2017

A Graph-Embedding Approach to Hierarchical Visual Word Mergence

Lei Wang
University of Wollongong, leiw@uow.edu.au

Lingqiao Liu
University of Adelaide, lingqiao.liu@anu.edu.au

Luping Zhou
University of Wollongong, lupingz@uow.edu.au

Publication Details

Wang, L., Liu, L. & Zhou, L. (2017). A Graph-Embedding Approach to Hierarchical Visual Word Mergence. IEEE Transactions on Neural Networks and Learning Systems, 28 (2), 308-320.

A Graph-Embedding Approach to Hierarchical Visual Word Mergence

Publication Details
Wang, L., Liu, L. & Zhou, L. (2017). A Graph-Embedding Approach to Hierarchical Visual Word Mergence. IEEE Transactions on Neural Networks and Learning Systems, 28 (2), 308-320.

A Graph-embedding Approach to Hierarchical Visual Word Mergence

Lei Wang, *Senior Member, IEEE,* Lingqiao Liu, *Member, IEEE,* and Luping Zhou, *Senior Member, IEEE*

***Abstract*—Appropriately merging visual words is an effective dimension reduction method for the bag-of-visual-words model in image classification. The approach of hierarchically merging visual words has been extensively employed because it gives a fully determined merging hierarchy. Existing supervised hierarchical merging methods take different approaches and realize the merging process with various formulations. In this paper, we propose a unified hierarchical merging approach built upon the graph-embedding framework. Our approach is able to merge visual words for any scenario where a "preferred" structure and an "undesired" structure are defined, and therefore can effectively attend to all kinds of requirements for the word-merging process. In terms of computational efficiency, we show that our algorithm can seamlessly integrate a fast search strategy developed in our previous work, and thus well maintain the state-of-the-art merging speed. To the best of our survey, the proposed approach is the first one that addresses hierarchical visual word mergence in such a flexible and unified manner. As demonstrated, it can maintain excellent image classification performance even after significant dimension reduction, and outperform all the existing comparable visual word merging methods. In a broad sense, our work provides an open platform for applying, evaluating and developing new criteria for hierarchical word-merging tasks.**

I. Introduction

During the past decade, the bag-of-visual-words (BoVW) model has achieved great success in image recognition and been applied to various vision applications. This model mimics the bag-of-words (BoW) model in text analysis. Conceptually, it regards each specific pattern of pixel intensities as a "visual word" and models an image by the distribution (for example, via a histogram) of various visual words therein. Usually, to effectively characterize diverse visual content, a large number of visual words have to be used, leading to a high-dimensional visual representation. This calls for effective dimension reduction approaches. Among them, the approach of merging visual words has recently attracted much attention.[1]

Merging words into word-clusters originates from text analysis [1]. It has the merits of obtaining more reliable estimate of word probabilities, reducing representation dimensions, and even improving classification accuracy [1], [2]. Also, as reported in [3], [4], merging words outperforms selecting words in terms of reducing dimensionality while maintaining classification performance. With the introduction of the BoW model to image recognition, several visual-word-merging algorithms have recently been developed [5], [6], [7], [8], [9], [10], [11]. Dimension reduction by merging visual words not only inherits the aforementioned merits, but can also well maintain the structure of a BoW model via a compact visual codebook. The latter will be difficult to achieve with a general-purpose dimension reduction method, which linearly combines all the words together. In addition, as shown in [12], dimension reduction by efficiently merging words can enjoy significant computational advantage over the general-purpose methods when the dimensions of image representation are very high. Among existing visual-word-merging algorithms, the approach of *hierarchically* merging words has attracted attention, because of its conceptual simplicity and the ability of giving a full merging hierarchy in a single run. It merges two words[2] at each level of the hierarchy via a predefined criterion [1], [2], [3], [5], [7], [8], [9], [10], [11]. Most of these algorithms focus on supervised learning. In order to maintain classification performance, they adopt various criteria that can be related to classification performance, including class conditional probability [5], mutual information loss [2], [3], [7], class separability [8], margin of separation [11] and so on. Differences among these criteria make these algorithms have to be implemented in their own specific ways.

In this paper, we aim to develop a hierarchical word mergence framework that is unified to a wide range of dimension reduction criteria. In specific, given a predefined large set of visual words, our goal is to hierarchically merge them into a small number of visual words, such that the lower-dimensional image representation obtained based on these new words can maximally maintain classification performance. In order to achieve our aim, we build the algorithm upon the graph-embedding framework [13], which has demonstrated its excellence on performance, openness and flexibility for dimension reduction, but has not been exploited for merging visual words. Taking advantage of this framework, our algorithm is able to merge visual words for any scenario where a "preferred" structure and an "undesired" structure are defined. This brings significant advantage on effectively attending to various requirements (say, what information shall be best preserved and what shall not) during the word-merging process. One obstacle here is the computational issue, that is, how to efficiently hierarchically merge a large number of visual words. By analyzing the criteria used by the graph-

This work was supported by Australian Research Council Linkage Grant LP0991757.

L. Wang and L. Zhou are with School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: {leiw, lupingz}@uow.edu.au).

Lingqiao Liu is with School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia (e-mail: liulq83@gmail.com).

[1]In this paper, the phrases "merging words", "word-merging", and "word mergence" are used for different context with the same meaning.

[2]Here, "merges two words" means that for each image, its features (e.g., the bins of a histogram) corresponding to the two words are summed. An example on hierarchically merging words is further provided in Section II.

embedding framework, we show that all of them can satisfy the conditions of a fast search strategy developed in our previous work [8], [14][3]. This interesting finding well mitigates the computational issue and ensures that the proposed approach still maintains the state-of-the-art merging speed. In addition, we further generalize our approach to work with additive kernels, an active topic in recent visual recognition research [15]. To the best of our survey, our approach is the first one that addresses hierarchical word mergence in such a flexible and unified manner.

Experimental study is conducted on multiple benchmark datasets to verify the effectiveness of the proposed approach. As observed, it can efficiently merge thousands of words and maintain excellent classification performance even after significant dimension reduction. More importantly, it can achieve better merging performance than all the supervised word-merging algorithms developed in the recent literature. In addition, we conduct an initial investigation of the potential of our graph-embedding framework for unsupervised hierarchical word mergence and obtain promising performance. At last, we summarize the contributions of this work, especially the improvements over our previous work [8], [14] from which the fast search strategy is adopted.

- We reveal that the class separability measure in [8], [14] is just a special case of the proposed graph-embedding approach. This was not done in those works;
- By using new merging criteria, the proposed approach achieves better classification performance than the class separability measure. It is also better or comparable to the current best algorithm in the literature, and has higher computational efficiency.
- The proposed approach represents a framework that has excellent flexibility and extendability for merging words in various tasks, while the existing works only focus on developing algorithms for specific settings.
- The proposed approach can be used to merge words in an unsupervised learning case, while the class separability measure in [8], [14] only works for the supervised case;
- The proposed approach can naturally incorporate additive kernels, which is not available to any of the existing comparable algorithms;
- We conduct extensive experimental study to compare all the existing supervised hierarchical word-merging algorithms to verify the advantages of the proposed approach.

