the components of the vector. A new vector is formulated by concatenating the original parent node vector and the learned vector including information from its children. This new vector for each parent node is inputted into the learning process of formulating vectorial representation at the upper level of the tree. This process is repeated until a vectorial representation is achieved for the entire tree. We examined our method in two applications: author recommendation and CBIR. Experimental results demonstrate the effectiveness of our proposed method.

The remaining sections of this paper are organized as follows. Section II presents a tree structure studied in this context and its modeling method for such a tree in order to formulate a vectorial representation. Section III describes two real-world applications using our proposed method: author tree and image tree. Experimental verifications have been conducted and explained in Section IV. Section V concludes this paper with suggestions for future work.

II. Learning a Vectorial Representation

In this section, we first define the tree studied within this context. A framework that aims at transforming the trees into vector space is then introduced. Finally, we summarize our entire algorithm together with its computational complexity analysis.

A. Tree Structure

By definition, a tree is a data structure including nodes (or vertices) and edges without having any cycle. An unempty tree has a root node and potentially many levels of additional nodes in a hierarchy. In this paper, we only focus on the trees that have a varying number of children per node, and each of which has the same number of levels (the same depth). Fig. 1 shows an example of a three-level tree structure with eight nodes. Formally, given a data set Ω, each sample is represented by a tree T_i, i.e., $\Omega = \{T_i\}$ $(i = 1, 2, \ldots, n)$. Each node in tree T_i can be included a set of information indicating how the nodes are dependent on each other in the tree structure. The set of information consists of: 1) node index; 2) node level at the tree; 3) parent node index; 4) child node indexes; and 5) node feature. Let $T_i = \{v_k\}$ $(k = 1, 2, \ldots, m)$, where m denotes the number of nodes in tree T_i, and v_k represents the kth node in T_i. Node v_k contains a well-defined set of information encoded in the form of $v_k = \{I_k^d, L_k^e, I_k^{\text{parent}}, I_k^{\text{children}}, F_k^G\}$, where I_k^d, L_k^e, I_k^{parent}, I_k^{children}, and F_k^G denote the index of v_k, the level (or depth) in T_i, the parent node index, the set of children

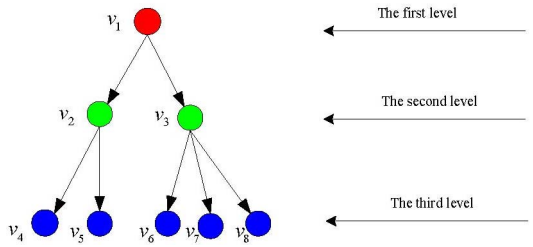

Fig. 1. Author representation by tree-structured feature.

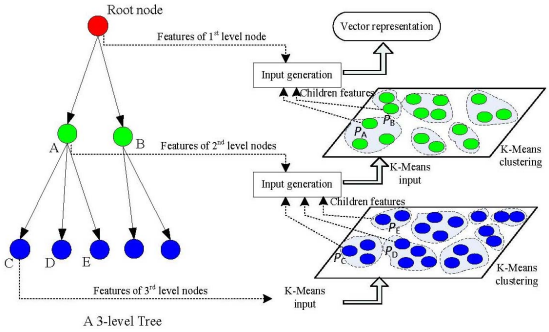

Fig. 2. Framework of Tree2Vector.

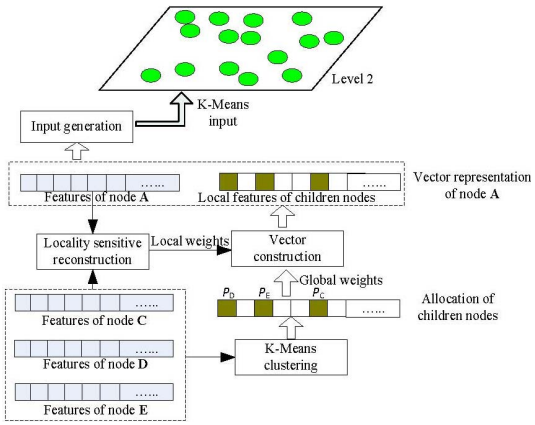

Fig. 3. Input generation process.

node indexes of v_k, and the features that represent the content information of v_k, respectively. It is worth noting that I_k^{parent} and I_k^{children} are empty for the root node and the nodes at the bottom layer in T_i, respectively.

B. Tree2Vector

Many real-world applications rely on in-between similarities among samples. If we only store these aforementioned trees into databases without any transformation, it is necessary for us to calculate the tree edit distances when querying, as done in [2]–[8]. However, in this paper, our objective lies in designing a learning framework for transforming tree-structured data into vector space. For brevity, we call this framework as Tree2Vector in the following context. This Tree2Vector transformation can not only save database space but also speed up the querying process in different applications. A challenging problem is determining how to aggregate the features of children nodes into the root node while preserving the discriminative information hidden at the low-level layers. In order to accomplish this, we design a new framework, Tree2Vector. Fig. 2 shows the process that three-level tree data are projected into vector space. First, the third-level nodes are inputted into a clustering algorithm. For simplicity, we used the k-means algorithm [30] for clustering. Certainly, other advanced algorithms [31] can also be employed for this task. We allocate nodes (e.g., C, D, and E) into different clusters, described by cluster indexes (e.g., p_C, p_D, and p_E). It is noted that the bottom layer usually comprises a huge number of child nodes in a data set. Using the clustering techniques enables us to project all of the nodes into a few numbers of clusters, but the topology over the original nodes is preserved. As a result, the input vector of layer 2 k-means contains the second-level node features and the corresponding child features derived from the third-level nodes. Fig. 3 shows an example that a second-layer input feature vector including the feature of node A and the features of its children is formulated. Fig. 3 also demonstrates how local features of children nodes are generated from the outputs of layer 3 k-means. In this generation process, we mainly used two weights, global and local weights, to construct a vector that is used for representing local features of children nodes. The method for learning the global and local weights will be described in detail in Section II-C. Similarly, the second-level nodes and the root nodes can be processed. This recursive process is utilized to embed the features associated with the information hidden at lower layers. The feature of root node at the top layer is able to compactly encode the whole tree as a vector, which includes the features of the root node on its own and the local features formulated by this bottom-to-up procedure. This vectorial representation can then be easily used for various applications, such as retrieval, classification, and clustering.

C. Algorithm and Implementation

1) Node Allocation: A node feature vector F_l at the lth level ($l = 1, 2, \ldots, L^e - 1$, where L^e is the depth of a tree) is denoted by $F_l = [f_{1,l}^G, f_{2,l}^G, \ldots, f_{d_l,l}^G, f_{1,l}^L, f_{2,l}^L, \ldots, f_{d_l,l}^L]^T$, where $f_{i,l}^G$ represents the ith feature of node F_l, and $f_{j,l}^L$ denotes the jth local feature of node F_l obtained from its children nodes. The first set of components $\{f_{i,l}^G\}$ in F_l indicates the information of a node on its own, whereas the second set of components $\{f_{j,l}^L\}$ includes the information delivered to its children. d_l is the dimension of node feature on its own; it is also the number of clusters predefined in k-means at the $(l+1)$th level under our framework. The local feature vector $[f_{1,l}^L, f_{2,l}^L, \ldots, f_{d_l,l}^L]^T$ can be obtained according to the spatial distribution over the k-means clustering result that will be discussed later in this section.

At each level, we input all of the node feature vectors $\{F_l\}$ into a k-means for clustering. In the clustering process, we have used different functions to compute the distance between two nodes for different applications (see Section III). The number of clusters is determined by the dimension size of node features at the upper level. The rationale behind this

Algorithm 1 Node Allocation Process

Initialize: Set $N_j \leftarrow 0,\ j = 1, 2, \ldots c_{\max}$
for $k = I_1$ to I_2 **do**
  Find three most nearest cluster center c_a, c_b, c_c for v_k
  if $N_{c_a} = 0$ **then**
    Allocate v_k into c_a and $N_{c_a} \leftarrow N_{c_a} + 1$
  else if $N_{c_b} = 0$ **then**
    Allocate v_k into c_b and $N_{c_b} \leftarrow N_{c_b} + 1$
  else if $N_{c_c} = 0$ **then**
    Allocate v_k into c_c and $N_{c_c} \leftarrow N_{c_c} + 1$
  else
    Allocate v_k into c_a and $N_{c_a} \leftarrow N_{c_a} + 1$
  end if
end for
Return: Set of indexes of nodes assigned into each cluster c_j

setting is to balance the global information contained in a parent node and the local information delivered by its child nodes. After clustering, each node is allocated to a cluster. It should be noted that the dimension size of node features (or the number of clusters for k-means) is usually larger than the number of children nodes for a given node. As a result, some of $[f^L_{1,l}, f^L_{2,l}, \ldots, f^L_{d_l,l}]^T$ may contain components with zeros. For example, in Fig. 3, nodes C, D, and E were assigned to the fifth, first, and third clusters, respectively. The other components in the local feature vector will be set to 0. We then calculate the distance $s^{\mathrm{NC}}_{\cdot,j}$ between each node and the center of the cluster to which the node belongs. The detailed calculation for the distance measure will be shown in our applications (see Section III). We regard these distance values between nodes and cluster centers as global weights, which are used to capture the discriminative information among child nodes at the same level.

