Organizing Books and Authors by Multilayer SOM

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

***Abstract*— This paper introduces a new framework for the organization of electronic books (e-books) and their corresponding authors using a multilayer self-organizing map (MLSOM). An author is modeled by a rich tree-structured representation, and an MLSOM-based system is used as an efficient solution to the organizational problem of structured data. The tree-structured representation formulates author features in a hierarchy of author biography, books, pages, and paragraphs. To efficiently tackle the tree-structured representation, we used an MLSOM algorithm that serves as a clustering technique to handle e-books and their corresponding authors. A book and author recommender system is then implemented using the proposed framework. The effectiveness of our approach was examined in a large-scale data set containing 3868 authors along with the 10 500 e-books that they wrote. We also provided visualization results of MLSOM for revealing the relevance patterns hidden from presented author clusters. The experimental results corroborate that the proposed method outperforms other content-based models (e.g., rate adapting poisson, latent Dirichlet allocation, probabilistic latent semantic indexing, and so on) and offers a promising solution to book recommendation, author recommendation, and visualization.**

***Index Terms*— Author recommendation, book recommendation, content-based recommendation, self-organizing map (SOM), tree structure.**

I. Introduction

SINCE the handheld electronic book (e-book) reader device was first launched in 2007 [1], the way people read books has been undergoing a major change. Current e-readers have the ability to store thousands of books, and are able to receive book promotions via Wireless Fidelity from book retailers; browsing of books in libraries and bookshops can be replaced in some way by browsing e-readers. This evolution has been so huge that we are witnessing hard print book copies suffer from the most significant market drop, since the printing press was invented by Guttenburg. In 2012, the book retail giant, Amazon, announced that its e-book retail has overtaken its conventional hard print sales; for every 100 hard print books, they sold, 114 e-books were downloaded to Amazon e-book readers [2]. The way that people read may have changed, but the experience that people obtain from reading has remained the same. We still see fellow commuters reading e-books using their e-readers or laptops around college campuses, coffee shops, and on public transportation. Using the Internet, e-readers can find all kinds of literature, ranging from the best sellers by established authors, to cutting-edge material by daring new authors, and everything in between. As a result, book and author recommendations have become essential to book sellers, book readers, and authors.

The widely employed techniques of recommender systems work on collaborative filtering [3], [4] and content-based recommendations [5]. Collaborative filtering relies on users' own explicit and implicit preferences, the preferences of other users, and the attributes of users and items. It assumes that a given user's preferences are similar to another user of the system and that a sufficient number of user ratings are available. There are three issues that stand out as problematic in collaborative filtering. First, items that have not been rated by a sufficient number of users cannot be effectively recommended. Second, a collaborative approach is not able to recommend items that no one has yet rated or purchased; this is the so-called cold-start problem. Third, statistics on library use show that most books are utilized by very few patrons [6]. This leads to a sparsity problem with respect to the user rated or historical purchase data. Content-based recommendations, on the other hand, can help to overcome these issues by inferring similarities between existing and new users, as well as between existing and new items [7]. It recommends items based on content features about the item itself rather than on the preferences of other users. A content-based approach was employed in one of the first book recommendation systems [8], [9]. However, its system developer had to laboriously hand label each book with values for a preselected set of features, and users had to provide specific traits about themselves in addition to evaluating recommended books. Other work to explore content-based book recommendation was proposed to apply automated text-categorization methods to semistructured text extracted from the Internet [10]. However, the content information considered in the proposed system only consisted of textual metadata rather than the actual text of the books themselves. Each book was represented as a hybrid bags-of-words vector. The employed inductive learner was a simple Bayesian classifier to handle the vector of the bags. In the industry, e.g., Google Books, full-text indexing has widely been used for book retrieval via search queries.

Apparently, when considering the full text of books, it is straightforward to use the traditional bag-of-words model to represent each book as a document vector. The similarities

Manuscript received June 10, 2014; revised October 23, 2015; accepted October 25, 2015. Date of publication November 13, 2015; date of current version November 15, 2016. This work was supported in part by the Shenzhen Foundation Research Fund under Grant JCYJ20150625142543464 and in part by the National Natural Science Foundation of China under Grant 61300209 and Grant 61572156.

H. Zhang is with the Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: hjzhang@hitsz.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.2015.2496281

between books can be measured by the widely used cosine distance. However, a book is usually a lengthy document, which may cover most of the vocabulary that we use in our daily lives. Therefore, in order to have a better understanding of the semantics hidden within a book, capturing the term spatial information in the book is critical. Traditional document modeling methods, however, rely on the bag-of-words models, such as the vector space model (VSM) [11], the latent semantic indexing (LSI) [12], the probabilistic LSI (PLSI) [13], the latent Dirichlet allocation (LDA) [14], and the rate adapting poisson (RAP) model [38]. These methods mainly consider the term frequency (*tf*) only. They use flat feature representation through formulating a function of *tf*. Apparently, this type of representation is only a document-level description, because two documents containing similar term frequencies can totally be contextually different when the spatial distribution of the terms is different. For example, school, computer, and science mean very different terms when they appear in different parts of a document when compared with the case of a school of computer science that strings the words together. Thus, solely relying on the *tf* information from the bag-of-words model is not a reliable way to discriminate contextual similarity, because it can be rather inaccurate when not considering the *tf*, word interconnections and spatial distributions together for the whole book.

Document analysis considering term associations has been studied by a few researchers. Graph matching-based methods exist in the Web documents analysis [16]. This type of approach is capable of delivering respectable accuracy particular when the documents are of small size. The graph matching process, however, is rather computationally complex, which is impractical when one requires to compare two e-books that consist of a few hundreds of pages. Moreover, Fuketa *et al.* [17] introduced a field understanding method using field association words for document classification. Others used either bigrams [18] or term association rules [19] to improve the classification accuracy. In [21], we focused on the multiple features (MFs) extraction schemes using different word graphs. In later studies, a dual wing harmonium model was developed to generate the latent representations of documents by modeling the MFs [15]. These methods, however, only work on short documents, because extracting appropriate field association words [17], bigrams [18], term association rules [19], or Term Connection Frequency [15], [21] from e-books requires large extra computational cost due to prominent term spatial distributions within e-books. Meanwhile, a flexible multilayer self-organizing map (MLSOM) [22] was developed for handling unsupervised learning of generic tree-structured data. More recently, it has been demonstrated that MLSOM can be effectively used in document retrieval and plagiarism detection [20].

Author recommendation is another important application for online stores in e-book market. It helps people to make the right choice of favorite authors together with their loved books. With the collaborative filtering technique, Heck *et al.* [23] developed an academic author recommendation system by considering cocitation and bibliographic coupling. Recently, Vaz *et al.* [24] introduced two item-based collaborative filtering algorithms to predict books and authors that a user will like. These two algorithms used only the rating features of books and authors. They did not examine the book content and author information (e.g., biography) at all. In practice, many expert recommenders have been designed mainly for business institutions. For example, Petry *et al.* [25] developed an expert recommendation system, Intelligent Context Awareness for Recommending Experts (ICARE), which was used for recommending experts in an organization. ICARE focused on author's organizational level, availability, and reputation, instead of author's publications and citations. In later study, Reichling and Wulf [26] implemented a recommender system for a European industrial association. Experts were defined according to their collection of documents. However, these approaches still have major shortcomings on privacy and data security in the Internet [23].

Inspired by the tree-structure representation [20], in this paper, we developed a framework for the organization of e-books and authors using a four-level tree representation and MLSOM. The information of authors is hierarchically organized into different layers of an MLSOM: 1) paragraphs are organized at the bottom layer; 2) pages are organized at the third layer; 3) e-books are organized at the second layer; and 4) authors are organized at the top layer. Thus, author information can be fully described by such a hierarchical tree. Different layers play different roles: the top layer performs author clustering, the second layer works on clustering e-books, and the other two bottom layers are used to compress local features of books. MLSOM is used as an efficient solution to the clustering problem of these structured data. Under the proposed framework, we implemented a book and author recommender system. Due to a lack of public e-book data sets for the purpose of book and author recommendation, we compiled a large-scale data set containing 3868 authors and 10 500 e-books. Our experimental results show that the proposed approach outperforms other content-based models, such as RAP [38], LDA [14], PLSI [13], LSI [12], and VSM [11], and can be applicable for book and author recommendation. It is worth pointing out that collaborative filtering methods, as reported in [24], can be incorporated into our proposed framework to predict a user's preference for books and authors in online applications. In addition, popular e-readers work with very different formats of e-books. In the industry, e-book providers can easily employ our proposed methodology in their recommending system, but they may need to adjust the hierarchy of the book representation according to their manufactured book formats.

The rest sections of this paper are organized as follows. Section II presents the details of tree-structured author feature representation. Section III presents the framework of organizing books and authors using MLSOM. Book and author recommender systems are implemented in Section IV. Extensive experiments are conducted in Section V. Finally, the conclusion is drawn in Section VI.