II. Related work

We first give an example to explain the hierarchical word-merging process. Let's assume that a set of training images are available and that d visual words have been generated. Based on these words, each image can be represented as a histogram with each bin indicating the number of occurrences of a visual word in this image. We denote this histogram by $\mathbf{x} = [x_1, x_2, \cdots, x_d]^\top$, where x_i represents the ith bin. Merging two words means that the two corresponding bins are summed, and this is applied to the histogram of each image. For example, merging the 3rd and 5th words as a new word means that x_3 and x_5 are summed as a new bin for each image, with the original two bins removed. In this way, the dimensions of image representation (the histogram) are reduced from d to $(d-1)$. In practice, which two words will be merged is identified by a predefined merging criterion from all the possible $\frac{d(d-1)}{2}$ pairs. Repeating the above merging process on the obtained $(d-1)$-dimensional histogram will further reduce the dimensions to $(d-2)$. Keeping doing this gives rise to a hierarchical word-merging process. A graphical illustration of this process is provided in Figure 1.

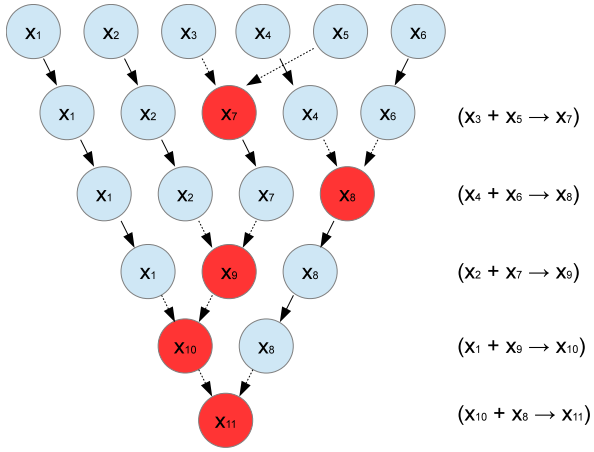


Fig. 1. A layer-by-layer illustration of a hierarchical word-merging process.

In image recognition with the BoVW model, the earliest work on supervised hierarchical word mergence is developed in [5], which merges visual words while maximally maintaining the posteriori probability of true class label for each sample. After that, we develop an algorithm based on class separability in [8] to maximally maintain the separability of different classes while merging visual words. In the same year, the authors of the work in [7] proposed to minimize the loss of mutual information between visual words and class label during the merging process. In the work in [11], we identify two key factors, class-conditional distribution model and the parameter estimate method, of the probabilistic formulation in [5]. On top of this, we generalize that probabilistic formulation to a framework that not only explains existing criteria such as those in [5], [7], [8], but can also guide the development of new criteria for merging words. In that work, we produce the current best supervised hierarchical word-merging algorithm. In addition to hierarchical mergence, an algorithm based on divisive information-theoretic clustering has recently been proposed for supervised word mergence in [16]. All of these algorithms will be compared with the proposed algorithm in the experiments of this paper.

Algorithms for the unsupervised case have also been seen in recent research. The work in [9] extends the supervised hierarchical algorithm in [7] by treating each sample as an individual class to minimize the mutual information loss. The work in [10] calculates the diffusion distance between visual words and groups them via k-means clustering. In our recent work, we utilize hashing technique to improve computational efficiency of word mergence for large visual data sets [12].

[3]This fast search strategy and the conditions are introduced in Section II.

We now review the class separability criterion in our previous work [8], especially its fast search strategy to be incorporated into the proposed approach later. Let $\mathbf{x}$ ($\mathbf{x} \in \mathbb{R}^l$) be the feature vector of a training sample represented with l words, e.g., a histogram of the number of occurrences of l visual words in an image. That work uses the class separability defined on the between-class and total scatter matrices [17]

$$\begin{aligned} \mathbf{S}_b &= \sum_{i=1}^{C} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^\top \\ \mathbf{S}_t &= \sum_{i=1}^{C}\sum_{j=1}^{n_i}(\mathbf{x}_{ij} - \mathbf{m})(\mathbf{x}_{ij} - \mathbf{m})^\top. \end{aligned} \tag{1}$$

where n_i is the number of training samples in the ith class and $n = \sum_{i=1}^{C} n_i$ is the total number of training samples. The sample mean of the ith class and the total mean are denoted by $\mathbf{m}_i$ and $\mathbf{m}$, respectively, and $\mathbf{x}_{ij}$ denotes the jth sample in class i. The class separability is measured by $\mathrm{tr}(\mathbf{S}_b^l)/\mathrm{tr}(\mathbf{S}_t^l)$, where $\mathrm{tr}(\cdot)$ is the trace of a square matrix and the superscript indicates the level of l. We found in [8] that once two words s and t are merged at level l, the resulting class separability at level $(l-1)$ can be quickly updated as

$$\mathrm{tr}(\mathbf{S}_b^{l-1}) = \mathrm{tr}(\mathbf{S}_b^l) + f_{st}; \quad \mathrm{tr}(\mathbf{S}_t^{l-1}) = \mathrm{tr}(\mathbf{S}_t^l) + g_{st}, \tag{2}$$

where f_{st} and g_{st} are two scalars depending only on words s and t. The values of f_{st} and g_{st} for each pair of s and t can be precomputed at the beginning of the merging process and quickly updated once two words are merged. The class separability after merging words s and t is expressed as

$$\mathcal{C}(s,t) = \frac{\mathrm{tr}(\mathbf{S}_b^l) + f_{st}}{\mathrm{tr}(\mathbf{S}_t^l) + g_{st}} = \frac{f_{st} - (-\mathrm{tr}(\mathbf{S}_b^l))}{g_{st} - (-\mathrm{tr}(\mathbf{S}_t^l))}. \tag{3}$$

The optimal pair of words to be merged at level l is regarded as the pair that maximizes this criterion value.

To swiftly identify the optimal pair of words, we developed a fast search strategy in [8]. To make the presentation self-contained, this strategy is illustrated in Figure 2. We define two points $P_1(g_{st}, f_{st})$ and $P_0(-\mathrm{tr}(\mathbf{S}_t^l), -\mathrm{tr}(\mathbf{S}_b^l))$. By doing so, we can interpret $\mathcal{C}(s,t)$ in Eq.(3) as the slope of a line $\overline{P_0P_1}$ (in red). Since each point P_1 associates with merging a specific pair of words, finding the optimal pair of words is equivalent to finding P_1 that gives the largest slope to line $\overline{P_0P_1}$. Note that the point $P_0(-\mathrm{tr}(\mathbf{S}_t^l), -\mathrm{tr}(\mathbf{S}_b^l))$ remains fixed during the search process at the level l.