In our applications, a node may contain a large number of child nodes. In the clustering process, some child nodes may be allocated to the same cluster. In order to increase the discriminative capability of local features, we utilized a node allocation procedure to assign each child into clusters. Formally, given a subset of the nodes in tree T_i, let $V_{i,n} = \{v_{I_1}, v_{I_1+1}, \ldots, v_{I_2}\}$ denote the children set of the nth node in tree T_i, where I_1 and I_2 are the minimal node index and the maximal node index of child node v_k, respectively. Let c_j $(j = 1, 2, \ldots, c_{\max})$ represent the jth cluster center, where $c_{\max}$ denotes the number of clusters at a specified level. Let N_j count the number of nodes from $V_{i,n}$ assigned into the jth cluster. The detailed node allocation process from $V_{i,n}$ can be accomplished by Algorithm 1.

For clarity, an example of this node allocation process is given in Fig. 4. For a parent node A, it contains five child nodes: C, D, E, F, and G. By using the k-means clustering, we can find three closest clusters for each child node. For example, there are ten clusters in total. The set of the three most matched clusters in descending order according to the distance between the child node and the center of a cluster is listed for each node on the right-hand side of Fig. 4. If we use

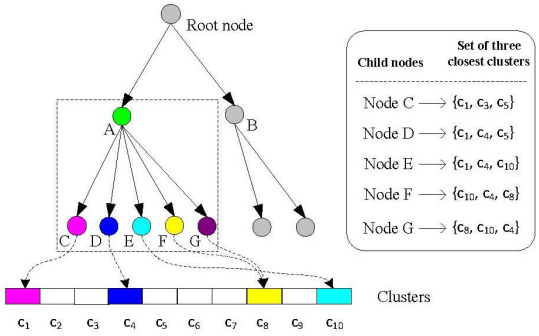

Fig. 4. Example of node allocation process.

the simple rule that each child node is assigned to its closest cluster, nodes C, D, and E will all be allocated into cluster c_1 together. This allocation will reduce the discriminative power among different child nodes to some extent. By using our allocation process shown in Algorithm 1, however, nodes C, D, E, and F will be allocated into clusters c_1, c_4, c_{10}, and c_8, respectively. For node G, each cluster in the set of three closest clusters has been visited by other nodes. In this situation, it will be allocated into the closest one, i.e., c_8. As a result, such a node allocation process makes it most possible to preserve the discriminative information hidden in child nodes.

2) Locality-Sensitive Reconstruction: In a tree structure, each child node usually contains its own new feature and certain partial information inherited from its parent node. Integrating a node's information on its own and the complementary information of new features produced by its child nodes is essential for generating a vectorial representation for a tree. In order to mine the complementary information of child nodes and measure the importance of each child node for its parent node, we introduce an LSR model, which is motivated by a recent success of sparse coding [32]–[34]. The original sparse coding aims at reconstructing an input instance as a compact linear combination of dictionary atoms. By observing that *data locality* constitutes a key issue in many applications, Wang *et al.* [33] proposed locality-constrained linear coding scheme, which replaces the ℓ_1-norm sparsity regularization by a locality adaptor. Inspired by this sparse coding scheme, we assume that a parent node in a tree structure is viewed as a compact linear combination of its child nodes. In practice, the semantic relationship between a parent node and its child nodes in a general tree structure should be very complicated. Developing nonlinear models for specific applications is worth investigating in the future. Specifically, we can use the following optimization problem to model the process of how child nodes reconstruct their parent node:

$$\begin{aligned} &\min_{\beta} \ \left(\left\| F^G_{i,l} - D\beta \right\|_2^2 + \lambda \| q \odot \beta \|_2^2\right) \\ &\text{s.t } 1^T\beta = 1 \end{aligned} \tag{1}$$

where $F^G_{i,l}$ represents the original node feature vector associated with the ith node at the lth level, the symbol $\odot$ indicates the element-wise multiplication, D is the set of feature vectors of nodes that are the child nodes of the ith node at the lth level, i.e., $D = [F^G_{1,l+1}, F^G_{2,l+1}, \ldots, F^G_{k_{\max},l+1}]$ ($k_{\max} > 1$ is

the number of child nodes), λ is a predefined parameter, q is the *locality adaptor*, and the kth entry of vector q, q_k, is the distance between $F_{i,l}^G$ and the kth column of D (i.e., $F_{k,l+1}^G$). The distance measures depend on different applications (see Section III). Similar to [33], the locality adaptor gives different freedom for each child node proportional to its relation to the parent node. Additional details about the locality adaptor can be found in [33]. The constraint enforced in (1) acts like a normalization of the reconstruction coefficient vector β, which has been commonly used in sparse coding [32], [34]. In this reconstruction process, given an input node and its child nodes, we calculate the LSR coefficient β for this input node. The derived β_k ($k = 1, 2, \ldots, k_{\max}$) reflects the importance of the associated child node $F_{k,l+1}^G$. In other words, the importance rate of child node $F_{k,l+1}^G$ for its parent node $F_{i,l}^G$ is measured by the amount of information that this child node inherits from its parent node.

In order to derive reconstruction coefficient β, we formulate the Lagrange function $L(\beta, \mu)$ from (1), and solve the minimization problem in the form of

$$L(\beta, \mu) = \left\| F_{i,l}^G - D\beta \right\|_2^2 + \lambda \| q \odot \beta \|_2^2 + \mu(1^T\beta - 1). \quad (2)$$

For calculation purposes, the above function can be rewritten as

$$L(\beta, \mu) = \beta^T \Gamma \beta + \lambda \beta^T \mathrm{diag}(q)^2 \beta + \mu(1^T\beta - 1) \quad (3)$$

where $\Gamma = (F_{i,l}^G 1^T - D)^T (F_{i,l}^G 1^T - D)$, and $\mathrm{diag}(q)$ is a diagonal matrix whose nonzero elements are the entries of q. Let $\partial L(\beta, \mu)/\partial\beta = 0$, we have

$$2(\Gamma + \lambda \mathrm{diag}(q)^2)\beta + \mu 1 = 0. \quad (4)$$

By premultiplying $1^T(\Gamma + \lambda \mathrm{diag}(q)^2)^{-1}/2$ on (4), we have $1^T\beta = -\mu(1^T(\Gamma + \lambda \mathrm{diag}(q)^2)^{-1}/2)1$. Since the LSR model is subject to the constraint $1^T\beta = 1$ as shown in (1), we then obtain $\mu = -2(1^T(\Gamma + \lambda \mathrm{diag}(q)^2)^{-1}1)^{-1}$. After substituting μ into (4), we have an analytical solution to β in the form of

$$\begin{aligned} \Lambda &= (\Gamma + \lambda \mathrm{diag}(q)^2)^{-1} 1 \\ \beta &= \Lambda / (1^T \Lambda). \end{aligned} \quad (5)$$

As previously mentioned before, the obtained reconstruction coefficient β indicates the importance rate of child nodes for their parent node, which is given as the amount that the child nodes contribute to their parent node from the reconstruction process. However, in our application, the complementary information that each node conveys for its parent node is more critical, because the proposed Tree2Vector method aims at extracting the information that includes the features of a parent node and the complementary information brought by its child nodes in a bottom-to-up manner. Therefore, we use the following form to reflect the amount of complementary information that each child node produces on its own:

$$\hat{\beta} = (1 - \beta)/(k_{\max} - 1). \quad (6)$$

For the case $k_{\max} = 1$ (i.e., a node has only one child), we do not need the above reconstruction process. We use the obtained *complementary reconstruction coefficient* $\hat{\beta}$ as the local weights to construct the local features as shown in Fig. 3.

3) Generating Local Features: For a given node F_l at the lth level, the local features $\{f_{j,l}^L\}$ can be defined in the following form:

$$f_{j,l}^L = \begin{cases} \dfrac{\sum_{k=1}^{k_{\max}} \delta(h_{k,j} = 1)\hat{\beta}_k s_{k,j}^{\mathrm{NC}}}{\sum_{k=1}^{k_{\max}} \delta(h_{k,j} = 1)}, & \text{if } \sum_{k=1}^{k_{\max}} \delta(h_{k,j} = 1) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where $\hat{\beta}_k$ is the kth element of local weights $\hat{\beta}$, $s_{k,j}^{\mathrm{NC}}$ is the distance between the kth child node and the jth cluster center as shown at the last section, $\delta(\cdot)$ is the indicator function (its value is 1 if the statement within its argument is true, and 0 otherwise), and $h_{k,j}$ is the node assignment variable, which is defined by

$$h_{k,j} = \begin{cases} 1, & \text{if child node } k \text{ is assigned to cluster} \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

From (7), it is noticed that a local feature $f_{j,l}^L$ (for $f_{j,l}^L > 0$) is determined by two factors: the global weight $s_{k,j}^{\mathrm{NC}}$, which contains the discriminative information from a global data-view by using the clustering technique, and the local weight $\hat{\beta}_k$, which conveys the complementary information from a local data-view learned from LSR. The mutual impact of these two factors is able to largely capture the characteristics hidden among child nodes at the low level of a tree. In fact, it is a straightforward way to combine these two factors in the form of (7). Instead of the linear form, the local features could be generated by other forms. We leave the exploration of other formulations of local features to future work.