II. Tree-Structured Author Representation

In this section, we first introduce the overall framework to construct an author tree. The implementation details, such as

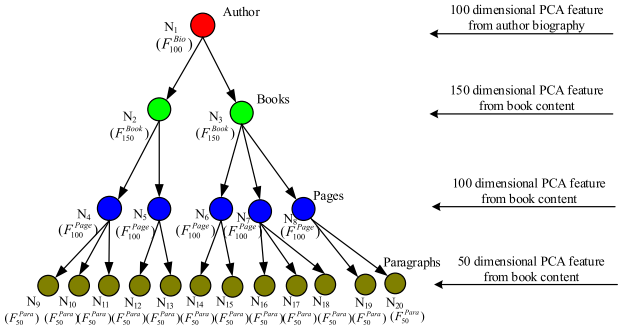


Fig. 1. Author representation by tree-structured feature.

preprocessing, vocabulary construction, word histogram formation, and feature projection, are then described.

A. Tree Structure

To fully describe the information of an author, we use a tree structure to represent the feature of the author, as shown in Fig. 1. The root node at the first level indicates the biography of an author. It may include the birth date, nationality, education history, political activities, and the early childhood and the later life experiences of an author. The second-level nodes represent books that the author has written. The third-level nodes represent different pages that are partitioned from the book. The bottom-level nodes represent paragraphs of the pages. Thus, an author description is constructed as author $\rightarrow$ books $\rightarrow$ pages $\rightarrow$ paragraphs tree. This is a natural way of generating a tree structure. It can be further improved using a form of author $\rightarrow$ books $\rightarrow$ chapters $\rightarrow$ pages $\rightarrow$ paragraphs $\rightarrow$ sentences with respect to different book segmentation strategies, but it will increase the computational burden substantially. For simplicity, in this paper, we use the aforementioned four-level tree-structured representation for each author.

Biographies of most authors can be easily found on online Web sites. The root node at the top level contains word-frequency features extracted from the author biography Web page. Besides the root node, nodes at other levels include the same word-frequency features, but they are extracted from different domains of books. The rationale behind this book segmentation is that two books that have similar word histograms at the book level, i.e., the second layer can be completely different in terms of semantics or context, because different spatial distributions of the same set of words may lead to different meanings. This can be reflected by the lower parts of the tree, for instance, third- or fourth-level nodes. To encode node features in the tree, word histograms are first evaluated for the nodes at different levels. In order to make the system computable, we apply the principal component analysis (PCA) to the word histogram vector. Thus, the nodes model the compressed PCA features that describe frequency distribution of different words.

The above-described tree-structured representation is an effective way to conduct content-based book and author recommendation. For book recommendation, the similarity between two books can be compared and measured using the nodes at the second-to-fourth level. The second-level nodes provide global similarity, and the third- and fourth-level nodes give us local similarity. Likewise, for author recommendation, the similarity between two authors can be evaluated using the whole tree, where the root nodes give comparative results between the authors with respect to their biographical introductions, and the children nodes at other levels provide the similarity lying in their writings by capturing the local spatial information of word distributions. As a result, a content-based author recommender system can be implemented using similarity measurements.

B. Implementation Details

Given an author, the overall implementation framework of a tree-structured representation for the author is shown in Fig. 2. First, the preprocessing, word extraction, vocabulary construction, and generation of a PCA projection matrix are performed. Second, books are partitioned into pages that are further partitioned into paragraphs, and then nodes' features are computed using the word histogram vectors. Third, features of nodes are projected into lower dimensional PCA space, and a tree structure encoding different features is returned for each author.

1) Preprocessing: The biographies of all the authors we used in this paper were collected from Wikipedia in the .html format. Only the texts were extracted from the main frame, and the contents of references, external links, and the other information which are not related to the biography were filtered out. In addition, the text appearing within the html tags used for formatting was not accounted for in the word count or the document feature extraction. The e-books used in this paper were downloaded from Project Gutenberg.[1] The copyright information embedded in the text body was also filtered out. After these preprocessing steps, words from all the documents were extracted in the data set, and stemming was subsequently applied to each word [37]. Stems are often used as basic features instead of original words. Thus, program, programs, and programming were all considered as the same word. We removed the stop words, a set of common words, such as a, the, and are, and then stored the stemmed words together with the information of the *tf*, f_u^t (the frequency of the uth word in all the documents), and the document frequency (*df*), and f_u^d (the number of documents, the uth word appears).

2) Vocabulary Construction: Forming a histogram vector for each document requires the construction of a word vocabulary that each histogram vector can refer to. For author biographies, according to the stored *tf* and *df*, we use the well-known *tf-idf* term-weighting measure to calculate the weight of each word

$$w_u = f_u^{t,\text{bio}} \cdot idf \tag{1}$$

where idf denotes the inverse document frequency that is given by $idf = \log_2(N_{\text{bio}}/f_u^d)$, and N_{bio} is the total number of biographies in a data set. The words are then sorted in descending order according to their weights. The first words T_{bio} are selected to construct the vocabulary M_{bio} for author biographies. In this paper, we set $T_{\text{bio}} = 10\,000$.

[1] http://www.gutenberg.org.

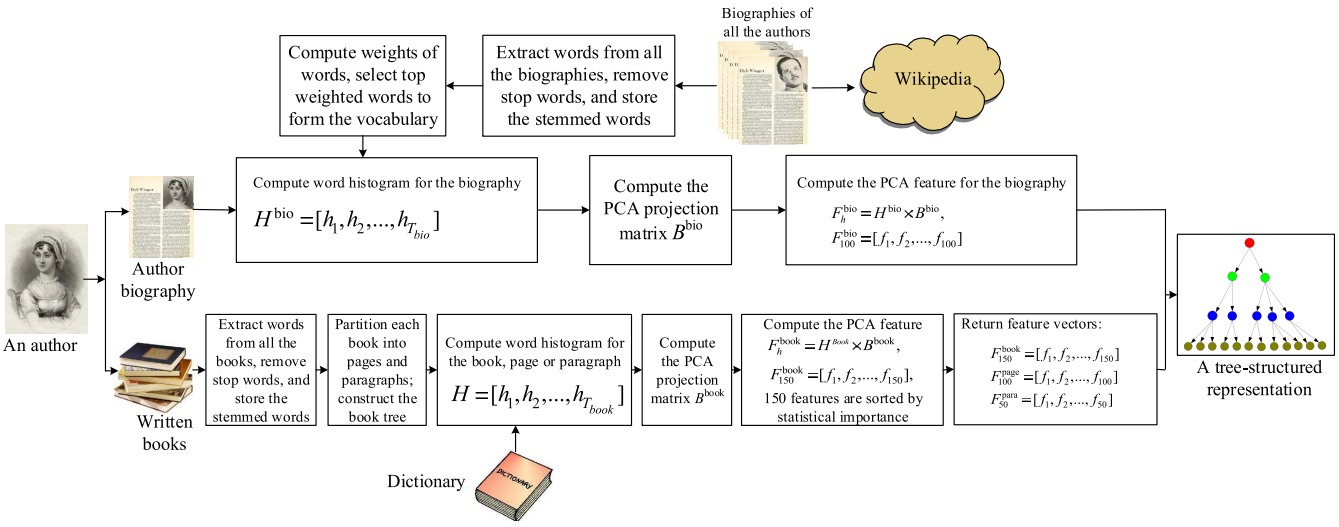


Fig. 2. Construction of tree-structured feature.

For e-books, we used an English–Chinese dictionary. It contains 24 678 words in total which are widely used in real life. After performing stemming and stop-word removal, 15 601 words were left in the final vocabulary. This vocabulary is available on our Web site.[2] There are two major reasons to use an external vocabulary. First, as books are usually lengthy, a large-scale book set may cover the full vocabulary we usually use. Second, most old novels were converted into e-books through an optical character recognition (OCR) process prior to being uploaded to the Internet. OCR errors may cause numerous words to be misspelled and also to be combined with neighboring words. As a result, the vocabulary will be hugely expanded if we use the common vocabulary construction method. For example, in keeping words that appeared over five times, the vocabulary size of the book set we used still reached to 1 232 445.

3) Book Segmentation: A book was partitioned into pages that were further partitioned into paragraphs in order to extract the spatial distributions of words over a book. We developed Java code to perform the segmentation task. In e-books, paragraphs can be easily identified using the separator $\backslash r\backslash n$. First, a book was segmented into a number of paragraph blocks using the paragraph separator $\backslash r\backslash n$. In order to control the number of paragraphs, we merged the subsequent blocks to form a new paragraph until the total number of words of the merged blocks exceeded the paragraph threshold. In this paper, we set the minimum threshold for the total number of words in a paragraph at 50; otherwise, the new paragraph block was merged with the previous paragraph. In this way, paragraph blocks that comprise only a few words, for example, one sentence as a paragraph, will be attached to relatively larger paragraph blocks. A few paragraphs were merged to form a new page when the total number of words of the merged paragraphs exceeded a page threshold value which was set at 1000. The overall book partitioning process can be summarized as follows.

1) Partition a book into paragraph blocks using the paragraph separator $\backslash r\backslash n$.
2) Merge the subsequent blocks to form a new page until the total number of words of the merged blocks exceeds a page threshold (set at 1000). There is no minimum threshold for the last page. The pages are formed.
3) Partition each generated page into paragraphs using the separator again. Merge these subsequent paragraphs in the same fashion as in Step 2) to form a new paragraph until the total number of words of the merged paragraphs exceeds the paragraph threshold value of 50. The minimum threshold for the last paragraph of a page is set at 30; otherwise, the paragraph is merged with the previous paragraph.