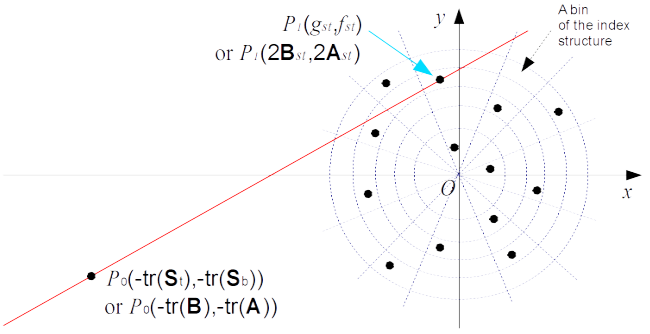


Fig. 2. The fast search strategy developed in our previous work [8].

The fast search strategy in [8] works as follows. The points (g_{st}, f_{st}) for every possible pair of s and t are precomputed at the beginning and indexed by a polar-coordinate-based structure (in blue dashed lines in the figure). For example, in Figure 2 the radial coordinate is quantized into five concentric circles and the polar angle from 0 to 2π is quantized into 16 segments, leading to $5 \times 16 = 80$ bins in total. A line passing the fixed point P_0 and tangent to the second-largest circle is firstly sought, partitioning the whole polar-coordinate-based index structure into two regions (region I is above the tangent line, and region II is below the tangent line). It is easy to observe that *any point P_1 in region I always gives $\overline{P_0P_1}$ a larger slope than a point in region II*. Therefore, all points in region II can be safely ignored as long as region I contains at least one point. Following this idea, the point P_1 producing the largest slope can be identified by merely examining the points in region I, which can be efficiently done via the pre-defined index structure. Note that there is no approximation in this fast search strategy, and the search result will be exactly same as the one obtained by a more timing-consuming exhaustive search. More details can be found in our previous work [8].

It is important to point out that, in addition to being able to be interpreted as a slope, the criterion $\mathcal{C}(s,t)$ must meet another key condition for the above strategy to work: $P_0(-\mathrm{tr}(\mathbf{S}_t^l), -\mathrm{tr}(\mathbf{S}_b^l))$ shall always reside outside of the cloud of $P_1(g_{st}, f_{st})$ on which the index structure is created. Otherwise, the above observation (in italic) will become untrue. The key condition can be satisfied as long as

$$-\mathrm{tr}\left(\mathbf{S}_t^l\right) \le g_{st} \text{ and } -\mathrm{tr}\left(\mathbf{S}_b^l\right) \le f_{st} \tag{4}$$

are true for any pair of s and t. These inequalities can be satisfied by the class separability criterion, because $\mathrm{tr}\left(\mathbf{S}_t^{l-1}\right)$ and $\mathrm{tr}\left(\mathbf{S}_b^{l-1}\right)$ in Eq.(2) are always non-negative due to the fact that $\mathbf{S}_t^{l-1}$ and $\mathbf{S}_b^{l-1}$ are positive semi-definite. This fast search strategy significantly helps to handle a large number of visual words. For example, as reported in [8], it can hierarchically merge 10000 words into two words in mere 90 seconds, while an exhaustive search will take more than two hours. Nevertheless, the strategy in [8] is only specifically designed for the class separability criterion. In the following, we integrate this strategy into the proposed new approach to handle more merging criteria.

III. Proposed Hierarchical Merging Approach

Before describing our approach, we would like to highlight that the word-merging algorithm in this work (and all the existing algorithms reviewed in Section II) is different from the feature coding algorithms or dictionary learning algorithms developed for the BoVW model in the literature [18], [19]. Feature coding or dictionary learning algorithms work with local invariant features, for example, the commonly used SIFT feature [20]. In contrast, word mergence is operated on the image-level representation (say, histograms of the number of occurrences of different words in an image) obtained after

feature coding and pooling steps. Also, its aim is to merge different visual words to generate a smaller-sized visual codebook, based on a given training image set. To obtain a merging hierarchy, word-merging algorithms do not need to access original local invariant features, which are however required by feature coding algorithms. At last, it is worth mentioning that compared with creating a smaller-sized codebook by setting a smaller k value in k-means clustering, word mergence can achieve much better classification performance, as have been shown in [7], [8].

A. The basic idea

Provided that a large set of d words (or equally, d feature dimensions[4]) are predefined, our aim is to *hierarchically merge the d words into p ($p \ll d$) new words such that when represented with the p new words, training samples can optimally preserve pre-defined "preferred" and "undesired" structures*. Since we take a hierarchical approach to merging words, the key question boils down to finding the two words s and t that should be merged at each level of the hierarchy.

For clarity, we call the top level of the hierarchy "level d", which includes the original d words. Let us assume that we have merged d words into l words, arriving at the level l. Recall that $\mathbf{x}$ ($\mathbf{x} \in \mathbb{R}^l$) denotes the feature vector of a training sample represented with these l words. Also, let $\mathbf{y}$ ($\mathbf{y} \in \mathbb{R}^{l-1}$) be the resulting feature vector of this sample after words s and t are merged. Their relationship can be conveniently expressed as a linear transform

$$\mathbf{y} = \mathbf{W}^\top \mathbf{x}, \tag{5}$$

where $\mathbf{W}$ is a $\{0,1\}$ matrix with the size of $l \times (l-1)$. It can be written as $[\mathbf{e}_1, \cdots, (\mathbf{e}_s + \mathbf{e}_t), \cdots, \mathbf{e}_l]$, where $\mathbf{e}_i$ is an l-dimensional unit vector with the ith entry being "1" only.

With the graph-embedding technique, it is convenient to express a structure over n training samples as follows. A weighted graph G with n vertexes is defined, where **each vertex corresponds to a training sample** $\mathbf{x}$. The edge weight is defined by a symmetric $n \times n$ weight matrix. In this paper, we use $\mathbf{P}_{n\times n}$ and $\mathbf{U}_{n\times n}$ to denote the weight matrices for the "preferred" and "undesired" structures, respectively. The weight values in each matrix reflect the information that we want to most or least preserve through the pair-wise distances of the samples embedded into a lower-dimensional Euclidean space, which will become clear soon.