By concatenating the original node features $\{f_{i,l}^G\}$ and the learned local features $\{f_{j,l}^L\}$ from child nodes, the newly produced node feature F_l will be processed onto the upper level. This process is repeated until all of the tree-structured data are mapped into vector space. Finally, the overall implementation of the Tree2Vector algorithm is summarized in Algorithm 2. Since we have saved all of the clusters' centers at different levels at the training process, for a given testing sample, it will be easy to allocate its nodes into these existing clusters by finding the nearest centers for each node.

D. Computational Complexity

Our method, Tree2Vector, is a bottom-up iterative process. At the training phase, the computational complexity is given by

$$\begin{cases} O\left(\sum_{l=2}^{L^e - 1} 2N_l d_{l-1} d_l t_l + N_{L^e} d_{L^e - 1} d_{L^e} t_{L^e} \right), & \text{if } L^e > 2 \\ O(N_{L^e} d_{L^e - 1} d_{L^e} t_{L^e}), & \text{if } L^e = 2 \\ O(1), & \text{if } L^e = 1 \end{cases} \quad (9)$$

where L^e is the maximum level of tree structures, N_l is the total number of nodes at the lth level of all of the trees, d_l is the dimension of the features of a node at the lth level, and t_l is the number of iterations of k-means. Given a testing sample, the clustering process is not required because we have stored the clustering centers at the training stage. Nodes of the testing

Algorithm 2 Training Procedure of Tree2Vector

Input: A training set Ω, where each sample is a tree T_i, i.e. $\Omega = \{T_i\}$ (i = 1,2,…,n)
Output: Vectorial representation of each tree T_i
Initialize: K-Means for all levels
Loop from the bottom to the first level
(1) Select all of the nodes of all tree-based data at the same level;
(2) Produce an input vector for each node by integrating the node's feature with projected child nodes' local features;
(3) Unless it is the first level, run the K-Means algorithm;
(4) Allocate each node into a cluster by running Algorithm 1;
(5) Calculate the distance between each node and its corresponding cluster's center and save clusters' centers for the testing stage;
(6) Learn local weights $\hat{\beta}$ using Eq.(6) for nodes that belong to the same parent;
(7) Generate local features $\{f_{j,l}^{L}\}$ using Eq.(7). This will be used for creating input vectors in Step (2);
End of level loop

sample can be allocated into clusters by finding the nearest centers from existing clusters. Therefore, at the testing stage, computational complexity is reduced to

$$\begin{cases} O\left(\sum_{l=2}^{L^e-1} 2N_l^T d_{l-1} d_l + N_{L^e}^T d_{L^e-1} d_{L^e}\right), & \text{if } L^e > 2 \\ O\left(N_{L^e}^T d_{L^e-1} d_{L^e}\right), & \text{if } L^e = 2 \\ O(1), & \text{if } L^e = 1 \end{cases} \tag{10}$$

where N_l^T is the number of nodes of a testing sample at the lth level.

The above computational complexity analysis does not include the computational cost produced by LSR. Given a tree, the reconstruction coefficient β (see Section II-C2) for each node can be calculated in advance. The computational complexity of LSR for each node (except for nodes at the bottom level) can be given by

$$O\left(2M_i^3 + (4+d_i)M_i^2 + (4+3d_i)M_i + 1\right) \tag{11}$$

where d_i is the dimension of the features of the ith node in a tree, and m_i denotes the number of child nodes of the ith node. Although m_i may be large in real-world applications, the calculation of reconstruction coefficient β for either training or test samples can be obtained offline.

III. Applications

In this section, we present two applications: author recommendation and CBIR, in order to demonstrate the effectiveness of our proposed Tree2Vector algorithm.

A. Author Recommendation

Author recommendation is an important application for online stores in e-book markets [1]. It helps people to make the optimal choice of selecting favorite authors together with their most loved books. Recommender systems have been widely studied in both academia and industry [29]. As an application example of our proposed Tree2Vector algorithm, a content-based book author recommender is considered in this context. Previous work on books [35] and authors [36] mainly rely on the traditional document modeling methods [22]–[25], i.e., the bag-of-words models. These methods usually aggregate all of the contents of book or author into a single document and consider a nontree representation. They use flat feature structure associated with a function of *tf*. This type of structure, however, is only a document-level representation, because two documents including similar term frequencies may be totally contextually different when the term spatial distribution is different. To overcome this issue, we adopt a tree structure to represent each author.

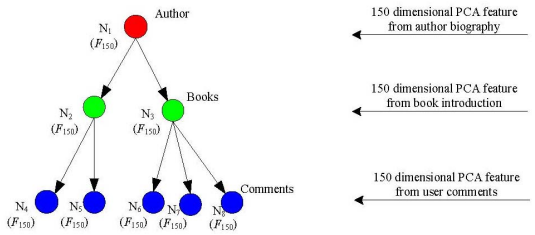

Fig. 5. Author representation by tree-structured feature.

1) Author Tree: In industry, online bookshops such as Amazon recommend books or authors by returning a list to users. For example, relevant authors are recommended by listing titles that "Customers Also Bought." In order to clearly represent the content of an author, a tree structure constructed by author biography, written books, and book comments collected from Amazon can be used. However, for content-based author recommendation, the traditional bag-of-words methods [22]–[25] usually use author biographies only, instead of combining book and comment information.

To fully describe the information of an author, a tree structure is utilized for the feature representation of Amazon authors, as presented in Fig. 5. This tree structure was previously introduced by Lu and Zhang [37]. At the first level, the root node contains the biography of an author, which may consist of the birth date, education history, political activities, early childhood, and later life of an author. Nodes at the second level indicate books that the author has written. Nodes at the third level represent book comments from other users. Thus, a tree structure in a way of a "author→books→comments" is constructed for each author. Author biographies were collected from Amazon author and book homepage. Word-frequency features of an author's biography at the top level can be extracted from the Web page. Similarly, word-frequency features of books and comments can be extracted from book homepages. In order to convey more contextual information, instead of word frequencies, other features learned by contextual machine learning methods such as word2vec [44] or paragraph2vec [45], can be fed into our Tree2Vector framework. In order to make the system computable, we compress the word histogram vectors by using principal component analysis (PCA). Thereafter, nodes are represented by compressed PCA features. Similarity between two authors can be measured

by comparing their tree-structured representations, where root nodes deliver their in-between similarity in terms of biographical introductions, and children nodes at lower levels indicate the similarity existing in their writings. Relying on such tree-structured representations, a content-based recommender can be easily implemented under our Tree2Vector framework.

2) Feature Extraction: This section briefly summarizes the generation process of an author tree. The process includes word extraction, vocabulary construction, histogram calculation, and PCA feature projection.

a) Word extraction: Author biographies were collected from the Amazon author and book homepage entitled by "About the Author" in the ".html" format. We only extracted the author texts, and other contents irrelevant to the author were filtered out. The links of written books were parsed from the author homepage. Book introduction was compiled from the book homepage and comments were collected from the title with "Customer Reviews." Subsequently, we extracted the words from all of the documents. Word correction and stemming were performed on each word. Stop words were removed from author and book nodes. For comment nodes, emotional words like "good" and "well" were retained. Then, the stemmed words associated with the *tf*, f_u^t (the frequency of the uth word in all of the documents), and the document frequency (*df*), f_u^d (the number of documents in which the uth word appears) were stored.

b) Vocabulary construction: For author biographies, according to the stored *tf* and *df*, the well-known *tf-idf* term-weighting measure is performed on each word in the form of

$$w_u = f_u^{t,\text{bio}} \cdot idf \tag{12}$$

where idf is the inverse-document-frequency calculated by $idf = \log_2(N_{\text{bio}}/f_u^d)$, and N_{bio} represents the total number of biographies in a data set. We sort the words in descending order according to their weights. Every level has its own vocabulary. In our data set (see Section IV-A1), we finally obtained $T_{\text{bio}} = 13\,234$ words for 7426 authors, $T_{\text{book}} = 18\,738$ words for 205 805 books, and $T_{\text{comment}} = 21\,582$ words for 3 027 502 comments.

c) Histogram calculation: Word histogram of a biography is denoted by $\{H_n^{\text{bio}} = [n_1, n_2, \ldots, n_u, \ldots, n_{T_{\text{bio}}}]\}$, where n_u is the number of times the corresponding word u appears in a biography. The word histogram of a book is described by $\{H_n^{\text{book}} = [n_1, n_2, \ldots, n_v, \ldots, n_{T_{\text{book}}}]\}$, where n_v is the number of times that the term v appears in the book introduction. The word histogram of a book comment is given by $\{H_n^{\text{comment}} = [n_1, n_2, \ldots, n_k, \ldots, n_{T_{\text{comment}}}]\}$, where n_k is the number of times that the word k appears in a comment. The *tf-idf* weighting scheme is then applied to histograms.

d) Compressed PCA features: The histogram vector of each node in an author tree is usually large sized and highly sparse. The total number of nodes in a data set is also large. In order to make the system computationally applicable, we apply the PCA to map each histogram vector into a lower dimensional feature vector [37]. The projected features of author biography, book, and comment, i.e., F_h^{bio}, F_h^{book}, and F_h^{comment}, are separately calculated. The resulting projected features will be used for representing the node features at different levels in an author tree.