It is worth noting that no specific rule exists for determining the minimum/maximum number of words for a page. However, the use of a word count threshold as an averaged number of words in normal books enables us to control the number of pages in each book. In addition, setting the minimum threshold for the total number of words in a paragraph enables us to attach the paragraphs that contain only a few words to relatively larger paragraphs. Thus, we can appropriately control the number of paragraphs in a page as well. Using above-described book segmentation process, we are able to build a hierarchical tree structure of each book for describing the semantic information from global data view to local data view.

4) Histogram Calculation: In the author tree structure, the root node contains the histogram of the biography of an author; the second-level nodes are used for books; the third-level nodes are used for pages; and the fourth-level nodes are used for paragraphs. Given a biography, the word histogram can be represented by $\tilde{H}^{\text{bio}} = [n_1, n_2, \ldots, n_u, \ldots, n_{T_{\text{bio}}}]$, where n_u is the number of times the corresponding word u appears in a biography. For each book, word histograms for paragraphs can be described in the first place by $\{\tilde{H}^{\text{para}} = [n_1, n_2, \ldots, n_v, \ldots, n_{T_{\text{book}}}]\}$, where n_v is the number of times that the word v appears in a paragraph. The histogram

[2]http://www.ee.cityu.edu.hk/~twschow/bookvocabulary.txt.

sets for the pages, and the entire books can be achieved by the hierarchical relationship $\{\tilde{H}^{\text{page}} = \sum \tilde{H}^{\text{para}}\}$ and $\{\tilde{H}^{\text{book}} = \sum \tilde{H}^{\text{page}}\}$, where $\tilde{H}^{\text{page}}$ and $\tilde{H}^{\text{book}}$ are histogram vectors corresponding to a page and a book, respectively. We then use the *tf-idf* scheme to weight the histograms. For each biography, we weight the histogram in the form of

$$H^{\text{bio}} = [h_1, h_2, \ldots, h_u, \ldots, h_{T_{\text{bio}}}] \quad \text{where} \quad h_u = n_u \times \log_2\left(N_{\text{bio}}/f_u^d\right). \tag{2}$$

Likewise, for a book (or a page, or a paragraph), the weighting histogram is in the form of

$$H = [h_1, h_2, \ldots, h_v, \ldots, h_{T_{\text{book}}}] \quad \text{where} \quad h_v = n_v \times \log_2\left(N_{\text{book}}/f_v^d\right) \tag{3}$$

where N_{book} is the total number of books in the data set.

5) PCA Projection: In a tree, the histogram of each node is usually a large-size vector, and the total number of nodes in a book set is also large. In order to make the framework computationally efficient, PCA was used to project each node histogram into a lower dimensional feature vector. It is noted that other dimensionality reduction methods, such as LSI [12], PLSI [13], and LDA [14], can be used for feature projection. We used the MATLAB tool [27] to calculate the PCA projection matrix. For the root node, i.e., the biography histogram, the projected feature is calculated by

$$F_h^{\text{bio}} = H^{\text{bio}} \times B^{\text{bio}} \tag{4}$$

where B^{bio} is the projection matrix of dimension $T_{\text{bio}} \times m_F^{\text{bio}}$, and m_F^{bio} is the dimension of the projected feature. In our study, m_F^{bio} was set at 100.

We constructed the projection matrix B for book contents only at the second-level nodes. It means that we used the histograms at the book level to compute the matrix B, and this same matrix B was used to project the histogram features for pages and paragraphs. According to our empirical study, generating the PCA projection matrix separately at the page and paragraph level delivers similar result using the reduced matrix generated from the projection matrix at the book level. The projection can be accomplished by

$$F_h = H \times B \tag{5}$$

where B is the projection matrix of dimension $T_{\text{book}} \times m_F$, where m_F is the dimension of the projected feature. The above-mentioned histogram features H for the second-, third-, and fourth-level nodes are extracted from a book, a page, and a paragraph, respectively. One advantage of such a projection is that the projected features in F_h are ordered according to their statistical importance. In this paper, m_F was initially set at 150 for the book-level nodes, but the number of features used at the third- and fourth-level nodes was further reduced with respect to pages and paragraphs, respectively. Thus, the new dimensions of the PCA features for the first-, second-, third-, and fourth-level nodes are 100, 150, 100, and 50, respectively.

In the end, each node in the tree structure was assigned a set of information such that we can see how the nodes are dependent on each other in the tree structure. The set of information included: 1) a node index; 2) the level in the tree; 3) the parent node index; 4) the child node index; and 5) the compressed PCA features. A tree-structured representation encoded with this set of information was returned, as shown in Fig. 1. It is worth noting that we need to save the vocabulary bases and the projection matrices for feature extraction of a new book query and a new author query. The tree-structured features of a new query can be extracted in a similar way.

III. Organization of Books and Authors Using MLSOM

This section introduces an MLSOM training framework. First, we briefly review the training process of the basic SOM algorithm. We then build a four-layer MLSOM system and provide its implementation details.

A. Self-Organizing Map

The SOM [28] is a versatile unsupervised neural network used for dimensionality reduction, vector quantization, and visualization. It is able to preserve a topologically ordered output map, where input data are mapped into a small number of neurons. A basic SOM comprises N neurons located on a regular low-dimensional grid, which is usually a 2-D grid. The lattice of the grid is either hexagonal or rectangular. The SOM algorithm is iterative. Each neuron i has a d-dimensional feature vector $w_i = [w_{i1}, \ldots, w_{id}]^T$. At each training step t, a sample data vector $x(t)$ is randomly selected from a training set. The distances between $x(t)$ and all the feature vectors $\{w_i\}$ are calculated. The winning neuron, denoted by c, is the neuron with the feature vector closest to $x(t)$ given by

$$c = \arg\min_i(S(x(t), w_i)), \quad i \in \{1, 2, \ldots, N\} \tag{6}$$

where $S(x(t), w_i)$ is a distance function between $x(t)$ and w_i.

In the sequential SOM algorithm, the winner neuron and its neighbor neurons are updated according to the weight-updating rule, which can be written as

$$w_j(t+1) = \begin{cases} w_j(t) + \eta(t)h_{jc}(t)(x(t) - w_j(t)) & \forall j \in N_c \\ w_j(t), & \text{otherwise} \end{cases} \tag{7}$$

where N_c is a set of neighboring neurons of the winning neuron. $\eta(t)$ is the learning rate which decreases monotonically with iteration t in the form of

$$\eta(t) = \eta_0 \cdot \exp\left(-\alpha \cdot \frac{t}{\tau}\right) \tag{8}$$

where η_0 is the initial learning rate, α is an exponential decaying constant, which is set to 3 in this paper, and τ is a time constant set to the maximum number of iterations. $h_{jc}(t)$ is the neighborhood kernel function that indicates the distance of a neighborhood neuron j with the coordinate (x_j, y_j) to the winning neuron c at the position (x_c, y_c). This neighborhood function is a nonincreasing function that can be taken as a Gaussian function

$$h_{jc}(t) = \exp\left(-\frac{[(x_j - x_c)^2 + (y_j - y_c)^2]}{2(\sigma(t))^2}\right) \tag{9}$$

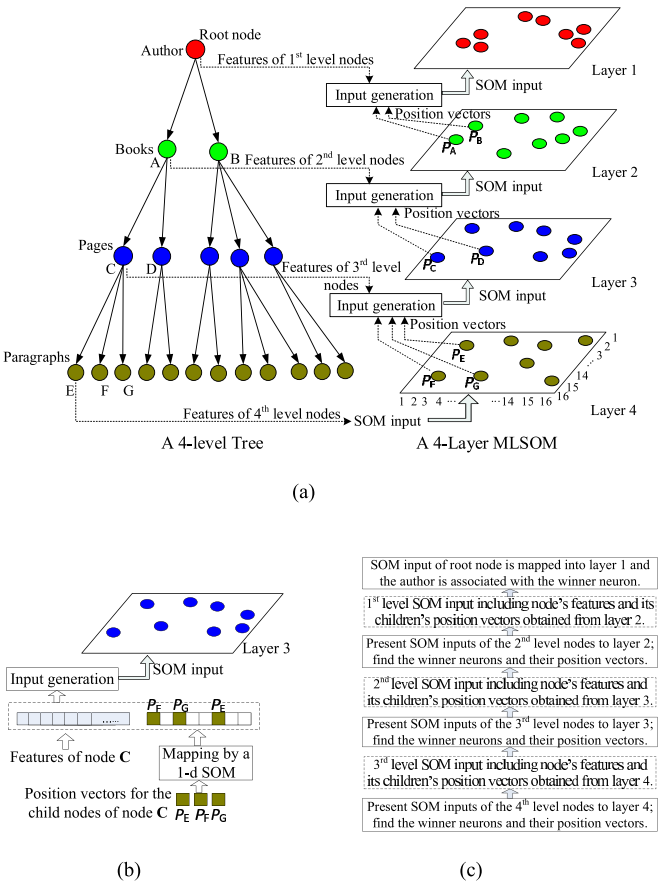


Fig. 3. Illustration of a four-layer MLSOM. (a) Mapping of a four-level tree-structured data into a three-layer MLSOM. (b) SOM input generation for node C. (c) Block diagram describing the mapping process of a tree-structured data into MLSOM.