Our algorithm uses a trace-ratio-based graph embedding criterion, which has been widely adopted in the literature [13], [21]. Working well with all criteria of this type will effectively demonstrate the "unified" characteristic of our algorithm. Our criterion for merging words is stated as: identify a pair of words s and t, by merging which the following ratio should be maximized,

$$\mathcal{C}(s,t) = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \cdot \mathbf{P}_{ij}}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \cdot \mathbf{U}_{ij}}, \tag{6}$$

[4]In the BoW model, each visual word corresponds to one dimension of final image-level representation. In the following parts, "word" can usually be equivalently understood as "dimension" from the context.

where $\mathbf{P}_{ij}$ and $\mathbf{U}_{ij}$ are the (i,j)-th entries of $\mathbf{P}$ and $\mathbf{U}$. Let $\mathbf{X}_{l\times n}$ be the matrix of n training samples represented by the l words, and $\mathbf{Y}_{(l-1)\times n}$ be the matrix represented by the $(l-1)$ words resulted from merging words s and t. Eq.(6) can be rewritten into a compact form as

$$\mathcal{C}(s,t) = \frac{\mathrm{tr}\left(\mathbf{Y}\mathbf{L}_\mathbf{P}\mathbf{Y}^\top\right)}{\mathrm{tr}\left(\mathbf{Y}\mathbf{L}_\mathbf{U}\mathbf{Y}^\top\right)} = \frac{\mathrm{tr}\left(\mathbf{W}^\top\mathbf{X}\mathbf{L}_\mathbf{P}\mathbf{X}^\top\mathbf{W}\right)}{\mathrm{tr}\left(\mathbf{W}^\top\mathbf{X}\mathbf{L}_\mathbf{U}\mathbf{X}^\top\mathbf{W}\right)}, \tag{7}$$

where $\mathbf{L}_\mathbf{P}$ and $\mathbf{L}_\mathbf{U}$ are the graph Laplacian of $\mathbf{P}$ and $\mathbf{U}$. They are defined as $\mathbf{L}_\mathbf{P} = \mathrm{diag}(\mathbf{P}\mathbf{1}) - \mathbf{P}$ and $\mathbf{L}_\mathbf{U} = \mathrm{diag}(\mathbf{U}\mathbf{1}) - \mathbf{U}$ [22], where $\mathbf{1}$ is a column vector consisting of all "1"s and $\mathrm{diag}(\mathbf{P}\mathbf{1})$ is a diagonal matrix whose diagonal elements are those in $\mathbf{P}\mathbf{1}$. Furthermore, by defining $\mathbf{A} \triangleq \mathbf{X}\mathbf{L}_\mathbf{P}\mathbf{X}^\top$ and $\mathbf{B} \triangleq \mathbf{X}\mathbf{L}_\mathbf{U}\mathbf{X}^\top$, Eq.(7) becomes

$$\mathcal{C}(s,t) \;=\; \frac{\mathrm{tr}\left(\mathbf{W}^\top\mathbf{A}\mathbf{W}\right)}{\mathrm{tr}\left(\mathbf{W}^\top\mathbf{B}\mathbf{W}\right)} \triangleq \frac{\mathrm{tr}\left(\mathbf{A}\right) + 2\mathbf{A}_{st}}{\mathrm{tr}\left(\mathbf{B}\right) + 2\mathbf{B}_{st}}, \tag{8}$$

where $\mathbf{A}_{st}$ is the (s,t)-th entry of $\mathbf{A}$. Note that two facts are used here: i) $\mathbf{W}$ has a special form of $[\mathbf{e}_1, \cdots, (\mathbf{e}_s + \mathbf{e}_t), \cdots, \mathbf{e}_l]$ and ii) $\mathbf{A}$ and $\mathbf{B}$ are symmetric. Given $\mathbf{L}_\mathbf{P}$ and $\mathbf{L}_\mathbf{U}$, both $\mathbf{A}$ and $\mathbf{B}$ only depend on $\mathbf{X}$, which is known and fixed at level l. Therefore, $\mathrm{tr}(\mathbf{A})$ and $\mathrm{tr}(\mathbf{B})$ remain constant during the search process at level l. The optimal pair of words can be obtained by finding the (s,t) pair that leads to the largest $\mathcal{C}(s,t)$ value.

Since Eq.(6) is the formulation commonly used by the graph-embedding technique for dimension reduction [13], [21], [23], the above result indicates that all the existing dimension reduction criteria used in graph-embedding methods could potentially be employed to guide hierarchical word mergence, serving various purposes in practical applications.

B. Class separability in [8] as a special case

It is easy to interpret our previous work [8] from the perspective of graph-embedding. Let $\mathbf{Z}_\mathbf{P}$ denote a matrix whose (i,j)-th entry is defined as

$$\mathbf{Z}_{\mathbf{P},ij} = \begin{cases} 1/n_c, & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are from class } c; \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where n_c is the number of training samples from class c and $\sum_c n_c = n$. Defining the preferred and undesired structures as $\mathbf{P} = \mathbf{1}/n - \mathbf{Z}_\mathbf{P}$ and $\mathbf{U} = \mathbf{1}/n$, we can verify that

$$\mathbf{A} = \mathbf{X}\mathbf{L}_\mathbf{P}\mathbf{X}^\top = \mathbf{S}_b; \quad \mathbf{B} = \mathbf{X}\mathbf{L}_\mathbf{U}\mathbf{X}^\top = \mathbf{S}_t. \tag{10}$$

where $\mathbf{S}_b$ and $\mathbf{S}_t$ are the scatter matrices defined in Eq.(1). Also, it can be shown that g_{st} and f_{st} defined before Eq.(3) equal $2\mathbf{B}_{st}$ and $2\mathbf{A}_{st}$, respectively. Identifying the work in [8] as a special case of the proposed approach helps revealing an important property of the class-separability-based merging method, which has not been noticed in [8]. That is, from the perspective of graph-embedding, that method essentially finds an embedding of the original image representations into a lower-dimensional Euclidean space. This property will be demonstrated in experimental study via the classification performance of the k-NN classifier with a Euclidean distance.

C. Supervised word mergence with the proposed approach

In this section, we aim to demonstrate the effectiveness of the graph-embedding approach proposed in Section III-A. In specific, we show that *with this approach, we can conveniently employ more sophisticated merging criteria than the class separability in [8], which helps to achieve better word-merging performance*. Supervised hierarchical word mergence is focused in this section.

As shown in Section II, the class separability measure used in [8] is built upon the between-class and total scatter matrices. From the definition of these matrices in Eq.(1), it can be known that they characterize the mean and covariance structure of data, implicitly assuming the Gaussianality for each class. However, this assumption is usually not satisfied in practice. In the literature, a number of variants of the class separability criterion have been developed to handle this issue by considering local data structure. Among them, nonparametric discriminant analysis (NDA) [24], marginal fisher analysis (MFA) [13] and local discriminant embedding (LDE) [25] may be the best known ones, and they share a similar spirit. In this work, we take the earliest criterion NDA, which was proposed three decades ago, to demonstrate the effectiveness of the proposed approach. That is, NDA will be integrated as a more sophisticated criterion into our graph-embedding approach to perform hierarchical word-mergence.