3) Generating Author Vectors: After feature extraction, each node in an author tree is encoded by projected PCA features. We can utilize the same process, shown in Section II-C, to generate a vector for each author. During the k-means clustering process utilized for node allocation, we employed the *cosine*-type function to calculate the distance between two nodes a and b in the form of

$$s_{a,b}^{\text{NN}} = w^A \left(1 - \frac{F_{l,a}^G \cdot F_{l,b}^G}{\left\|F_{l,a}^G\right\|_2 \cdot \left\|F_{l,b}^G\right\|_2}\right) + (1 - w^A)\left(1 - \frac{F_{l,a}^L \cdot F_{l,b}^L}{\left\|F_{l,a}^L\right\|_2 \cdot \left\|F_{l,b}^L\right\|_2}\right) \tag{13}$$

where $F_{l,\cdot}^G$ represents the original node features, i.e., $F_{l,\cdot}^G = \{f_{i,l,\cdot}^G\}_{i=1}^{d_l}$, $F_{l,\cdot}^L$ denotes the local features learned from children nodes, i.e., $F_{l,\cdot}^L = \{f_{j,l,\cdot}^L\}_{j=1}^{d_l}$, and w^A is a weight parameter. The first part of (13) calculates the global distance using the node feature, and the second part calculates the local distance with the features learned from child nodes (see Section II-C). In the bottom-level nodes, only the first part of (13) is used. The selection of (13) to measure the distance between two nodes is based on the fact that cosine-type function usually performs well for document data [22], [23], [26]. The weight parameter w^A is used to balance those two parts. It puts relative emphasis between the global and local distance measures. A larger value of w^A, $w^A > 0.5$, implies that emphasis is placed on a parent node's features on its own, and a smaller value of w^A, $w^A < 0.5$, suggests that the local distance from the child nodes' features is emphasized. The effect of this parameter on results is discussed in our experiment.

After k-means clustering, we used the similar distance function shown in (13) to calculate the distance between each child node and its assigned cluster's center (see Section II-C1). A weight parameter w^A is also used for balancing global and local features. Finally, each tree-structured author under our Tree2Vector framework is represented by a vectorial representation.

4) Recommendation: A reader (or user) is most likely to read or purchase the writings of preferred authors; according to these behaviorial data, an online shop can suggest the user reading or buying other authors' books, which are most relevant to the user's favorite authors. Despite many factors influencing the relationship between two authors, biography, book contents, and associated user comments are useful features that can be used for measuring the relevance between two authors. This is because biographies provide a brief description of an author's life, whereas book contents and comments received from readers highlight an author's writings. In this paper, the tree-structured author representation appears to be a combination of biography, books, and comments. In order to formulate a recommendation list, we used the similar distance form shown in (13) to compare a query author with the ones in the database. A weight parameter w^A is also utilized for balancing global and local features.

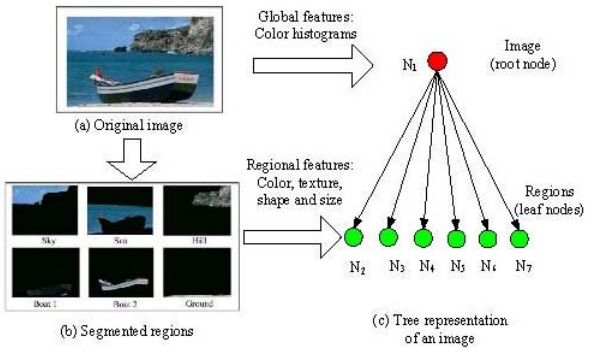

Fig. 6. Image representation by a two-level tree.

B. Content-Based Image Retrieval

Given a query image, CBIR enables us to retrieve relevant images from an image database using semantic similarity between imagery contents [17]. Image features used for measuring the relevance include colors, textures, and shapes. The past decades have witnessed immense attention paid to CBIR in applications of entertainment, military, education, and biomedicine [10]–[17]. In line with the proposed CBIR system [17], we used the similar tree structure to represent each image and took CBIR as another application example of our proposed Tree2Vector framework.

1) Image Tree: Feature extraction, an essential step for analysis of image contents, usually involves learning statistical information of raw image pixels. The resulting feature vector is often small sized in comparison to the number of raw pixels. Besides relevance based on low-level global features such as color histograms, image features should also be empowered by high-level semantics perceived by human. Previous studies [12]–[16] have suggested that region-based methods can offer better solutions to image understanding. As an application example of our Tree2Vector framework within the context of image analysis, we employed a two-level tree structure for image representation reported in [17]. In particular, the JSEG method [38] was utilized for image segmentation in order to extract regional features. Fig. 6 shows the formulation process of a two-level tree. The root node represents the entire image, and child nodes indicate the regions segmented from the original image. Color histograms delivering the color distributions over the entire image are used for global features in order to evaluate in-between visual similarities. For regional image features, color moments and texture are adopted, as regions are segmented according to their uniqueness of these properties. As shown in Fig. 6, we assign color histograms to the root node as global features and assign local region-based features, such as color moment, texture, size, and shape, for child nodes. As a result, a two-level tree is formulated for organizing global and local image features.

2) Detailed Extraction of Image Features: For completion, we briefly summarize the feature extraction process introduced in [17]. First, the hue saturation value (HSV) color space was adopted for representing global features. Sixteen bins were used to compute the histograms for each of the three HSV channels. We calculated the histogram of the hue channel in the form of

$$h_{H_i} = \frac{n_{H_i}}{n_I}, \quad (i = 1, 2, \ldots, 16) \tag{14}$$

where H_i is the ith quantized level of the hue channel, n_{H_i} denotes the number of pixels with their hue values at that level, and n_I indicates the total number of pixels in the image. The features in the saturation (h_{S_i}) and value (h_{V_i}) channels were obtained similarly. At last, the feature vector at the root node of the tree can be described by

$$F_1^G = [h_{H_1}, \ldots, h_{H_{16}}, h_{S_1}, \ldots, h_{S_{16}}, h_{V_1}, \ldots, h_{V_{16}}]^T. \tag{15}$$

Since an image region may contain only one or few colors, color moments were adopted for representing regions instead of a color histogram [11], [38]. The first- and the second-order color moments in the HSV color space are calculated in the form of

$$C^L = [\mu_{c_1}, \mu_{c_2}, \mu_{c_3}, \sigma_{c_1}, \sigma_{c_2}, \sigma_{c_3}] \tag{16}$$

where $\mu_{c_1}, \mu_{c_2}, and\, \mu_{c_3}$ are the means of the three channels of a region, and $\sigma_{c_1}, \sigma_{c_2}, \sigma_{c_3}$ are their corresponding standard deviations.

Three shape features, which are normalized inertia with orders from 1 to 3, were used for capturing the shape property of a region. Given a region M in the Euclidean space $\Re^2$, the normalized inertia with order α can be calculated by

$$I_{\alpha,M}^L = \frac{\sum_{(x,y)\in M} [(x-\bar{x})^2 + (y-\bar{y})^2]^{\alpha/2}}{(V_M)^{1+\alpha/2}} \tag{17}$$

where (x, y) is a point in region M, $(\bar{x}, \bar{y})$ is the centroid of the region, α is the order, and V_M is the number of pixels in the region. The three shape features are denoted by

$$S^L = \left[\frac{I_{1,M}}{I_1^o}, \frac{I_{2,M}}{I_2^o}, \frac{I_{3,M}}{I_3^o}\right] \tag{18}$$

where I_1^o, I_2^o, and I_3^o are the minimum normalized inertia produced by a circular area.

Given a region M in image I, the size feature can be simply calculated by

$$A^L = \frac{V_M}{V_I} \tag{19}$$

where V_I denotes the total number of pixels in the image.

A previous study on region-based image retrieval has shown that it is possible to use wavelet coefficients as features and the moments of wavelet coefficients in high-frequency bands are effective for texture representation [17]. Moments of wavelet coefficients in high-frequency bands were used as texture features by employing the Daubechies-4 wavelet transform [17]. After a one-level wavelet transform of the intensity images, a 4×4 image block can be decomposed into four frequency bands, i.e., the LL, LH, HL, and HH bands, each of which consists of 2×2 coefficients. Suppose $(a_1, a_2, a_3, and a_4)$ represents the coefficients in the LH band, the texture feature of this block in the LH band is defined by

$$t_1 = \left[(a_1^2 + a_2^2 + a_3^2 + a_4^2)/4\right]^{1/2}. \tag{20}$$

The other two texture features t_2 and t_3 generated from the HL and HH bands, respectively, are calculated similarly. We used the means and standard deviations of the LH, HL, and HH bands of all blocks in the image as texture features in the form of

$$T^L = [\mu_{t_1}, \mu_{t_2}, \mu_{t_3}, \sigma_{t_1}, \sigma_{t_2}, \sigma_{t_3}]. \tag{21}$$

Finally, the feature vector for representing the kth segmented region (or leaf node in the image tree) is given by

$$F^G_{k,2} = [\, C^L \quad S^L \quad A^L \quad T^L \,]^T \tag{22}$$

which is a 16-D feature vector.