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

$$\sigma(t) = \eta_0 \cdot \exp\left(-\frac{t}{\tau} \cdot \log(\sigma_0)\right) \tag{10}$$

where σ_0 is the initial width. The detailed training procedures can be found in [28].

B. MLSOM

The conventional SOM cannot represent tree-structured data, because its inputs are only represented by flat vectors. MLSOM [20], [22], an extension of SOM model, was developed for processing tree-structured data. Its multilayer structure is particularly designed for the node representation of a tree. A tree locates nodes at different levels. A node in the tree consists of two types of information: the node features of its own and its child nodes. In an MLSOM, there are as many SOM layers as the number of levels in the tree, for example, four different SOMs for a four-layer MLSOM used in our application. Nodes at each level are processed by the corresponding layer of an MLSOM. Fig. 3(a) shows how four-level tree data are mapped into a four-layer MLSOM. First, the fourth-level nodes, which comprise the features of different paragraphs, are mapped onto layer 4 as the bottom SOM output. The winner neurons of different nodes or paragraphs (e.g., nodes E, F, and G) are described by their corresponding position vectors (e.g., p_E, p_F, and p_G). It is worth noting that the fourth layer comprises a huge number of child nodes for representing all the paragraphs in a data set. Using the fourth-layer SOM enables us to compress all the paragraphs into 2-D position vectors, but the topology of the original paragraphs is preserved. As a result, the input vector of layer 3 SOM includes the third-level node (page) features and the corresponding position vectors of the fourth-level nodes. In this way, a compressed input vector for layer 3 SOM can be formulated. Fig. 3(b) shows how a third-layer input feature vector is encoded using the feature of node C and the position vectors p_E, p_F, and p_G. Fig. 3(b) also demonstrates how position vectors are formed from the outputs of layer 4 SOM into a structured format generating the input vector of layer 3 SOM. Similarly, the second-level nodes (books) and the root nodes (authors) can be processed at layer 2 SOM and layer 1 SOM, respectively. It should be noted that the fourth-, third-, and second-layer SOMs are used to compress the features with respect to the information of paragraphs, pages, and books, respectively. At the top layer, the root node can compactly represent the entire tree at the first-layer SOM. It is encoded by the information of author biographies and book contents. The whole feature compression procedures are summarized in the block diagrams, as shown in Fig. 3(c).

It is worth pointing out that the MLSOM considered here is different from traditional hierarchical SOMs, such as growing neural gas [31], growing hierarchical SOM [32], and tree-structured SOM [33], or even more advanced SOMs using spatial access methods (SAMs) [34] or metric access methods [35] to speed up the training process of hierarchical SOMs. An excellent survey on these tree-based variants of SOM can be found in [36]. These methods work on the SOM topology aiming at mapping a data set (or a set of patterns) onto a set of tree-structured neurons according to the spatial relationship among data samples (or patterns). The feature of each sample must have a flat structure, although the entire data set has a hierarchical structure constructed by these samples. In our proposed book and author organization system, each sample, however, has a tree-structured representation that aims at encoding the spatial information of the features of each sample. MLSOM designed for handling tree-structured data is then used to integrate the information from leaf nodes layer by layer.

C. MLSOM Training

In MLSOM, a node feature vector F_k at the kth level (here, $k = 1, 2, 3,$ or 4) is represented by $F_k = [f_{1,F}, f_{2,F}, \ldots, f_{m,F}, p_{1,F}, p_{2,F}, \ldots, p_{c_{\max},F}]^T$, where $f_{i,F}$ represents the ith feature of node F_k, $p_{j,F} = [x_{j,F}, y_{j,F}]^T$ ($x_{j,F} \in [0, 1]$, $y_{j,F} \in [0, 1]$) is a normalized 2-D position vector of a neuron that compactly represents the jth child node of node F_k, and $c_{\max}$ is the maximum number of child nodes of the nodes at the kth level. The position vectors $[p_{1,F}, p_{2,F}, \ldots, p_{c_{\max},F}]$ can be obtained according to the spatial positions that will be discussed later in this section. It should be noted that a node may have less than the $c_{\max}$ number of child nodes. As a result, some of $p_{j,F}$ may

contain zero vector [0, 0]. In MLSOM training, the weight vector W_k of a neuron at the kth layer is represented by $W_k = [f_{1,w}, f_{2,w}, \ldots, f_{m,w}, p_{1,w}, p_{2,w}, \ldots, p_{c_{\max},w}]^T$, where $f_{i,w}$ is the ith weight, and $p_{j,w} = [x_{i,w}, y_{i,w}]^T$ is a 2-D vector. To find the winner neuron, we have used the following function to compute the distance between a node and a neuron:

$$S(F, W) = C \cdot \left(1 - \frac{\sum_{i=1}^{m} f_{i,F} \times f_{i,w}}{\sqrt{\sum_{i=1}^{m} (f_{i,F})^2} \times \sqrt{\sum_{i=1}^{m} (f_{i,w})^2}}\right) + (1 - C) \cdot \frac{1}{\sum_{j=1}^{c_{\max}} (p_{j,F})} \sum_{j=1}^{c_{\max}} (p_{j,F}) \cdot d(p_{j,F}, p_{j,w}) \quad (11)$$

where

$$(p_{j,F}) = \begin{cases} 1, & \text{if } p_{j,F} \neq (0, 0) \\ 0, & \text{otherwise} \end{cases}$$

where C is a weight parameter, and d is a Euclidean distance function. The first part of the expression calculates the global distance in the form of the cosine distance using the node feature, and the second part computes the local distance using position vectors of child nodes. It is worth noting that, in the bottom-level nodes, only the first part of (11) is used. The weight parameter C is used to balance those two parts. It puts relative emphasis between the global and local distance measure. A larger value of C, $C > 0.5$, indicates that emphasis is placed on the global information, and a small value of C, $C < 0.5$, indicates that the local distance from the term spatial distributions is emphasized in between-document comparison. In this way, it provides users with flexibility to change the value of C to balance the distance measure according to their expectations. The effect of C will also be discussed in our experiment.

MLSOM training process starts with the fourth-layer SOM, which is trained using the node features at the fourth level (paragraph level). Then, the feature inputs of the third-level nodes (pages) are generated by integrating the nodes' features at this level and the position vectors of the corresponding child nodes, as shown in Fig. 3(b). The third-layer SOM is trained with these combined feature inputs. Similarly, the second-layer SOM is trained with the second-level nodes' features (books) and the position vectors of the child nodes generated from the third layer. Finally, the top-layer SOM is trained by the SOM inputs for the root nodes (authors). It is worth noting that mapping the position vectors of child nodes into the SOM input vector is required, while the inputs for each layered SOM are generating. Here, we use a simple 1-D SOM to produce the mapping of position vectors. This mapping aims at making an appropriate matching between two sets of nodes with respect to the local distance in the second part of (11). Despite the fact that two sets may contain a different number of nodes, the mapping produced by 1-D SOM enables us to compare a child node of the former set with only one similar child node of the latter set. This comparison can largely avoid the mismatch between any two sets. Thus, the local distance from leaf nodes can be well-estimated. The 1-D SOM is trained by all the position vectors over the data set at a certain level [see Fig. 3(b)]. Then, this well-trained 1-D SOM is employed to map the set of position vectors. The detailed mapping procedure can be found in [20].

D. Computational Complexity

MLSOM training is an iterative process. The computational complexity of conducting one epoch of a four-layer MLSOM training is $O(N_1 m_1 (n_1 + 2c_1) + \sum_{k=2}^{4} N_k (m_k (n_k + 2c_k) + 2c_k \bar{c}_k))$, where N_k is the total number of nodes at the kth level of all the trees, m_k is the number of neurons at the kth layer of MLSOM, n_k is the number of input features of a node at the kth level, c_k and $\bar{c}_k$ are the maximum and average number of children nodes at the kth level of all the trees. At the testing stage, given an author query, the computational complexity is around $O(N_1^Q m_1 (n_1 + 2c_1) + \sum_{k=2}^{4} N_k^Q (m_k (n_k + 2c_k) + 2c_k \bar{c}_k))$, where N_k^Q is the number of nodes at the kth level of the given author tree. The largest time cost comes from finding the best matching unit (BMU) using the sequential approach [36]. The number of operations for distance comparison at each level to find the BMU is $N_k^Q m_k$. Finding the BMU can be done in parallel.

IV. Book and Author Recommendation

For the purpose of practical applications, it is straightforward to develop a book and author recommender system under our proposed framework. In this section, we introduce the implementation details of the system using our framework.

A. Training Preprocess

Before conducting above-mentioned applications, we first perform the MLSOM preprocessing. The preprocessing procedures include the following.

1) Train the MLSOM with all the tree-structured data in the data set.
2) For the top-layer SOM, save the index of author data against their winner neurons. This will be used for author recommendation.
3) For the second-layer SOM, save the index of books and their node indexes against the corresponding winner neurons. This will be used for book recommendation.
4) Save all the weights of MLSOM and 1-D SOM. Thus, the trained MLSOM, together with the vocabulary bases and PCA projection matrices, gets ready to perform various applications.