We firstly describe the basic idea of NDA as follows. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the training sample sets of two classes. NDA defines a "local" between-class scatter matrix as

$$\begin{aligned}\hat{\mathbf{S}}_b &= \sum_{\mathbf{x}_i\in\mathcal{D}_1}\left[\mathbf{x}_i-\mathbf{m}_2^k(\mathbf{x}_i)\right]\left[\mathbf{x}_i-\mathbf{m}_2^k(\mathbf{x}_i)\right]^\top \\ &+ \sum_{\mathbf{x}_i\in\mathcal{D}_2}\left[\mathbf{x}_i-\mathbf{m}_1^k(\mathbf{x}_i)\right]\left[\mathbf{x}_i-\mathbf{m}_1^k(\mathbf{x}_i)\right]^\top,\end{aligned} \quad (11)$$

where $\mathbf{m}_2^k(\mathbf{x}_i)$ is the mean of the k nearest neighbors found in $\mathcal{D}_2$ for $\mathbf{x}_i$ in $\mathcal{D}_1$. A similar definition applies to $\mathbf{m}_1^k(\mathbf{x}_i)$ for $\mathbf{x}_i$ in $\mathcal{D}_2$. Note that the "local" property is achieved through the use of $\mathbf{m}_2^k(\mathbf{x}_i)$ and $\mathbf{m}_1^k(\mathbf{x}_i)$, which are the mean computed within a local neighborhood of $\mathbf{x}_i$ instead of from a whole class. This arrangement makes the NDA criterion more effective than the class separability [8] in handling non-Gaussian data distribution. Through the proposed graph-embedding approach, we are able to conveniently untilize the advantage of NDA to improve word-merging performance.

We firstly show that from the perspective of graph-embedding, the NDA criterion essentially defines a "preferred" structure over data. Let $\mathbf{Z_P}$ be a matrix whose (i,j)-th entry is defined as

$$\mathbf{Z_P}_{,ij}=\begin{cases}1/k, & \text{if } \mathbf{x}_j\in\mathcal{N}_{\setminus i}^k(\mathbf{x}_i);\\ 0, & \text{otherwise,}\end{cases} \quad (12)$$

where $\mathcal{N}_{\setminus i}^k(\mathbf{x}_i)$ denotes the k nearest neighbors found in the class other than that of $\mathbf{x}_i$. Through some derivation (see the Appendix), the weight matrix $\mathbf{P}$ (in Eq.(6)) for the preferred structure implicitly defined by NDA can be expressed as

$$\mathbf{P}=\mathbf{Z_P}+\mathbf{Z_P}^\top-\mathbf{Z_P}^\top\mathbf{Z_P}. \quad (13)$$

In this way, we can rewrite $\hat{\mathbf{S}}_b$ into a form readily used by the graph-embedding approach as $\mathbf{A}=\mathbf{X}\mathbf{L_P}\mathbf{X}^\top$. The equivalence of $\mathbf{A}$ and $\hat{\mathbf{S}}_b$ in Eq.(11) can be seen from the derivation in the Appendix.

Secondly, to be consistent with $\hat{\mathbf{S}}_b$, we also develop a "local" total scatter matrix which is defined as

$$\hat{\mathbf{S}}_t = \frac{1}{k'}\sum_{\mathbf{x}_i\in\mathcal{D}}\sum_{\mathbf{x}_j\in\mathcal{N}^{k'}(\mathbf{x}_i)}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^\top, \quad (14)$$

where $\mathcal{N}^{k'}(\mathbf{x}_i)$ denotes the k' nearest neighbors of $\mathbf{x}_i$ in the whole training set $\mathcal{D}$. Again, we can derive (see the Appendix) that this is equivalent to defining an "undesired" structure. And the weight matrix $\mathbf{U}$ (in Eq.(6)) for this undesired structure is

$$\mathbf{U}=\mathbf{Z_U}+\mathbf{Z_U}^\top-\text{diag}(\mathbf{Z_U}^\top\mathbf{1}_{n\times 1}), \quad (15)$$

with the (i,j)-th entry of $\mathbf{Z_U}$ defined as

$$\mathbf{Z_U}_{,ij}=\begin{cases}1/k', & \text{if } \mathbf{x}_j\in\mathcal{N}^{k'}(\mathbf{x}_i);\\ 0, & \text{otherwise.}\end{cases} \quad (16)$$

Similarly, we can then rewrite $\hat{\mathbf{S}}_t$ as $\mathbf{B}=\mathbf{X}\mathbf{L_U}\mathbf{X}^\top$. The equivalence of $\mathbf{B}$ and $\hat{\mathbf{S}}_t$ in Eq.(14) can also be seen from the derivation in the Appendix.

In sum, to implement hierarchical visual word mergence, we compute the matrices $\mathbf{P}$ and $\mathbf{U}$ on a given training set and follow the criterion in Eqs. (6)-(8) to identify the optimal pair of words s and t at each level of the hierarchy.

D. Unsupervised word mergence with the proposed approach

The proposed graph-embedding approach can also be applied to unsupervised hierarchical word mergence, although this paper is focused on the supervised case. In the following part, we demonstrate this property through a Laplacian-based criterion widely used in the literature.

Locality preserving projection (LPP) [26] is a well-known Laplacian-based criterion and has been applied to dimension reduction in the unsupervised case. This criterion considers the underlying manifold structure of high-dimensional data to find better Euclidean embedding for them. This is also implemented in the way of modelling the local neighborhood of each sample. Since LPP has been widely used, we do not reiterate its technical details in this paper and readers are referred to the original work [26]. In short, LPP can also be expressed in the ratio form in Eq.(6) and the corresponding preferred and undesired structures are shown as follows.

Let $\kappa(\mathbf{x}_i,\mathbf{x}_j)$ be a similarity function of two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, e.g., a heat kernel function used in [26]. The undesired structure of LPP can be expressed as (see the Appendix)

$$\mathbf{U}_{ij} = \begin{cases}\kappa(\mathbf{x}_i,\mathbf{x}_j), & \text{if } \mathbf{x}_j\in\mathcal{N}_k(\mathbf{x}_i)\\ & \text{or vice versa;}\\ 0, & \text{otherwise.}\end{cases} \quad (17)$$

Penalizing the presence of this structure ensures that for two samples having larger similarity, their Euclidean embeddings $\mathbf{y}_i$ and $\mathbf{y}_j$ will have a shorter Euclidean distance. Also, the preferred structure can be expressed as (see the Appendix)

$$\mathbf{P}=\frac{\mathbf{U}\mathbf{1}_{n\times 1}(\mathbf{U}\mathbf{1}_{n\times 1})^\top}{\mathbf{1}_{n\times 1}\mathbf{U}\mathbf{1}_{n\times 1}}. \quad (18)$$

It models the data variance information. Maintaining this structure avoids the samples from being collapsed into a single point after Euclidean embedding. In this way, unsupervised word mergence can also be implemented through the proposed graph-embedding approach by computing $\mathbf{P}$ and $\mathbf{U}$ matrices on a given sample set. An initial investigation in this regard will be conducted in the experiment. For clarity, we summarize $\mathbf{P}$ and $\mathbf{U}$ matrices for CSM, NDA and LPP in the Appendix.