3) Generating Image Vectors: After feature extraction, the root node in a two-level image tree is encoded by a feature vector F^G_1 shown in (15), and each leaf node is represented by a vector, $F^G_{k,2}$. Subsequently, we can use the same process mentioned in Section II-C to generate one vector for each image. In the k-means clustering process, only node features at the low level are used for computing the distance between two nodes a and b in the form of

$$s^{\mathrm{NN}}_{a,b} = \sum_{u=1}^{16} \left| F^G_{u,a,2} - F^G_{u,b,2} \right| \tag{23}$$

where $F^G_{u,a,2}$ and $F^G_{u,b,2}$ represent the uth components of the feature vectors $F^G_{a,2}$ and $F^G_{b,2}$ of nodes a and b, respectively. After k-means clustering, we used the similar distance function shown in (23) to calculate the distance between each child node and its assigned cluster's center (see Section II-C1). The feature dimension of each segmented region is different with the size of feature vector of the whole image (the global feature). Each region is represented by a 16-D feature vector, whereas the whole image (the root node) is encoded by a 48-D feature vector in the HSV color space. In order to calculate the complementary reconstruction coefficient $\hat{\beta}$ (see Section II-C3), we used the segmented regions to reconstruct the whole image in the three HSV channels separately. We then utilize the averaged coefficient over these three reconstruction coefficients obtained from the three channels as the local weights in the vector generation process.

4) Image Retrieval: It is easy to conduct image retrieval once each tree-structured image is represented by a vector. In the retrieval process, the distance between a query image q and a source image p in the database is calculated by

$$s_{q,p} = w^I \sum_{m=1}^{48} \left| f^G_{m,q} - f^G_{m,p} \right| + \frac{(1-w^I)}{\sum \pi\left(f^L_q\right)} \sum_{n=1}^{48} \pi\left(f^L_{n,q}\right) \cdot \left| f^L_{n,q} - f^L_{n,p} \right| \tag{24}$$

where $f^G_{m,\cdot}$ is the mth element in the global feature vector for image q or p, $f^L_{n,\cdot}$ is the nth element in the learned local feature vector for image q or p, w^I is a weight parameter, and $\pi(f^L_{n,q})$ is defined by

$$\pi\left(f^L_{n,q}\right) = \begin{cases} 1, & \text{if } f^L_{n,q} \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

The first part of (24) represents the distance measure in the global feature space, while the distance measure in the local region-based feature space is calculated in the second part of (24). w^I is used for controlling the weight between the global and local distance.

IV. Experiment

A. Author Recommendation

1) Data Set: A real-life data set was compiled from Amazon.com. At first, a full list of author names and their homepage links was parsed from Amazon. The author biography, the number of written books associated with his or her links, and the names of related authors were extracted from the author's homepage. Subsequenctly, book introduction, comments, and additional author biography can be parsed from each book homepage. In this paper, the authors on the condition that the number of their written books is larger than five were selected for experiment. In addition, we only selected the top 50 comments if the number of book comments is larger than 50. Finally, a data set which contains 7426 authors, 205 805 books, and 3 027 502 comments was compiled for experiment.

2) Evaluation Measure: In order to evaluate the performance, we adopted three metrics, which have been commonly used in recommenders [1], [29], [39]:

1) mean reciprocal rank (MRR), measuring where in the ranking list the first relevant author is given by the recommender, averaged over all the test authors;
2) success at rank k (S@k), defined as the probability of searching a relevant author among the top k returned authors;
3) normalized discounted cumulative gain at rank k (NDCG@k), measuring the gain of an author with respect to its position in the ranking list.

The detailed implementation of these measures can be found in [1], [29], and [39].

Given a query author, recommender can return an author list including the authors most relevant to the query. In Amazon, each author is related to approximately ten authors identified by the title "Customers Also Bought Items By." So this labeled information can be utilized for objective evaluation. For each test author, we built a random pool with size 11 (one positive +ten negative) to evaluate different recommenders. The aforesaid evaluation measures were used for quantifying the performance. Essentially, the larger the values these measures deliver, the better performs the algorithm.

3) Comparative Study: In this section, we evaluate the performance of our proposed method, Tree2Vector. The PCA projection dimension sizes of author biographies, books, and comments were all set at 150 (see Section III-A2). λ in (1) was set at 0.5 and the weight parameter w^A in (13) was set at 0.5. In this paper, these settings of the above parameters were observed to be a good choice. We also conducted an empirical study on these parameters in Section IV-A.4. Furthermore, we tested another four algorithms:

1) Tree2vector-Global, which only relies on the global distance in Tree2Vector (i.e., the weight parameter $w^A = 1$);

TABLE I

COMPARATIVE RESULTS FOR AUTHOR RECOMMENDATIONS

Method	MRR	S@1	S@5	NDCG@5	NDCG@11
Tree2Vector	79.43	67.37	95.94	83.76	85.07
Tree2Vector-Global	70.43	56.58	88.94	74.45	78.05
Tree2Vector-Local	40.77	22.41	62.04	42.56	54.81
Tree2Vector-Clustering	78.02	64.85	95.94	82.68	84.00
Tree2Vector- Clustering -Local	51.92	32.35	80.25	57.34	63.84
MLSOM	76.41	64.57	93.84	80.75	82.73
MLSOM-Global	70.43	56.58	88.94	74.45	78.05
MLSOM-Local	33.64	14.29	53.78	34.58	49.26
LDA	52.04	36.27	70.31	54.14	63.57
PLSI	63.05	47.20	83.75	66.97	72.29
LSI	70.10	56.16	89.08	74.26	77.79
VSM	60.37	44.40	81.79	64.19	70.20

2) Tree2vector-Local, which uses the local distance in Tree2Vector (i.e., the weight parameter $w^A = 0$);
3) Tree2vector-Clustering, which only considers the global weights generated from the clustering process [see Section II-C1 and (7)];
4) Tree2vector-Clustering-Local, which uses the local distance in Tree2Vector-Clustering.

In fact, Tree2Vector-Clustering-Global (which relies on the global distance in Tree2Vector-Clustering) is equal to Tree2Vector-Global. For comparison, we compared Tree2Vector-related methods with state-of-the-art algorithms, including LDA [25], PLSI [24], LSI [23], VSM [22], and MLSOM [1]. LDA [25], PLSI [24], LSI [23], and VSM [22] are representatives of bag-of-words methods in document modeling, which consider the root node features. In line with Tree2Vector, the resulting dimension sizes of features used in these bag-of-words models were set at 150. MLSOM [1], newly investigated by authors, is a method that considers term spatial distributions over tree-structured data. We also tested another two versions of MLSOM: MLSOM-Global and MLSOM-Local, which utilize the global and local distance in MLSOM, respectively. (Implementation details can be found in [1].) Both Tree2Vector-Clustering and MLSOM used the optimal weight for balancing the global and local distance. All of the experiments were performed on a PC with Intel Xeon CPU X3430@ 2.40 GHz and 8-GB memory. The feature extraction programs were written in the Java programming language. The Tree2Vector programs were tested on MATLAB 7.12.0 (R2011a). We held out 90% of the data corpora as a candidate set and 10% as a test set utilized for queries. We performed tenfold cross validation, and the results were averaged over each query, then over the tenfold.

The numerically comparative results are listed in Table I. The results of MLSOM and Tree2Vector-Clustering are based on the optimal weights setting at 0.6 and 0.5, respectively. We have compared our method, Tree2Vector, with other methods in terms of a paired t-test at $p = 0.05$ significance level to judge whether the results of the Tree2Vector differ from the other algorithms in a statistical method. The p-values computed for t-test based on outcomes of Table I are listed in Table II. In Table II, the p-values that are less than 0.05 indicate a sufficient evidence against the null hypothesis. We also include the results of average NDCG@k visually shown in Fig. 7 for k ranging from 1 to 11. From Table I, MLSOM-Global (which considers the global distance in MLSOM) delivers similar results to

TABLE II

p-VALUES COMPUTED FOR PAIRED-SAMPLE t-TEST AGAINST TREE2VECTOR FOR AUTHOR RECOMMENDATIONS (THE RESULTS UNDER $p \geq 0.05$ HAVE BEEN UNDERLINED)

Method	MRR	S@1	S@5	NDCG@5	NDCG@11
Tree2Vector	—	—	—	—	—
Tree2Vector-Global	5.7435e-17	9.0395e-11	5.9348e-11	3.8465e-20	1.1495e-17
Tree2Vector-Local	1.1161e-87	1.3850e-68	3.3726e-55	4.7100e-91	4.4500e-89
Tree2Vector-Clustering	1.5055e-14	6.0453e-09	5.5735e-11	6.7075e-17	3.7781e-15
Tree2Vector- Clustering -Local	7.1459e-85	4.8974e-65	5.3684e-55	2.8953e-89	3.4979e-86
MLSOM	0.0016	0.0587	0.0221	0.0005	0.0012
MLSOM-Global	5.7435e-17	9.0395e-11	5.9348e-11	3.8465e-20	1.1495e-17
MLSOM-Local	8.0687e-125	1.3732e-97	9.4714e-76	1.0464e-126	6.4110e-126
LDA	4.8102e-60	9.9326e-43	3.7082e-42	3.3194e-65	1.1024e-61
PLSI	3.5606e-30	2.0541e-21	3.0908e-18	5.2166e-35	2.3209e-31
LSI	1.1283e-17	1.8305e-11	1.0044e-10	1.0792e-20	2.2975e-18
VSM	4.3787e-41	4.9370e-29	1.3451e-21	9.0407e-45	3.1381e-42

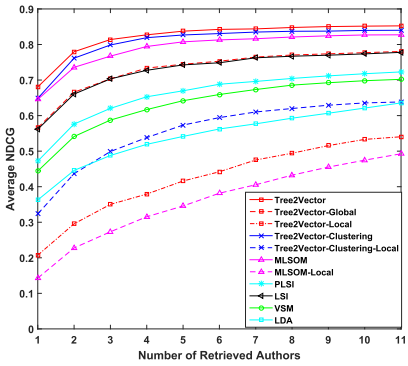

Fig. 7. Comparative results for author recommendation.