B. Book Recommendation

It is natural to use our framework for book recommendation. Each book is represented by a node at the second level in the tree structure. These nodes are processed at the second-layer SOM. Each book can be indexed against its winner neuron at the grid of the second-layer SOM, because the book and neuron association has been achieved at the preprocessing stage. Thus, a content-based book recommender system considering term spatial distributions can be implemented in the following steps.

TABLE I
DISTRIBUTION OF NODES IN THE WHOLE DATA SET

Level	1 (Author)	2 (Book)	3 (Page)	4 (Paragraph)	All levels
Max. number of children nodes	100	1932	18	0	1932
Total number of nodes in data set	3868	10500	612176	2831898	3458442

1) For a given book query, extract its tree structure. Compute the projected features using prestored vocabulary base and PCA projection matrix.
2) Match the nodes of the tree level by level from the bottom layer (paragraph level) to the second layer (book level), find the closer neurons on the second-layer SOM grid and return their attached books.
3) Go through the sorted neurons in descending order and add their attached books into the recommendation list until at least user-defined N_{ret} books are appended.
4) Sort the books in the recommendation list by comparing the query with the inputs of the second-layer nodes according to 12. Recommend the first N_{ret} books to users.

C. Author Recommendation

In practice, a user (or reader) is most likely to buy or read the writings of his/her favorite author. Relying on these behavior data, an online store can recommend other authors together with their books, which are most relevant to the user's preferred authors. Although many factors may affect the relevance between two authors, biography and book contents are two critical features that can be used to evaluate the relationship between two authors, because the biography provides a detailed description of an author's life (i.e., global information), whereas book contents highlight the author's work, which appears as local information. In this paper, our proposed tree structure entails a combination of biography and book contents. The MLSOM-based author recommender is similar to the book recommender. Due to space limitation, the detailed recommendation procedure is not explicitly given.

V. EXPERIMENT

We conducted extensive experiments to evaluate the proposed framework—in particular, the performance of various models on content-based recommendation. Data collection, implementation details, experimental results, parametric study, and MLSOM visualization results are presented in the following.

A. Data Collection

At present, there is no publicly available data set for this kind of research. In order to provide a real life and demanding testing platform, we have established a large-scale data set which contains 3868 authors and 10500 e-books in English that they wrote. These books that are in the .txt format were collected from Project Gutenberg. Due to copyright, we cannot distribute the original raw text data at this stage, but the author and book list can be downloaded at www.ee.cityu.edu.hk/~twschow/booklist.txt for other researchers. Statistics on node distributions over the whole data set are listed in Table I. It is also observed

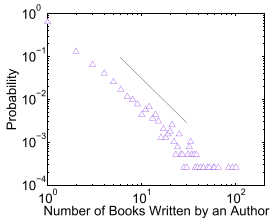


Fig. 4. Probability distribution against number of books written by an author.

from the data set that the numbers of writings of most authors are smaller than 10. The top three authors are Fenn, Kingston, and Ballantyne, who have written 100, 92, and 82 books, respectively. To further investigate the distribution property of number of books against different authors in the data set, we plotted the probability distribution against the number of books written by an author, as shown in Fig. 4. From the perspective of complex networks [29], this distribution can be described by a power-law function of the form

$$P(k) \sim k^{-\gamma} \tag{12}$$

where $P(k)$ represents the probability that an author has written exact k books, and γ is an exponential constant determined by the given data set, for example, in our data set, $\gamma \approx 1.69$.

B. Experimental Design

According to the scale of the data set, the size of the top, second, third, and bottom layers of MLSOM was set at 30×30, 36×36, 42×42, and 48×48, respectively. 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 the rounded multiple of the number of data-nodes in the corresponding level for the top, second, third, and bottom layers, respectively. In this paper, these settings of the above parameters were observed to be a good choice. For comparison, we compare the MLSOM-based method with the state-of-the-art algorithms, including RAP [38], LDA [14], PLSI [13], LSI [12], and VSM [11]. We also tested another two algorithms: 1) MLSOM-Global, which only uses the global distance in MLSOM (i.e., the weight parameter $C = 1$) and 2) MLSOM-Local, which relies on the local distance in MLSOM (i.e., the weight parameter $C = 0$). All the experiments were performed on a PC with Intel(R) Xeon(R) CPU X3430@ 2.40 GHz and 8.00 GB memory. The feature extraction programs were written in Java programming language. The MLSOM programs were tested on MATLAB 7.12.0 (R2011a). To quantify the recommendation results, we used three metrics, which have been commonly adopted in recommender systems [30]: 1) mean reciprocal rank (MRR); 2) success at rank k (S@k), where k is the number of recommended books/authors; and 3) precision at rank k (P@k).

C. Book Recommendation

To evaluate recommendation performance according to the aforementioned metrics, we first need to assess the relevance

TABLE II

DISTRIBUTION OF NODES IN bookset-goodreads.com FOR BOOK RECOMMENDATION

Level	2 (Book)	3 (Page)	4 (Paragraph)	All levels
Max. number of children nodes	1932	13	0	1932
Total number of nodes in training set	2438	158765	726923	888126
Total number of nodes in testing set	270	19824	88753	108847

TABLE III

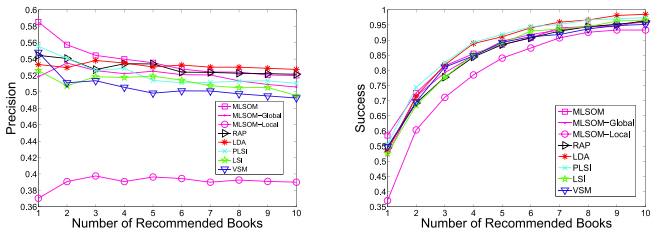
COMPARATIVE RESULTS OF DIFFERENT METHODS ON bookset-goodreads.com (PERCENTAGE)

Method	MRR	S@1	S@5	P@5	P@10
MLSOM	71.46	58.52	88.52	53.56	52.00
MLSOM-Global	67.76	51.85	89.63	52.52	50.59
MLSOM-Local	57.01	37.04	84.07	39.63	39.00
RAP	68.49	54.44	88.52	53.41	52.15
LDA	69.22	53.33	91.11	53.04	52.74
PLSI	70.87	55.56	91.85	51.41	51.04
LSI	67.56	52.59	88.89	51.93	49.56
VSM	69.01	54.81	89.26	49.85	49.26

between two books. It will be laborious and time consuming to have them assessed manually over the whole data set. To overcome this difficulty, we used categorical information of books. Two books are assumed to be relevant if they are from the same category. However, the books collected from Project Gutenberg do not have such categorical information. To obtain book categories, we input the whole book list from our data set into two online Web sites: www.goodreads.com and www.ebooks.com, which consist of many popular categorized e-books. Two subsets, i.e., bookset-goodreads.com and bookset-ebooks.com, were found in these two sites, respectively. The found books were attached to category labels.

1) Results on Bookset-Goodreads.com: The bookset-goodreads.com contains 2708 books. Each book is labeled by multiple genres associated with the number of users who label the book as these genres. For instance, the genres of book *Mother Carey's Chickens* include classics labeled by five users, kids labeled by two users, and children's labeled by two users. Based on this observation, we define that two books are relevant if any of their labeled genres are overlapped. We used this subset to test the performance of various algorithms on recommendation task. We randomly assigned 2438 books, i.e., around 90% of the subset, as a candidate set and 270 books as a test set that is used for query. The lists for training and test are available in our site. Node distribution in different tree-levels of book data (second layer to fourth layer, because we only need to use the book, page, and paragraph level for book recommendation) is listed in Table II.

The numerically comparative results of different methods are summarized in Table III. The results of MLSOM are based on the optimal weight $C = 0.4$. We have included the effectiveness study on this parameter latter in this section. We also include the P@k and S@k results visually shown in Fig. 5 for k ranging from 1 to 10. As shown in Table III, MLSOM delivers better MRR value than other methods. From Fig. 5(a), it is observed that MLSOM outperforms VSM, LSI, PLSI, MLSOM-Global, and MLSOM-Local with respect to precision when the top ten books are recommended. In addition, MLSOM performs better than LDA when the top five books are returned, but LDA slightly outperforms



(a) (b)

Fig. 5. Results of different methods on bookset-goodreads.com in terms of (a) P@k and (b) S@k ($k = 1, 2, \ldots, 10$).

TABLE IV

DISTRIBUTION OF NODES IN bookset-ebooks.com FOR BOOK RECOMMENDATION

Level	2 (Book)	3 (Page)	4 (Paragraph)	All levels
Max. number of children nodes	373	10	0	373
Total number of nodes in training set	1863	118057	542933	662853
Total number of nodes in testing set	206	13314	60636	74156

TABLE V

COMPARATIVE RESULTS OF DIFFERENT METHODS ON bookset-ebooks.com (PERCENTAGE)

Method	MRR	S@1	S@5	P@5	P@10
MLSOM	68.95	60.19	81.07	57.09	55.19
MLSOM-Global	66.44	53.88	80.58	53.20	52.09
MLSOM-Local	55.59	43.20	70.39	43.50	42.62
RAP	66.75	56.31	76.70	55.44	55.78
LDA	67.07	56.31	78.16	55.63	54.90
PLSI	68.95	59.22	82.04	53.98	53.30
LSI	66.81	55.34	80.10	53.01	51.75
VSM	66.52	55.83	79.13	52.33	51.94

MLSOM with the increase of the number of recommended books. It is clear that combining the information from the global and local does bring performance improvement when comparing MLSOM with MLSOM-Global and MLSOM-Local. From Fig. 5(b), we can observe that MLSOM can produce at least 3% improvement over other methods in terms of S@1, but it produces worse results compared with PLSI and LDA when the number of recommended books, k, ranging from 4 to 10.