E. Computational issue and the fast search strategy

The proposed algorithm mainly has two parts of computations. One is to construct the preferred and undesired structures, which needs to conduct nearest neighbor search. When the number of samples is large, we can utilize fast or approximate nearest neighbor search techniques, which have been well studied in the literature [27]. The other is to hierarchically merge words. This is specific to the proposed algorithm and therefore focused in the following discussion.

Recall that there are n training samples represented by d words. For computing the symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, the time complexity is at most $\mathcal{O}(n^2d + nd^2)$. Saving them into memory results in the space complexity of $\mathcal{O}(d(d-1))$. When d is as large as 10000 and the format is double-precision, this needs 800MB memory. This memory requirement can be well met by common desktop computers. The main problem is the computational load of hierarchical mergence. Exhaustively searching for the optimal words at every single level shall be avoided, especially when d is large. In the following part, we show that *our graph-embedding-based approach can work seamlessly with the fast search strategy in [8] and well maintain the state-of-the-art merging speed*. Recall that the fast search strategy has been illustrated in Figure 2.

From Eq.(8), we see that $\mathcal{C}(s,t)$ can be interpreted as *the slope of the line passing a fixed point* $P_0(-\mathrm{tr}(\mathbf{B}), -\mathrm{tr}(\mathbf{A}))$ *and a point* $P_1(2\mathbf{B}_{st}, 2\mathbf{A}_{st})$*, which corresponds to the two words to be merged*. Now, to find the optimal words s and t, we only need to *find the optimal point* $P_1(2\mathbf{B}_{st}, 2\mathbf{A}_{st})$ *that produces the maximum slope with respect to the fixed point* P_0. This geometric interpretation is the same as the one upon which the fast search strategy is designed. In addition, as indicated at the end of Section II, to utilize that strategy the proposed approach must meet the key condition: i) $2\mathbf{A}_{st} \geq -\mathrm{tr}(\mathbf{A})$ and ii) $2\mathbf{B}_{st} \geq -\mathrm{tr}(\mathbf{B})$. They can be readily proven for our approach. Because a Laplacian matrix is always positive semi-definite, we can obtain that $\mathbf{Y}_{(i,:)}\mathbf{L}_\mathbf{P}\mathbf{Y}_{(i,:)}^\top \geq 0$, where $\mathbf{Y}_{(i,:)}$ denotes the i-th row of the matrix $\mathbf{Y}$. As a result,

$$\begin{aligned} & \mathrm{tr}\left(\mathbf{Y}\mathbf{L}_\mathbf{P}\mathbf{Y}^\top\right) \geq 0 \qquad (19) \\ \Longleftrightarrow \quad & \mathrm{tr}(\mathbf{A}) + 2\mathbf{A}_{st} \geq 0 \Longleftrightarrow 2\mathbf{A}_{st} \geq -\mathrm{tr}(\mathbf{A}). \end{aligned}$$

Similarly, we can obtain $2\mathbf{B}_{st} \geq -\mathrm{tr}(\mathbf{B})$. Therefore, it can be concluded that our approach can readily utilize the fast search strategy. This mitigates the computational issue and allows our approach to effectively handle a reasonably large number of words as the state-of-the-art methods. Note that compared with our previous work [8] specifically designed for the class separability criterion, the proposed approach can use matrices $\mathbf{A}$ and $\mathbf{B}$ to uniformly accommodate various dimension reduction criteria. The proposed algorithm is outlined in Table I.

TABLE I
THE PROPOSED GRAPH-EMBEDDING-BASED ALGORITHM FOR SUPERVISED HIERARCHICAL WORD MERGENCE

Input: n training samples represented as $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \cdots, c\}$ (or $y_i \in \emptyset$ for the unsupervised case).
d is the number of words to be merged, y_i the class label of $\mathbf{x}_i$ and p the number of words after word mergence.
Output: The $(d-p)$-level hierarchy of merging words.

Initialization:
define the preferred structure $\mathbf{P}$ and the undesired structure $\mathbf{U}$.
compute the matrices $\mathbf{A}$ and $\mathbf{B}$ in Eq.(8) at the level d and store them in memory.
index the $\frac{d(d-1)}{2}$ points of $P_1(2\mathbf{B}_{st}, 2\mathbf{A}_{st})$ to prepare for the fast search strategy in [8].
compute $P_0(-\mathrm{tr}(\mathbf{B}), -\mathrm{tr}(\mathbf{A}))$.

Merging operation:
for $l = d, d-1, \cdots, p$
(1) **fast search** for the point P_1 that gives the line $\overline{P_1P_0}$ the largest slope. Mark the optimal pair of words s and t.
(2) **update** point $P_0(-\mathrm{tr}(\mathbf{B}), -\mathrm{tr}(\mathbf{A}))$ as:
$\mathrm{tr}(\mathbf{A}) := \mathrm{tr}(\mathbf{A}) + 2\mathbf{A}(s,t)$;
$\mathrm{tr}(\mathbf{B}) := \mathrm{tr}(\mathbf{B}) + 2\mathbf{B}(s,t)$;
(3) **update** $\mathbf{A}$ and $\mathbf{B}$ after merging word t into word s:
$\mathbf{A}(s,i) := \mathbf{A}(s,i) + \mathbf{A}(t,i)$; $\mathbf{A}(i,s) := \mathbf{A}(s,i)$; $(1 \leq i \leq l, \quad i \neq s,t)$
$\mathbf{B}(s,i) := \mathbf{B}(s,i) + \mathbf{B}(t,i)$; $\mathbf{B}(i,s) := \mathbf{B}(s,i)$; $(1 \leq i \leq l, \quad i \neq s,t)$
remove $\mathbf{A}(t,i)$, $\mathbf{A}(i,t)$, $\mathbf{B}(t,i)$, and $\mathbf{B}(i,t)$; $(1 \leq i \leq l)$
(4) **index** the newly added points $P_1(2\mathbf{B}_{si}, 2\mathbf{A}_{si})$ $(1 \leq i \leq l-1, \quad i \neq s)$
end

TABLE II
ADDITIVE KERNELS NATURALLY HANDLED BY OUR ALGORITHM

Kernel type	$k(x_i, z_i)$
Histogram intersection	$\min(x_i, z_i)$
Hellinger's	$\sqrt{x_i z_i}$
Chi-square	$\frac{2x_i z_i}{x_i + z_i}$
Jensen-Shannon	$\frac{x_i}{2}\log_2\frac{x_i+z_i}{x_i} + \frac{z_i}{2}\log_2\frac{x_i+z_i}{z_i}$
Linear	$x_i z_j$

F. Extension to the (additive) kernelized version

We can readily incorporate the kernels that work better with histograms or the alike encountered in the BoVW model into our approach. Especially, as listed in Table II, the *additive* kernels have recently attracted much attention and demonstrated promising performance in visual recognition [15]. For all of them, the kernel function between two samples $\mathbf{x}$ and $\mathbf{z}$ can be expressed as $k(\mathbf{x}, \mathbf{z}) = \sum_i k(x_i, z_i)$, where $k(x_i, z_i)$ denotes a kernel defined on their ith components.