Tree2Vector-Global. So we did not plot the result of MLSOM-Global in Fig. 7. As shown in Table I, Tree2Vector delivers better MRR value than other methods. From Fig. 7, it is observed that Tree2Vector consistently outperforms other methods with respect to average NDCG when the top k authors are recommended. In particular, Tree2Vector can produce over 10% improvement in terms of S@1 over the traditional bag-of-words models such as LDA, PLSI, LSI, and VSM. In addition, Tree2Vector obtains over 3% performance gain on both MRR and NDCG@5 in comparison to MLSOM. It is interesting to note that Tree2Vector-Clustering-Local delivers better results than does Tree2Vector-Local at a significant rate (over 10% improvement). However, Tree2Vector outperforms Tree2Vector-Clustering by integrating the complementary reconstruction coefficient into the local features [see (7)]. We believe that this phenomenon is caused by the fact that Tree2Vector-Local places great emphasis on the local information produced by the locality reconstruction process.

4) Parameter Study: Considering the assumption that the use of combined distances from the global and local may produce performance enhancement, there must be an optimal weight w^A to balance this effect on (13). Fig. 8 shows the MRR, average NDCG, and average success values produced by Tree2Vector against the weight values varying from 0 to 1 at an increment of 0.05. From Fig. 8, we can observe that setting the weight w^A at approximately 0.5 appears to be a good selection for the Tree2Vector algorithm. An appropriate selection of w^A is dependent upon data sets. According to our observation, setting the weight w^A to the range of [0.4, 0.6] makes it more probable to deliver promising results.

The number of PCA projected feature dimensions may also affect the recommendation results of our system. To show this

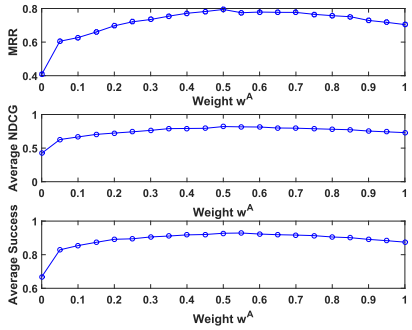

Fig. 8. Results against different weights w^A.

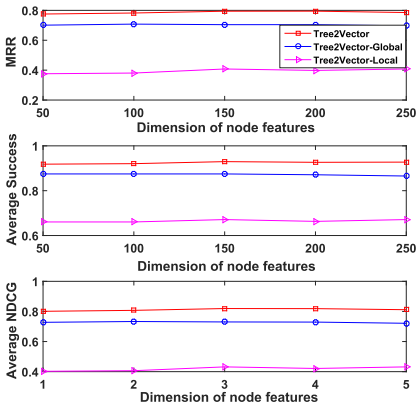

Fig. 9. Results against different dimensions of node features.

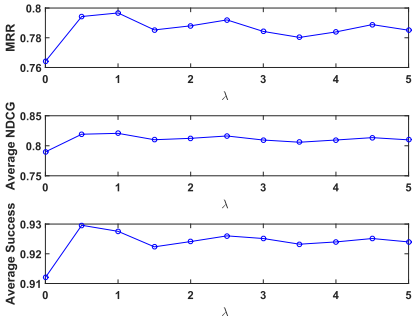

Fig. 10. Results against different settings of λ for author recommendation.

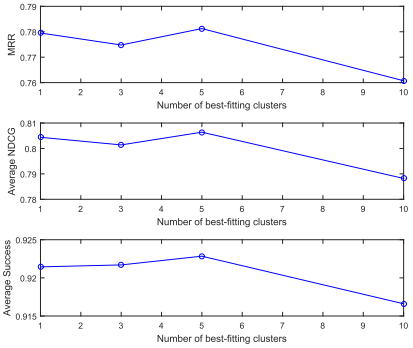

Fig. 11. Results against different settings of numbers of best-fitting clusters in Algorithm 1 for author recommendation.

effect on the Tree2Vector system, we provided the results of Tree2Vector-related methods with different feature dimensions varying from 50 to 250 at an increment of 50, as presented in Fig. 5. The Tree2Vector used the optimal weight. The results are shown in Fig. 9. It is observed that Tree2Vector consistently outperforms Tree2Vector-Global and Tree2Vector-Local. This indicates that integrating the global information based on node features and the local information generated from child features under our framework is, indeed, able to deliver promising performance.

Moreover, we also conducted an empirical study on λ used in the LSR model shown in (1). The results are illustrated in Fig. 10. The Tree2Vector used the optimal weight (i.e., $w^A = 0.5$). The number of PCA projected feature dimensions was set at 150. From Fig. 10, the results produced by Tree2Vector are relatively stable against different settings of λ. This suggests that our method, Tree2Vector, is not sensitive to this parameter.

According to our observation, several nodes belonging to the same parent node might be allocated into a single cluster if fully relying on the output of k-means. Therefore, we developed a heuristic allocation method that selects one of the k_a best-fitting clusters in a first-come first-serve manner. If $k_a = 1$, the allocation relies on the output of k-means solely; if k_a is set at a large integer, the nodes will distribute over different clusters, which performs like a random manner. We have performed an empirical study on different selections of this parameter (i.e., $k_a = 1, 3, 5, 10$). The result is shown in Fig. 11. We can observe that setting $k_a \leq 5$ has slight effect on the author recommendation performance. Considering the results on both author recommendation and image retrieval, k_a was set at 3 across all experiment.

B. Image Retrieval

1) Data Set: For a fair comparison with other methods, the proposed Tree2Vector-based CBIR system was tested on a subset of the COREL image database [11], [17], which consists of 1000 color images. The images were categorized in ten classes, and each class contained 100 images. Before performing feature extraction, each image was resized into approximately 10 000 pixels under their original width–height ratio. We ran our program by using each image of the database as a query.

2) Performance Metrics: To quantify the retrieval results, we used averaged precision and recall values [11], [17] for each query image. In addition, the following measure is known as "area under the precision-recall curve" (AUC) which is related to both of the above two measures:

$$\text{AUC} = \sum_{i_A=2}^{n_{\max}} \frac{(p(i_A) + p(i_A - 1)) \times (R(i_A) - R(i_A - 1))}{2} \tag{26}$$

where $n_{\max}$ denotes the maximum number of retrieved images, and $P(i_A)$ and $R(i_A)$ denote the precision and recall values with i_A images retrieved, respectively.

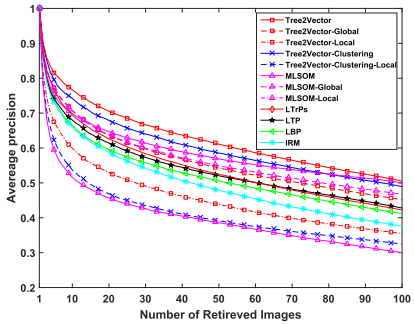

Fig. 12. Comparative results for image retrieval against average precision.

TABLE III
COMPARATIVE RESULTS FOR IMAGE RETRIEVAL

Method	AUC	No. of retrieved images					
		10	50	100	10	50	100
		Precision (%)			Recall (%)		
Tree2Vector	0.3220	76.67	61.08	50.52	7.67	30.54	50.52
Tree2Vector-Global	0.2601	70.74	54.29	45.17	7.07	27.15	45.17
Tree2Vector-Local	0.1687	59.89	43.80	35.47	5.99	21.90	35.47
Tree2Vector-Clustering	0.3002	74.20	58.55	48.93	7.42	29.27	48.93
Tree2Vector-Clustering-Local	0.1383	53.65	39.11	32.42	5.37	19.55	32.47
MLSOM	0.2943	69.82	56.90	49.86	6.98	28.45	49.86
MLSOM-Global	0.2703	70.62	54.96	46.68	7.06	27.48	46.68
MLSOM-Local	0.1251	51.56	38.59	29.99	5.16	19.29	29.99
LTrPs	0.2378	69.42	52.47	42.44	6.94	26.24	42.24
LTP	0.2370	68.01	52.04	42.80	6.80	26.02	42.80
LBP	0.2225	66.47	50.59	41.21	6.65	25.30	41.21
IRM	0.1970	66.07	47.88	37.59	6.61	23.94	37.59