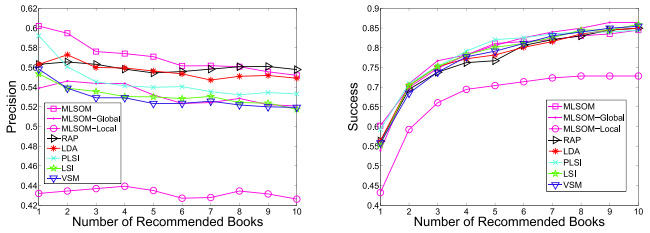
2) Results on Bookset-ebooks.com: The bookset-ebooks.com contains 2069 books. Each book is labeled in multiple levels by the Web site. For instance, the book *Jungle Tales of Tarzan* is labeled as Fiction > Classics Fiction > Action and Adventure. In this paper, we define that two books are relevant if their labels are the same at the top level. We randomly assigned 1863 books as a candidate set and 206 books as a test set. The lists for training and test can be found in our site. Node distribution in different tree-levels of book data is listed in Table IV.

Table V shows the quantity results of different methods. The results of MLSOM are based on the optimal weight $C = 0.35$. It is observed that MLSOM and PLSI deliver better MRR value than other methods. In particular, MLSOM performs the best in terms of S@1. It can produce around 4% improvement over MLSOM-Global, MLSOM-Local, RAP, LDA, LSI, and VSM. Furthermore, MLSOM outperforms other methods in terms of P@5. Fig. 6 visually demonstrates the P@k and S@k results ($k = 1, 2, \ldots, 10$). From Fig. 6(a), MLSOM consistently outperforms other methods when eight books are recommended. It is noted that MLSOM performs

TABLE VI

RESULTS OF DIFFERENT METHODS USING BOOK *DATA MINING CONCEPTS AND TECHNIQUES* AS A QUERY

No.	Amazon.com	MLSOM	MLSOM-Global	MLSOM-Local	RAP	LDA	PLSI	LSI	VSM
1	Data Mining, Practical Machine Learning Tools and Techniques_Ian H. Witten.txt	8	9	69	8	2	7	9	6
2	Data Mining with Microsoft SQL Server_Jamie MacLennan.txt	6	6	1901	12	11	8	8	8
3	Introduction to Data Mining_Pang-Ning Tan.txt	14	8	875	6	6	5	7	5
4	Data Mining Techniques For Marketing, Sales, and Customer Relationship Management_Gordon S. Linoff.txt	1	1	42	2	5	4	2	1
5	Pattern Recognition And Machine Learning_Christopher M. Bishop.txt	16	16	15	14	16	19	16	15
6	Data Mining with R Learning with Case Studies_Luis Torgo.txt	10	11	961	5	3	6	11	10
7	Principles of Data Mining_David J. Hand.txt	9	10	86	3	10	12	10	3
8	Mining the Social Web_Matthew A. Russell.txt	5	5	1900	9	4	1	4	12
9	Machine Learning-A Probabilistic Perspective_Kevin P. Murphy.txt	15	17	23	13	17	17	17	14
10	Data Science for Business_Foster Provost.txt	4	3	2044	1	9	11	1	2
11	An Introduction to Statistical Learning_Gareth James.txt	11	13	1548	10	15	14	12	13
12	Python for Data Analysis_Wes McKinney.txt	2	2	2043	7	1	2	3	9
13	Learning from data_Yaser S. Abu-Mostafa.txt	12	14	49	23	23	13	13	19
14	Data Analysis with Open Source Tools_Philipp K. Janert.txt	7	7	83	4	8	10	6	4
15	Fundamentals of Database Systems_Ramez Elmasri.txt	3	4	18	11	12	9	5	7
16	Introduction to Information Retrieval_Christopher D. Manning.txt	13	15	53	16	7	3	15	17
17	Probabilistic Graphical Models Principles and Techniques_Daphne Koller.txt	18	20	41	18	29	20	20	18
18	Introduction to Algorithms_Thomas H. Cormen.txt	19	21	27	21	13	45	21	21
19	Bayesian Reasoning and Machine Learning_David Barber.txt	17	19	14	15	19	18	18	16
Average Rank	10	13.63	15.91	216.1	17.02	17.6	19.89	15.35	16.81



(a) (b)

Fig. 6. Results of different methods on bookset-ebooks.com in terms of (a) P@k and (b) S@k ($k = 1, 2, \ldots, 10$).

clearly better than LDA and RAP when a few numbers of books (less than five) are retrieved, but the precision curves tend to merge with the increase of the number of recommended books. In contrast to the comparative results of precision, MLSOM, MLSOM-Global, RAP, LDA, PLSI, LSI, and VSM deliver similar S@k results, as shown in Fig. 6(b).

3) Example: To further evaluate the performance of the proposed framework, we used a popular book in data mining, *Data Mining Concepts and Techniques* (second edition), written by Han and Kamber as a query. We then found another 19 books recommended by Amazon on the home page of book *Data Mining Concepts and Techniques*. These 19 books were combined into the subset, bookset-ebooks.com. Thus, 2088 books in total were used as the candidate set. Apparently, these 19 books are more relevant to book *Data Mining Concepts and Techniques* compared with the books in bookset-ebooks.com. The ranking list recommended by Amazon recommender system is shown in the second column of Table VI. The file name contains the book title and its author name which are separated by the symbol "_". We used this ranking list as the ground truth. The values in Table VI are the ranking numbers of these 19 books recommended by different methods. For instance, the ranking number of book *Data Mining, Practical Machine Learning Tools and Techniques* written by Witten is 1 given by Amazon, 8 using MLSOM, 9 using MLSOM-Global, 69 using MLSOM-Local, 8 using RAP, 2 using LDA, 7 using PLSI, 9 using LSI, 6 using VSM, respectively. From Table VI, we can observe that the top 19 books are all relevant books by MLSOM recommendation, whereas a few irrelevant books have been mistakenly returned in the ranking list using other methods. We can observe this by checking if the number of ranking numbers is larger than 19 in Table VI. For example, LDA recommends two irrelevant books in the top 19 ranking list, and the ranking number of the first irrelevant book is 14; MLSOM-Global also recommends two irrelevant books in the top 19 ranking list, and the ranking numbers of these two irrelevant books are 12 and 18. We also summarize the average rank for different approaches. It is clear that MLSOM performs better than other methods, because the average rank of 13.63 given by MLSOM is closer to the ground truth, 10, given by Amazon.

TABLE VII

DISTRIBUTION OF NODES IN AUTHOR SUBSET FOR RECOMMENDATION

Level	1 (Author)	2 (Book)	3 (Page)	4 (Paragraph)	All levels
Max. number of children nodes	100	1932	11	0	1932
Total number of nodes in training set	923	4358	264784	1220157	1490222
Total number of nodes in testing set	102	539	24004	111507	136152

D. Author Recommendation

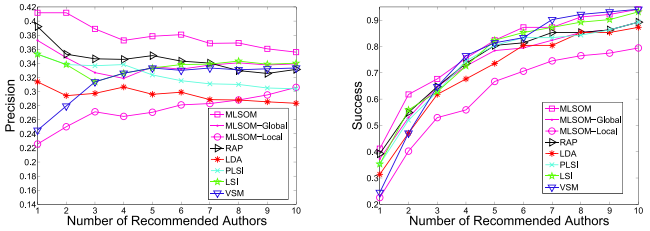
As discussed before, given an author query, the recommendation can be implemented by returning an author list, in which each author is the most relevant to the query. The entire data set includes 3868 authors associated with 10 500 books they have written. To evaluate the performance of author recommendation, we inputted the whole author list into Web site: www.goodreads.com and searched the authors that have been labeled by this site. We found 1025 authors. These authors wrote 4897 books in total. Each author has been multilabeled by the site. For instance, author Baum is labeled as Children's Books, Fantasy and Fiction. We define that two authors are relevant if any of their labels are matched. We used this subset to test the performance of various algorithms on author recommendation. We randomly assigned 923 authors as a candidate set and 102 authors as a test set. The lists for training and test are available in our site. Node distribution in different tree levels of author data is listed in Table VII.