Recall that in our approach, the preferred and undesired structures are to be preserved (most or least) in the Euclidean space of $\mathbf{y}$ obtained after merging two words. To better measure the quality of preservation, it is desirable to evaluate the criterion in Eq.(6) in a kernel-induced feature space. This motivates us to generalize our approach to incorporate the additive kernels.

Let $\mathbf{Y}_\phi = [\phi(\mathbf{y}_1),\ \phi(\mathbf{y}_2),\ \cdots,\ \phi(\mathbf{y}_n)]$ be the data matrix obtained by a nonlinear, implicit mapping $\phi(\cdot)$ induced by an

additive kernel. The ith column of $\mathbf{Y}_\phi$ is $\phi(\mathbf{y}_i)$, which denotes the mapping of $\mathbf{y}_i$. In this case, Eq.(7) can be rewritten as

$$\begin{aligned}\mathcal{C}(s,t) &= \frac{\mathrm{tr}\left(\mathbf{Y}_\phi \mathbf{L_P} \mathbf{Y}_\phi^\top\right)}{\mathrm{tr}\left(\mathbf{Y}_\phi \mathbf{L_U} \mathbf{Y}_\phi^\top\right)} = \frac{\mathrm{tr}\left(\mathbf{L_P K_Y}\right)}{\mathrm{tr}\left(\mathbf{L_U K_Y}\right)} \qquad (20)\\ &= \frac{\mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right) - \left(-\mathrm{tr}\left(\mathbf{L_P K_X}\right)\right)}{\mathrm{tr}\left(\mathbf{L_U}\tilde{\mathbf{K}}_{st}\right) - \left(-\mathrm{tr}\left(\mathbf{L_U K_X}\right)\right)},\end{aligned}$$

where $\mathbf{K_Y} \triangleq \mathbf{Y}_\phi^\top \mathbf{Y}_\phi$ is an $n \times n$ kernel matrix computed with $\mathbf{Y}_\phi$, and its (i,j)-th entry is $\langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j)\rangle = k(\mathbf{y}_i, \mathbf{y}_j)$. The second equality of Eq.(20) is achieved due to the property of $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$ for multipliable matrices. Also, let $\mathbf{K_X}$ be the $n \times n$ kernel matrix computed with the data matrix $\mathbf{X}$. We use $\tilde{\mathbf{K}}_{st}$ to denote the difference of $\mathbf{K_Y}$ and $\mathbf{K_X}$, that is,

$$\tilde{\mathbf{K}}_{st} = \mathbf{K_Y} - \mathbf{K_X}. \qquad (21)$$

Replacing $\mathbf{K_Y}$ with $\tilde{\mathbf{K}}_{st} + \mathbf{K_X}$ gives the third equality in Eq.(20).

Finding the optimal words at level l can therefore still be interpreted as finding the maximum slope of $\overline{P_1 P_0}$, where points P_0 and P_1 now become $P_0\left(-\mathrm{tr}\left(\mathbf{L_U K_X}\right), -\mathrm{tr}\left(\mathbf{L_P K_X}\right)\right)$ and $P_1\left(\mathrm{tr}\left(\mathbf{L_U}\tilde{\mathbf{K}}_{st}\right), \mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right)\right)$, respectively. Also, because the Laplacian matrix $\mathbf{L_P}$ is positive semi-definite, it can be obtained that $\mathrm{tr}\left(\mathbf{Y}_\phi \mathbf{L_P} \mathbf{Y}_\phi^\top\right) \geq 0$ and therefore we have $\mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right) \geq -\mathrm{tr}\left(\mathbf{L_P K_X}\right)$. This applies to $\mathbf{L_U}$ too. Hence, it can again be guaranteed that point P_0 is always outside of the cloud of points P_1.

Now we need to find a way to quickly evaluate $\tilde{\mathbf{K}}_{st}$ for each pair of words s and t. Recall that $k(\mathbf{x}, \mathbf{z}) = \sum_i k(x_i, z_i)$. With this property, we can expand the kernel matrix $\mathbf{K_X}$ as

$$\mathbf{K_X} = \sum_i \mathbf{K}_i, \qquad (22)$$

where $\mathbf{K}_i$ denotes the kernel matrix computed with the ith row of $\mathbf{X}$. As mentioned above, $\mathbf{Y}$ is the matrix represented by the $(l-1)$ words resulted from merging words s and t. Therefore, $\mathbf{Y}$ can be obtained by deleting the sth and rth rows of $\mathbf{X}$ and then inserting another row that is the sum of the two rows. Applying this to $\mathbf{K_Y}$ (defined immediately after Eq.(20)) leads to

$$\begin{aligned}&\mathbf{K_Y} = \mathbf{K_X} - \mathbf{K}_s - \mathbf{K}_t + \mathbf{K}_{st} \qquad (23)\\ \Longleftrightarrow \quad &\tilde{\mathbf{K}}_{st} = \mathbf{K}_{st} - \mathbf{K}_s - \mathbf{K}_t,\end{aligned}$$

where $\mathbf{K}_{st}$ denotes the kernel matrix computed with the sum of the sth and rth rows of $\mathbf{X}$.

With the result of Eq.(23), we can expand $\mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right)$ as

$$\mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right) = \mathrm{tr}\left(\mathbf{L_P K}_{st}\right) - \mathrm{tr}\left(\mathbf{L_P K}_s\right) - \mathrm{tr}\left(\mathbf{L_P K}_t\right). \qquad (24)$$

Also, from the result of Eq.(22), $\mathrm{tr}\left(\mathbf{L_P K_X}\right)$ can be expanded as $\sum_i \mathrm{tr}\left(\mathbf{L_P K}_i\right)$. Therefore, by precomputing the building blocks $\mathrm{tr}\left(\mathbf{L_P K}_i\right)$ $(i = 1, 2, \cdots, d)$ at the beginning and iteratively updating the four traces at the second row in Eq.(20), we can obtain the kernelized version of the fast hierarchical word-merging algorithm in Table III.

To be consistent with the use of matrices $\mathbf{A}$ and $\mathbf{B}$ in Table I, we define matrices $\mathbf{A}'$ and $\mathbf{B}'$ as

$$\mathbf{A}'_{st} = \mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right); \quad \mathbf{B}'_{st} = \mathrm{tr}\left(\mathbf{L_U}\tilde{\mathbf{K}}_{st}\right), \qquad (25)$$

where $1 \leq s, t \leq d$. As highlighted by the underlines in Table III, the main differences of the kernelized version from the non-kernel version in Table I lie at the way of updating $\mathbf{A}'$ and $\mathbf{B}'$. Due to the use of kernel function, the entries $\mathbf{A}'(s,i)$ and $\mathbf{B}'(s,i)$ cannot be conveniently calculated as before. Instead, the kernel matrix $\mathbf{K}_{si}$ has to be dynamically computed to obtain $\mathrm{tr}\left(\mathbf{L_P K}_{si}\right)$ and $\mathrm{tr}\left(\mathbf{L_U K}_{si}\right)$. At the same time, to speed up computation we maintain a list of $\mathrm{tr}\left(\mathbf{L_P K}_i\right)$ and $\mathrm{tr}\left(\mathbf{L_U K}_i\right)$ for all the remaining words and dynamically update it, as shown in the first underscored line.