3) Comparative Results: We first evaluate the retrieval performance of the Tree2Vector-related methods based on the above metrics. We empirically set weight w^I and λ at 0.2 and 0.5, respectively. We also included an effect study on these parameters in Section IV-B4. For comparison, we implemented an IRM-based CBIR system [11], which is a straightforward retrieval method. The same JSEG [38] method was utilized for image segmentation in both IRM and our proposed approach. The same shape, color, texture, and size features of regions in Section III-B2 were also used in the IRM-based CBIR system. Furthermore, we compared our method with three typical methods: local binary pattern (LBP) [40], local texture pattern (LTP) [41], and local tetra patterns (LTrPs) [42], which consider flat feature vectors by using different feature extraction methods. MLSOM [17], the most relevant method to ours, is also compared thoroughly with our method. The numerically comparative results of different methods are summarized in Table III, in which the results of Tree2Vector-Clustering and MLSOM are based on the optimal weight. On the basis of the outcomes of Table III, the p-values computed for paired t-test at $p = 0.05$ significance level are presented in Table IV. We also summarize the precision results visually shown in Fig. 12 when the number of retrieved images, the most similar candidate images from the data set for each query, varies from 1 to 100. It is clear to observe that our method, Tree2Vector, consistently outperforms other approaches. In particular, Tree2Vector achieves over 7% precision improvement in comparison to IRM, LBP, LTP, and LTrPs, when the number of retrieved images is smaller than 50. Tree2Vector delivers approximately a 9% improvement of average precision over MLSOM when ten images are retrieved. Tree2Vector can also produce over 2% AUC larger than Tree2Vector-Clustering. This indicates that Tree2Vector with integrating the local weights generated from the LSR model can deliver better results in comparison to

TABLE IV
P-VALUES COMPUTED FOR PAIRED-SAMPLE T-TEST AGAINST TREE2VECTOR FOR IMAGE RETRIEVAL

Method	AUC	No. of retrieved images					
		10	50	100	10	50	100
		Precision			Recall		
Tree2Vector	—	—	—	—	—	—	—
Tree2Vector-Global	5.4513e-60	1.1446e-20	9.2636e-65	1.4267e-78	1.1446e-20	9.2636e-65	1.4267e-78
Tree2Vector-Local	2.6312e-172	1.1433e-107	1.9443e-154	3.0528e-189	1.1433e-107	1.9443e-154	3.0528e-189
Tree2Vector-Clustering	3.6644e-15	0.0004	1.8894e-17	3.8832e-14	0.0004	1.8894e-17	3.8832e-14
Tree2Vector-Clustering-Local	2.3096e-146	8.0505e-108	6.6327e-148	4.2012e-163	8.0505e-108	6.6327e-148	4.2012e-163
MLSOM	1.1987e-50	2.0313e-61	1.1668e-64	2.2863e-54	2.0313e-61	1.1668e-64	2.2863e-54
MLSOM-Global	7.2937e-34	6.8483e-21	6.0719e-52	4.8213e-41	6.8483e-21	6.0719e-52	4.8213e-41
MLSOM-Local	4.7058e-101	8.3190e-114	6.3969e-117	1.0871e-114	8.3190e-114	6.3969e-117	1.0871e-114
LTrPs	2.2785e-26	5.4817e-12	8.5798e-25	1.5506e-37	5.4817e-12	8.5798e-25	1.5506e-37
LTP	2.0961e-21	8.6391e-17	1.6213e-26	2.7728e-31	8.6391e-17	1.6213e-26	2.7728e-31
LBP	5.8826e-31	3.1868e-22	3.8783e-34	1.4793e-43	3.1868e-22	3.8783e-34	1.4793e-43
IRM	8.8932e-127	4.8793e-39	4.2547e-100	4.2617e-147	4.8793e-39	4.2547e-100	4.2617e-147

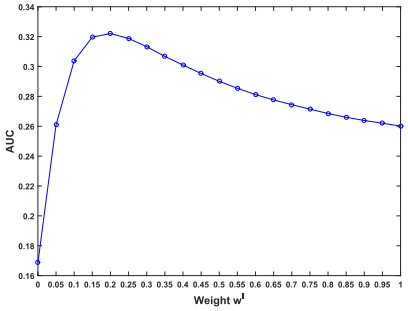

Fig. 13. AUC against different weights w^I for image retrieval.

Tree2Vector-Clustering by using the node allocation process only. However, the optimal weights for Tree2Vector-Clustering and MLSOM are 0.8 and 0.9, respectively, whereas the optimal weight for Tree2Vector appears to be 0.2. This suggests that in both Tree2Vector-Clustering and MLSOM, the features of a parent node play more important role than the local features generated from its child nodes. In contrast, the Tree2Vector can increase performance by placing more emphasis on the local information by using the complementary reconstruction coefficients.

4) Parameter Study: In order to evaluate the effect of weight w^I [see (24)] on the performance of Tree2Vector, we summarized the AUC values produced by Tree2Vector against different weights varying from 0 to 1 at an increment of 0.05, as shown in Fig. 13. We can observe that there is an optimal weight for balancing the global and local information in the Tree2Vector algorithm. For this paper, setting the weight w^I at approximately 0.2 appears to be a good selection for the Tree2Vector.

We also conducted an empirical study on λ used in the LSR model [see (1)]. The results are illustrated in Fig. 14. The Tree2Vector used the optimal weight (i.e., $w^I = 0.2$). From Fig. 14, the results produced by the Tree2Vector are similar when $\lambda \geq 0.5$. This suggests that setting $\lambda \geq 0.5$ is sufficient to obtain promising performance in real-world applications.

Moreover, Fig. 15 illustrates the results using different numbers of best-fitting clusters in the node allocation process as shown in Algorithm 1. We can observe that setting the number of best-fitting clusters at 3 is a better option to the application of image retrieval.

C. Discussion

Currently developing an efficient framework for transforming tree-structured data into vectorial representations

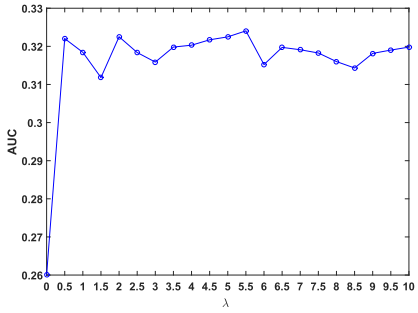

Fig. 14. AUC with different settings of λ for image retrieval.

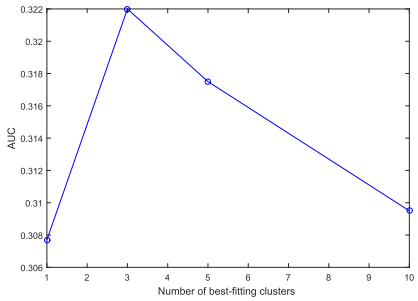

Fig. 15. Results against different settings of numbers of best-fitting clusters in Algorithm 1 for image retrieval.

constitutes very demanding work, because many methods in real-world applications rely on vector space. Although our method, Tree2Vector, was examined in a simulation platform, many interesting results can be observed.

1) *Tree2Vector Versus Nontree Representations:* In document modeling, traditional bag-of-words models [22]–[25] usually transform tree-structured data into single vectors without considering term spatial information over these tree-structured data. As a result, semantics hidden at different levels of trees are overlooked. Researchers mainly focus on designing effective methods [40]–[42] for producing generative features for image representation. Instead, the Tree2Vector works on natural tree-structured data. This enables us to formulate a vectorial representation for each tree while preserving the local information delivered by child nodes.
2) *Tree2Vector Versus MLSOM:* MLSOM [1], [17] is most relevant to our method, Tree2Vector. They both work on tree-structured data. However, the limits of MLSOM include dependence on designing a specific SOM structure through a careful training process and incapability of formulating an independent vector for each tree. To overcome these issues, we designed a new framework, Tree2Vector. The experimental results have shown that it can deliver better performance than does MLSOM.
3) *Tree2Vector Versus Node Features:* Under our Tree2Vector framework, different feature representation schemes for describing each node in a tree can be adopted. For example, instead of PCA, other dimensionality reduction techniques, such as PLSI [24] and LDA [25], and contextual machine learning methods, such as word2vec [44] and paragraph2vec [45], can also be utilized in an author tree. For image representation, the two-level tree structure can be extended in a finer segmentation manner and the features for representing each segmented region can be replaced by deep learning methods [43]. The aim of this paper lies in the design of a flexible tree-to-vector framework. We leave these extensions with respect to node feature representation to future work.
4) *Tree2Vector versus Parameter Settings:* There are two parameters associated with the Tree2Vector algorithm: λ and weight w^* (* indicates $\{A$ or $I\}$). According to our empirical study, the Tree2Vector is able to deliver satisfactory results when $\lambda \geq 0.5$ in the LSR model. For weight w^* used for balancing the global and local features, the setting of this parameter is dependent on data sets in real applications. According to our empirical study, setting the weight w^* at the range of [0.2, 0.6] makes it more probable to accomplish promising results.

V. Conclusion

This paper presents new models and algorithms for transforming tree-structured data into vectorial representations. We developed a node allocation process by employing the clustering technique. This process aims at describing the global discriminative information embedded in the same levels of all trees. An LSR model was introduced to preserve the local information hidden among child nodes for a parent node. The resulting unified framework, Tree2Vector, was examined in two real-world applications: Amazon author recommendation and CBIR. Experimental results have demonstrated the effectiveness of our method. In future work, the node allocation process favored by more advanced clustering techniques deserves further investigation. Moreover, it will be interesting to examine more flexible tree structures for image representation under our framework. Motivated by GNNs [46] and RNNs [47], it is possible to extend our method, Tree2Vector, into supervised versions, which can be explored in future work.