The numerically comparative results are summarized in Table VIII. The results of MLSOM are based on the optimal weight $C = 0.6$. It is clear that MLSOM produces superior performance in comparison to other algorithms. In particular, MLSOM brings over 3% improvement of MRR

TABLE VIII

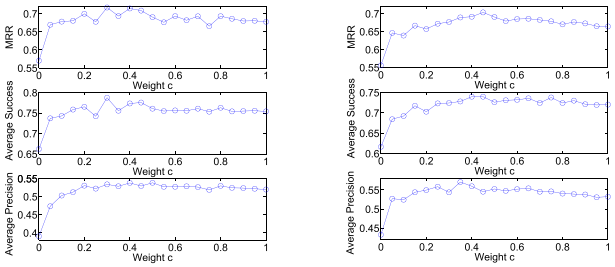
RESULTS OF DIFFERENT METHODS FOR AUTHOR RECOMMENDATION (PERCENTAGE)

Method	MRR	S@1	S@5	P@5	P@10
MLSOM	58.67	41.18	82.35	37.84	35.59
MLSOM-Global	54.16	37.25	78.43	33.53	33.92
MLSOM-Local	41.33	22.55	66.67	27.06	30.59
RAP	55.66	39.22	80.39	35.10	33.14
LDA	49.82	32.35	72.55	28.63	29.22
PLSI	53.05	35.29	80.39	32.35	30.39
LSI	54.14	35.29	82.35	33.33	34.02
VSM	47.57	24.51	81.37	33.33	33.33



(a) (b)

Fig. 7. Results of different methods on author recommendation in terms of (a) P@k and (b) S@k ($k = 1, 2, \ldots, 10$).



(a) (b)

Fig. 8. Results against different weights C for book recommendation on (a) bookset-goodreads.com and (b) bookset-ebooks.com.

and over 2.7% improvement of P@5 compared with other methods. Fig. 7 visually demonstrates the P@k and S@k results ($k = 1, 2, \ldots, 10$). From Fig. 7(a), MLSOM performs consistently better than other methods with respect to precision. As presented in Fig. 7(b), MLSOM outperforms other approaches in terms of S@k ($k = 1, 2, 3$).

E. Parametric Study

Based on the assumption that using combined distances from the global and local is able to deliver performance improvement, there must have an optimal weight C to balance this effect on 12. Fig. 8 shows the MRR, average P@k, and average S@k ($k = 1, 2, \ldots, 10$) values produced by MLSOM against the weight values varying from 0 to 1 at an increment of 0.05 for book recommendation data sets. Fig. 9 shows the results of MLSOM against the weight values ranging from 0 to 1 at an increment of 0.1 for author recommendation data set. It is observed that there is an optimal weight to balance the importance of the global and local information in a way that the contribution of the global and local semantics is demonstrated. As shown in Fig. 8(a) and (b), setting the weight C at around 0.35 and 0.4 for bookset-goodreads.com and bookset-ebooks.com, respectively, appears

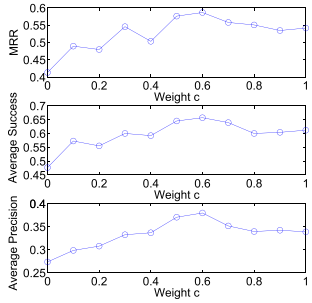
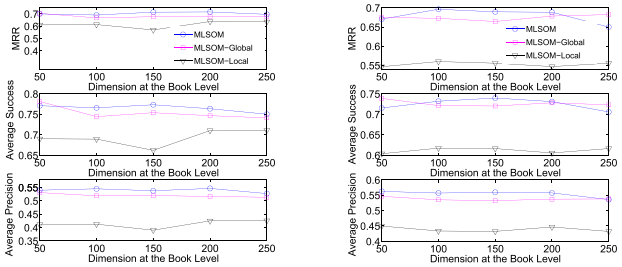


Fig. 9. Results against different weights C for author recommendation.



(a) (b)

Fig. 10. Results against different PCA dimensions at the book level on (a) bookset-goodreads.com and (b) bookset-ebooks.com.

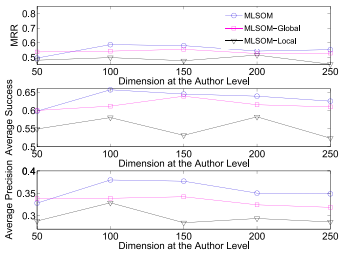
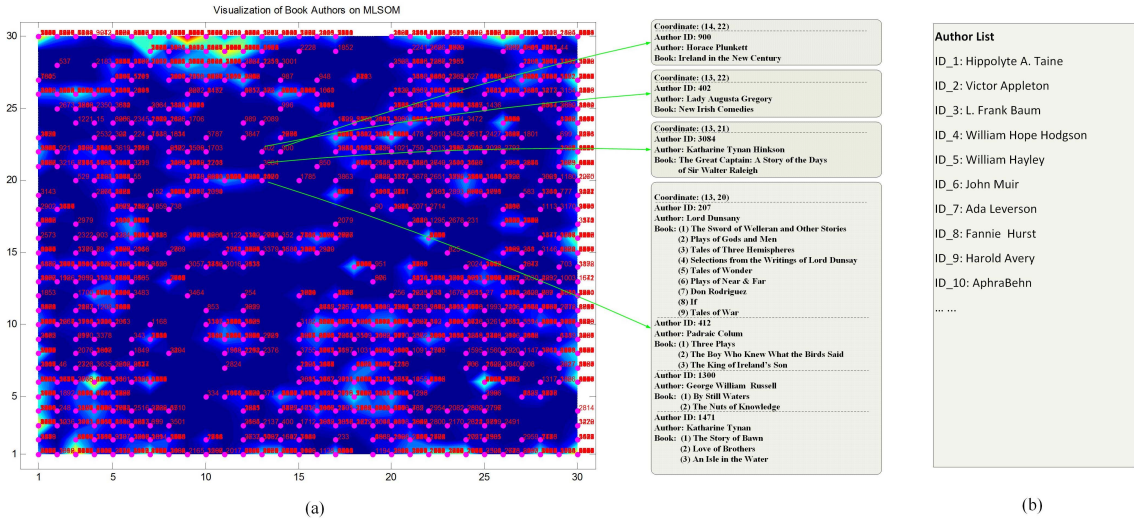


Fig. 11. Results against different PCA dimensions at the author level.

to be a good selection for the MLSOM algorithm. At the meantime, the optimal value of C stays around 0.6 for the author recommendation subset, as shown in Fig. 9. The optimal value of C is dependent upon different data sets. Users can specify the relative emphasis between the global and local distances by choosing an appropriate value of C according to the nature of books or authors. According to our empirical study, setting the weight C in the range of 0.3–0.5 and in the range of 0.5–0.6 is more probable to achieve promising results for book recommendation and author recommendation, respectively.

The number of PCA projected feature dimensions may also affect the recommendation results of our system. To show this effect on the MLSOM system, for book recommendation, we provided the results of MLSOM-related methods with different feature dimensions varying from 50 to 250 at an increment of 50 at the book level, as shown in Fig. 1. MLSOM has used the optimal weight. The results are shown in Fig. 10(a) and (b). It is observed that MLSOM-Global performs slightly better than MLSOM in terms of MRR and average S@k ($k = 1, 2, \ldots, 10$) when the PCA dimension is equal to 50 on bookset-goodreads.com. On the other hand, MLSOM-Global



(a) (b)

Fig. 12. Visualization map of authors. (a) Visualization of authors on MLSOM. (b) Author list.

outperforms MLSOM in terms of MRR and average S@k ($k = 1, 2, \ldots, 10$) when the PCA dimensions are equal to 50 and 250 on bookset-ebooks.com. However, for both book data sets, MLSOM consistently outperforms MLSOM-Global and MLSOM-Local in terms of average precision. At the meantime, the performance of MLSOM on average precision is very stable along with the change of feature dimensions. It suggests that the result of MLSOM is not sensitive to the PCA projected feature dimension for book recommendation. Moreover, Fig. 11 shows the results of MLSOM-related methods with different feature dimensions at the author level. The results show that MLSOM performs better than MLSOM-Global and MLSOM-local along with the change of feature dimensions from 100 to 250 at the author level. The best performance is delivered by MLSOM when the PCA feature dimension is equal to around 100.

F. Computational Time Performance

To examine the time performance of our proposed technique, we conduct an empirical study on the two book sets, bookset-goodreads.com and the bookset-ebooks.com. The average time cost of different methods for one book query is summarized in Table IX. Apparently, the recorded computational time [under a PC with Intel(R) Xeon(R) CPU X3430@2.4 GHz and 8 GB Memory] is all around 1 s which is well within satisfactory scale for practical applications. As LDA and PLSI deliver similar time performance with LSI, their results are not explicitly given. From Table IX, we can observe that LSI performed slightly faster than MLSOM, because it uses only the low-dimensional document-level feature. Compared with LSI, MLSOM requires a fraction of a second more, as it needs to integrate the information at the page and paragraph level. To further test the time performance of MLSOM for recommendation, we compiled a set including 5000 full-text books that are used for the candidate books. The time cost of MLSOM for one book query is ∼1.81 s. On the other hand, given a database, the book relevance in terms of book content can be achieved offline using MLSOM.

TABLE IX
TIME COST OF DIFFERENT METHODS FOR BOOK RECOMMENDATION(S)

Method	MLSOM	LSI	VSM
Bookset-goodreads.com	1.08	0.09	1.92
Bookset-ebooks.com	0.91	0.06	1.63

To support a production system, a parallel version of MLSOM deserves a further investigation in the future.

G. Visualization

Visualizing authors on a map is another potential application of our proposed framework. In online applications, authors are usually listed in an alphabetical order. For example, Project Gutenberg contains over ten thousands of authors which can be browsed under an author list. However, it is difficult for users to find the hidden relevance among these authors. With the aid of our proposed framework, we can take advantage of MLSOM to lay out all of the authors through a map. In a well-trained MLSOM with all the tree-structure author features, the top-layer SOM is a map of authors that demonstrates author clusters capturing the relevance pattern among authors. Thus, we can offer a visualization solution to online applications. Furthermore, the system is able to assist users to analyze or extract hidden pattern from presented author clusters. In addition, the system can provide an interesting visualization result on the similarity distance between two authors, e.g., Tynan and Russell, who are Irish-born novelists.

Based on our compiled data set, the visualization result of our system is shown in Fig. 12. The 3868 authors were mapped onto a 30 × 30 grid at the top layer of MLSOM, in which the numbers represent author identities, and the contour lines denote the densities of those neighborhoods with respect to the number of authors projected onto the grid. The darker blue indicates a smaller number of authors on the grid. On the contrary, the darker yellow represents a larger number of authors on the grid. Through clicking on the grid, the system can show the corresponding coordinate,

the authors and the books that they have written. For example, Plunkett and Gregory are located at two neighboring grids, (14, 22) and (13, 22), respectively. The result also shows that the topics of their books are all related to Ireland. In addition, it is noticed that Tynan and Russell are very close to each other, because they are mapped onto the same grid in our MLSOM system. Finally, the presented results demonstrate that the proposed MLSOM is capable of forming a semantic map of authors according to their books' topics, which can be useful for e-book applications.

VI. Conclusion

This paper investigated the use of MLSOM for the organization of e-books and their authors. A tree structure was proposed to represent the rich features of an author. These features included the biography regarding the author's background information and book contents in terms of the author's previous works. Book content was partitioned into a book-page-paragraph hierarchy. Content-based book and author recommender systems were implemented under the proposed framework. Our presented results demonstrate that our system can be used as an effective solution to e-book applications. Finally, the visualization results of our system were also presented. With the e-books being getting increasingly popular, we can perceive the potential applications of the proposed MLSOM framework. In the future, developing other methods to capture the local distance from leaf nodes, instead of using 1-D SOM to generate position vectors in MLSOM, may well be worth a further investigation. It is also interesting to incorporate SAM [34] into MLSOM for time performance improvement. We may achieve better scalability of the system by gaining insight from string kernels computation based on suffix arrays [39]. We also plan to extend our method in a supervised manner for automatic parameter settings. In addition, a parallel version of MLSOM deserves further study.

Acknowledgment

The authors would like to thank reviewers for their detailed and useful comments.

References

[1] Grant Gross, IDG News Service, and PCWorld. (Nov. 19, 2007). *Amazon Launches Kindle e-Book Reader*. MacBooks. [Online]. Available: http://www.macworld.com/article/1061113/kindle.html

[2] S. Malik. (Aug. 6, 2012). *Kindle eBook Sales Have Overtaken Amazon Print Sales, Says Book Seller*. The Guardian. [Online]. Available: http://www.theguardian.com/books/2012/aug/06/amazon-kindle-ebook-sales-overtake-print

[3] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, 2014, Art. ID 3.

[4] Y. Cai, H.-F. Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 766–779, Mar. 2014.

[5] M. Soares and P. Viana, "Tuning metadata for better movie content-based recommendation systems," *Multimedia Tools Appl.*, vol. 74, no. 17, pp. 7015–7036, 2014.

[6] A. Kent *et al.*, *Use of Library Materials: The University of Pittsburgh Study*. New York, NY, USA: Marcel Dekker, 1979.

[7] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in *Proc. 2nd Int. Conf. Ubiquitous Inf. Manage. Commun.*, Suwon, Korea, Jan. 2008, pp. 208–211.

[8] E. Rich, "User modeling via stereotypes," *Cognit. Sci.*, vol. 3, no. 4, pp. 329–354, Oct. 1979.

[9] E. Rich, "Users are individuals: Individualizing user models," *Int. J. Man-Mach. Stud.*, vol. 18, no. 3, pp. 199–214, 1983.

[10] 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, Jun. 2000, pp. 195–204.

[11] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1983.

[12] 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.

[13] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Berkeley, CA, USA, Aug. 1999, pp. 50–57.

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

[15] H. Zhang, T. W. S. Chow, and M. K. M. Rahman, "A new dual wing harmonium model for document retrieval," *Pattern Recognit.*, vol. 42, no. 11, pp. 2950–2960, 2009.

[16] A. Schenker, M. Last, H. Bunke, and A. Kandel, "Classification of Web documents using graph matching," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 18, no. 3, pp. 475–496, 2004.

[17] M. Fuketa, S. Lee, T. Tsuji, M. Okada, and J.-I. Aoe, "A document classification method by using field association words," *Inf. Sci.*, vol. 126, nos. 1–4, pp. 57–70, 2000.

[18] C.-M. Tan, Y.-F. Wang, and C.-D. Lee, "The use of bigrams to enhance text categorization," *Inf. Process. Manage., Int. J.*, vol. 38, no. 4, pp. 529–546, 2002.

[19] M.-L. Antonie and O. R. Zaiane, "Text document categorization by term association," in *Proc. IEEE Int. Conf. Data Mining*, Maebashi, Japan, Dec. 2002, pp. 19–26.

[20] 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.

[21] T. W. S. Chow, H. Zhang, and M. K. M. Rahman, "A new document representation using term frequency and vectorized graph connectionists with application to document retrieval," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12023–12035, 2009.

[22] M. K. M. Rahman, W. P. Yang, T. W. S. Chow, and S. Wu, "A flexible multi-layer self-organizing map for generic processing of tree-structured data," *Pattern Recognit.*, vol. 40, no. 5, pp. 1406–1424, 2007.

[23] T. Heck, I. Peters, and W. G. Stock, "Testing collaborative filtering against co-citation analysis and bibliographic coupling for academic author recommendation," in *Proc. 3rd ACM Workshop Rec. Syst. Social Web (RecSys)*, Chicago, IL, USA, Jul. 2011, pp. 16–23.

[24] P. C. Vaz, D. M. de Matos, B. Martins, and A. Calado, "Improving a hybrid literary book recommendation system through author ranking," in *Proc. 12th ACM/IEEE-CS Joint Conf. Digit. Libraries*, Washington, DC, USA, Jun. 2012, pp. 387–388.

[25] H. Petry, P. Tedesco, V. Vieira, and A. C. Salgado, "ICARE: A context-sensitive expert recommendation system," in *Proc. 18th Eur. Conf. Artif. Intell.*, Patras, Greece, Jul. 2008, pp. 53–58.

[26] 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.

[27] E. Høgh-Rasmussen. (2005). *BBTools—A MATLAB Toolbox for Black-Box Computations*. Neurobiology Research Unit, Copenhagen University Hospital. [Online]. Available: http://nru.dk/software/bbtools/

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

[29] M. E. J. Newman, *Networks: An Introduction*. New York, NY, USA: Oxford Univ. Press, 2010.

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

[31] B. Fritzke, "A growing neural gas network learns topologies," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 7. 1995, pp. 625–632.

[32] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.

[33] J. A. F. Costa and M. L. de Andrade Netto, "A new tree-structured self-organizing map for data analysis," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 3. Washington, DC, USA, Jul. 2001, pp. 1931–1936.

[34] E. Cuadros-Vargas and R. A. F. Romero, "A SAM-SOM family: Incorporating spatial access methods into constructive self-organizing maps," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2. Honolulu, HI, USA, May 2002, pp. 1172–1177.
[35] M. R. Vieira, C. Traina, Jr., F. J. T. Chino, and A. J. M. Traina, "DBM-tree: Trading height-balancing for performance in metric access methods," *J. Brazilian Comput. Soc.*, vol. 11, no. 3, pp. 37–51, 2006.
[36] C. A. Astudillo and B. J. Oommen, "Topology-oriented self-organizing maps: A survey," *Pattern Anal. Appl.*, vol. 17, no. 2, pp. 223–248, 2014.
[37] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.
[38] P. V. Gehler, A. D. Holub, and M. Welling, "The rate adapting Poisson model for information retrieval and object recognition," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, Jun. 2006, pp. 337–344.
[39] C. H. Teo and S. V. N. Vishwanathan, "Fast and space efficient string kernels using suffix arrays," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, Jun. 2006, pp. 929–936.

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.

He was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, from 2010 to 2011. Since 2012, he has been with the Shenzhen Graduate School, 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, pattern recognition, evolutionary computing, and communication networks.

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

He has been involved in different consultancy projects with the Mass Transit Railway, Kowloon–Canton Railway Corporation, Hong Kong. He has also conducted other collaborative projects with Hong Kong Electric Company, Ltd., Hong Kong, MTR, Hong Kong, and Observatory Hong Kong, Hong Kong, where he is involved in the application of neural networks for machine fault detection and forecasting. He is currently a Professor with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. He has authored or co-authored 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.

Prof. Chow received the best paper award in the IEEE Industrial Electronics Society Annual Meeting, Seville, Spain, in 2002.

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 has been with the National Research Council of Canada for ten years since 1995, where he became a Senior Research Officer and 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 the technical program committees and international advisory committees for many prestigious conferences.