References

[1] H. Zhang, T. W. S. Chow, and Q. M. J. Wu, "Organizing books and authors by multilayer SOM," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2537–2550, Dec. 2016.
[2] R. Yang, P. Kalnis, and A. K. Tung, "Similarity evaluation on tree-structured data," in *Proc. SIGMOD*, 2005, pp. 754–765.
[3] S. M. Selkow, "The tree-to-tree editing problem," *Inf. Process. Lett.*, vol. 6, no. 6, pp. 184–186, 1977.
[4] K. Zhang, "Algorithms for the constrained editing distance between ordered labeled trees and related problems," *Pattern Recognit.*, vol. 28, no. 3, pp. 463–474, 1995.
[5] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM J. Comput.*, vol. 18, no. 6, pp. 1245–1262, 1989.
[6] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava, and T. Yu, "Approximate XML joins," in *Proc. SIGMOD*, 2002, pp. 287–298.
[7] M. Garofalakis and A. Kumar, "Correlating XML data streams using tree-edit distance embeddings," in *Proc. Symp. Principles Database Syst.*, 2003, pp. 143–154.
[8] K. Kailing, H.-P. Kriegel, S. Schönauer, and T. Seidl, "Efficient similarity search for hierarchical data in large databases," in *Proc. Int. Conf. Extending Database Technol.*, 2004, pp. 676–693.

[9] A. Sanfeliu, R. Alquézar, J. Andrade, J. Climent, F. Serratosa, and J. Vergés, "Graph-based representations and techniques for image processing and image analysis," *Pattern Recognit.*, vol. 35, no. 3, pp. 639–650, 2002.

[10] J. R. Smith and C.-S. Li, "Image classification and querying using composite region templates," *Comput. Vis. Image Understand.*, vol. 75, nos. 1–2, pp. 165–174, 1999.

[11] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics-sensitive integrated matching for picture libraries," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 947–963, Sep. 2001.

[12] X. Wu, "Image coding by adaptive tree-structured segmentation," *IEEE Trans. Inf. Theory*, vol. 38, no. 6, pp. 1755–1767, Nov. 1992.

[13] H. Radha, M. Vetterli, and R. Leonardi, "Image compression using binary space partitioning trees," *IEEE Trans. Image Process.*, vol. 5, no. 12, pp. 1610–1624, Dec. 1996.

[14] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Trans. Image Process.*, vol. 9, no. 4, pp. 561–576, Apr. 2000.

[15] S.-Y. Cho, Z. Chi, W.-C. Siu, and A. C. Tsoi, "An improved algorithm for learning long-term dependency problems in adaptive processing of data structures," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 781–793, Jul. 2003.

[16] S.-Y. Cho and Z. Chi, "Genetic evolution processing of data structures for image classification," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 2, pp. 216–231, Feb. 2005.

[17] T. W. S. Chow, M. K. M. Rahman, and S. Wu, "Content-based image retrieval by using tree-structured features and multi-layer self-organizing map," *Pattern Anal. Appl.*, vol. 9, no. 1, pp. 1–20, 2006.

[18] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer-Verlag, 1997.

[19] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2169–2178.

[20] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 1794–1801.

[21] S. Chen and Y. L. Tian, "Pyramid of spatial relatons for scene-level land use classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 1947–1957, Apr. 2015.

[22] G. Salton, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1983.

[23] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[24] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. Conf. Uncertainty Artif. Intell.*, 1999, pp. 289–296.

[25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[26] T. W. S. Chow and M. K. M. Rahman, "Multilayer SOM with tree-structured data for efficient document retrieval and plagiarism detection," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1385–1402, Sep. 2009.

[27] H. Zhang and T. W. S. Chow, "A coarse-to-fine framework to efficiently thwart plagiarism," *Pattern Recognit.*, vol. 44, no. 2, pp. 471–487, 2011.

[28] H. Zhang and T. W. S. Chow, "A multi-level matching method with hybrid similarity for document retrieva," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2710–2719, 2012.

[29] H. Zhang, Y. Ji, J. Li, and Y. Ye, "A triple wing harmonium model for movie recommendation," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 231–239, Feb. 2016.

[30] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[31] X. Huang, Y. Ye, and H. Zhang, "Extensions of kmeans-type algorithms: A new clustering framework by integrating intracluster compactness and intercluster separation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 8, pp. 1433–1446, Aug. 2014.

[32] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.

[33] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 3360–3367.

[34] C.-M. Huang, S. Chen, M. Cheng, and Y.-C. F. Wang, "A sparse-coding based approach to clothing image retrieval," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Nov. 2012, pp. 314–318.

[35] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proc. 5th ACM Conf. Digit. Libraries*, San Antonio, TX, USA, 2000, pp. 195–204.

[36] T. Reichling and V. Wulf, "Expert recommender systems in practice: Evaluating semi-automatic profile generation," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Boston, MA, USA, 2009, pp. 59–68.

[37] L. Lu and H. Zhang, "A tree-structured representation for book author and its recommendation using multilayer SOM," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[38] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 800–810, Aug. 2001.

[39] B. Sigurbjörnsson and R. Van Zwol, "Flickr tag recommendation based on collective knowledge," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 327–336.

[40] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[41] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1635–1650, Jun. 2010.

[42] S. Murala, R. P. Maheshwari, and R. Balasubramanian, "Local tetra patterns: A new feature descriptor for content-based image retrieval," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2874–2886, May 2012.

[43] H. Zhang, X. Cao, J. K. L. Ho, and T. W. S. Chow, "Object-level video advertising: An optimization framework," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 520–531, Apr. 2017.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781

[45] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 1–9.

[46] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[47] M. Bianchini, M. Gori, and F. Scarselli, "Processing directed acyclic graphs with recursive neural networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1464–1470, Nov. 2001.

[48] M. Collins and N. Duffy, "Convolution kernels for natural language," in *Proc. NIPS*, 2002, pp. 1–8.

[49] J. Suzuki, Y. Sasaki, and E. Maeda, "Kernels for structured natural language data," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 2003, pp. 1–8.

[50] J. Suzuki, H. Isozaki, and E. Maeda, "Convolution kernels with feature selection for natural language processing tasks," in *Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics*, 2004, pp. 119–126.

[51] A. Buja, D. F. Swayne, M. L. Littman, N. Dean, H. Hofmann, and L. Chen, "Data visualization with multidimensional scaling," *J. Comput. Graph. Statist.*, vol. 17, no. 2, pp. 444–472, 2012.

[52] Y. M. M. Bishop *et al.*, "Book review: Discrete multivariate analysis: Theory and practice," *Appl. Psychol. Meas.*, vol. 1, no. 2, pp. 297–306, 1977.

[53] L. J. P. van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

Haijun Zhang (M'13) received the B.Eng. and master's degrees from Northeastern University, Shenyang, China, in 2004 and 2007, respectively, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010.

From 2010 to 2011, he was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. Since 2012, he has been with the Shenzhen Graduate School of Harbin Institute of Technology, Shenzhen, China, where he is currently an Associate Professor of computer science. His current research interests include multimedia data mining, machine learning, and computational advertising.

Dr. Zhang is currently an Associate Editor of *Neurocomputing*, *Neural Computing and Applications*, and *Pattern Analysis and Applications*.

Shuang Wang received the B.Sc. degree from the Department of Mathematics, Northeastern University at Qinhuangdao, Qinhuangdao, China, in 2015. She is currently pursuing the Ph.D. degree with the Department of Computer Science, Shenzhen Graduate School of Harbin Institute of Technology, Shenzhen, China.

Her current research interests include tree-structured data processing and text mining.

Xiaofei Xu received the B.S., M.S., and Ph.D. degrees from the Department of Computer Science and Engineering, Harbin Institute of Technology (HIT), Shenzhen, China, in 1982, 1985, and 1988, respectively.

He is currently the Vice President of HIT. His current research interests include service computing and engineering, effective radiated power and supply management systems, databases and data mining, knowledge management, and software engineering.

Tommy W. S. Chow (M'94–SM'03) received the B.Sc. degree (First Class Hons.) and the Ph.D. degree from the Department of Electrical and Electronic Engineering, University of Sunderland, Sunderland, U.K.

He has been involved on different consultancy projects with the Mass Transit Railway, Kowloon-Canton Railway Corporation, Hong Kong. He has also conducted other collaborative projects on the application of neural networks for machine fault detection and forecasting with the Hong Kong Electric Co. Ltd, Hong Kong, the MTR Hong Kong, Hong Kong, and Observatory Hong Kong, Hong Kong. He is currently a Professor with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. He has authored or coauthored over 170 journal articles related to his research, five book chapters, and one book. His current research interests include neural networks, machine learning, pattern recognition, and documents analysis and recognition.

Dr. Chow was a recipient of the Best Paper Award in the 2002 IEEE Industrial Electronics Society Annual Meeting in Seville, Spain.

Q. M. Jonathan Wu (M'92–SM'09) received the Ph.D. degree in electrical engineering from the University of Wales, Swansea, U.K., in 1990.

He was with the National Research Council of Canada for ten years from 1995, where he became a Senior Research Officer and a Group Leader. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. He has authored over 250 peer-reviewed papers in computer vision, image processing, intelligent systems, robotics, and integrated microsystems. His current research interests include 3-D computer vision, active video object tracking and extraction, interactive multimedia, sensor analysis and fusion, and visual sensor networks.

Dr. Wu holds the Tier 1 Canada Research Chair in Automotive Sensors and Information Systems. He was an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART A, and the *International Journal of Robotics and Automation*. He has served on technical program committees and international advisory committees for many prestigious conferences.