As a hierarchical word-merging algorithm that can uniformly handle various additive kernels, the proposed algorithm is faster and more efficient when compared with a *direct* implementation of the kernelized version. Certainly, when compared with the non-kernel version described in Table I, this kernelized version will incur higher computational complexity as expected. For example, the space complexity is mildly increased to $\mathcal{O}(n(n-1)+2d)$ because $\mathbf{L_P}$, $\mathbf{L_U}$, $\mathrm{tr}\left(\mathbf{L_P K}_i\right)$ and $\mathrm{tr}\left(\mathbf{L_U K}_i\right)$ $(i = 1, 2, \cdots, d)$ need to be stored in memory during the course. The time complexity rises significantly because a term in the form of $\mathrm{tr}\left(\mathbf{LK}\right)$ needs to be frequently calculated. The complexity to compute $\mathbf{A}'$ and $\mathbf{B}'$ at the initialization stage becomes $\mathcal{O}(d(d-1)n(n+1))$. Also, computing $\mathrm{tr}\left(\mathbf{L_P K}_i\right)$ and $\mathrm{tr}\left(\mathbf{L_U K}_i\right)$ for $i = 1, 2, \cdots, d$ incurs the complexity of $\mathcal{O}(2n(n+1)d)$. In addition, at each level of the hierarchy, the complexity of steps (2), (3) and (4) goes up to $\mathcal{O}(2n(n+1))$, $\mathcal{O}(2n(n+1))$ and $\mathcal{O}(2n(n+1)l)$, respectively, instead of being trivially computed as in Table I.

G. Discussion

By hierarchically merging words, the proposed approach produces a hierarchical tree structure as any other hierarchical clustering algorithms. Hierarchical tree structure has been frequently seen in the literature of computer vision, and one of the examples is the vocabulary tree in [28]. The major differences of the vocabulary tree and the proposed hierarchical word-mergence approach are summarized as follows.

Firstly, they work in different spaces and deal with different things. Vocabulary tree employs hierarchical k-means clustering to quantize local region descriptors, e.g., the SIFT used in [28]. Our proposed approach employs hierarchical feature merging to cluster different feature dimensions, e.g., different bins of a histogram-based representation; Secondly, their goals are different. Vocabulary tree is designed to efficiently use a large number of visual words to improve retrieval performance. In contrast, our work aims to reduce the feature dimensions (e.g., the number of visual words) while maximally preserving classification performance. In addition, the difference of goals also results in different criteria. The quantization error is used in [28], whereas various graph-based criteria are used in our work. At the same time, the two approaches can be connected from the perspective that both of them have a hierarchical tree structure. More importantly, the work in [28] has efficiently

TABLE III
THE PROPOSED GRAPH-EMBEDDING-BASED ALGORITHM (THE (ADDITIVE-) KERNELIZED VARIANT)

Input: n training samples represented as $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \cdots, c\}$ (or $y_i \in \emptyset$ for the unsupervised case).
d is the number of words to be merged, y_i the class label of $\mathbf{x}_i$ and p the number of words after hierarchical word mergence.
Output: The $(d-p)$-level hierarchy of merging words.

Initialization:
define the preferred and undesired structures $\mathbf{P}$ and $\mathbf{U}$.
compute $\mathbf{L_P}$, $\mathbf{L_U}$ and the matrices $\mathbf{A}'$ and $\mathbf{B}'$ in Eq.(25) at the level d and store them in memory.
compute $\mathrm{tr}\left(\mathbf{L_P}\mathbf{K}_i\right)$ and $\mathrm{tr}\left(\mathbf{L_U}\mathbf{K}_i\right)$ for $i = 1, 2, \cdots, d$ and store them in memory.
index the $\frac{d(d-1)}{2}$ points of $P_1\left(\mathbf{B}'_{st}, \mathbf{A}'_{st}\right)$ to prepare for the fast search strategy in [8].
compute $P_0\left(-\mathrm{tr}\left(\mathbf{L_U}\mathbf{K_X}\right), -\mathrm{tr}\left(\mathbf{L_P}\mathbf{K_X}\right)\right)$ at the level d.

Merging operation:
for $l = d, d-1, \cdots, p$
(1) **fast search** for the point P_1 that gives the line $\overline{P_1P_0}$ the largest slope. Mark the optimal pair of words s and t.
(2) **update** point $P_0\left(-\mathrm{tr}\left(\mathbf{L_U}\mathbf{K_X}\right), -\mathrm{tr}\left(\mathbf{L_P}\mathbf{K_X}\right)\right)$ as:
$\mathrm{tr}\left(\mathbf{L_U}\mathbf{K_X}\right) := \mathrm{tr}\left(\mathbf{L_U}\mathbf{K_X}\right) + \mathrm{tr}\left(\mathbf{L_U}\tilde{\mathbf{K}}_{st}\right)$;
$\mathrm{tr}\left(\mathbf{L_P}\mathbf{K_X}\right) := \mathrm{tr}\left(\mathbf{L_P}\mathbf{K_X}\right) + \mathrm{tr}\left(\mathbf{L_P}\tilde{\mathbf{K}}_{st}\right)$;
(3) **update** $\mathrm{tr}\left(\mathbf{L_P}\mathbf{K}_s\right)$ and $\mathrm{tr}\left(\mathbf{L_U}\mathbf{K}_s\right)$ after merging word t into word s. **Remove** $\mathrm{tr}\left(\mathbf{L_P}\mathbf{K}_t\right)$ and $\mathrm{tr}\left(\mathbf{L_U}\mathbf{K}_t\right)$.
(4) **update** $\mathbf{A}'$ and $\mathbf{B}'$ by using Eq.(24):
$\mathbf{A}'(s,i) := \mathrm{tr}\left(\mathbf{L_P}\mathbf{K}_{si}\right) - \mathrm{tr}\left(\mathbf{L_P}\mathbf{K}_s\right) - \mathrm{tr}\left(\mathbf{L_P}\mathbf{K}_i\right)$;
$\mathbf{A}'(i,s) := \mathbf{A}'(s,i)$; $\quad (1 \le i \le l, \quad i \ne s, t)$
$\mathbf{B}'(s,i) := \mathrm{tr}\left(\mathbf{L_U}\mathbf{K}_{si}\right) - \mathrm{tr}\left(\mathbf{L_U}\mathbf{K}_s\right) - \mathrm{tr}\left(\mathbf{L_U}\mathbf{K}_i\right)$;
$\mathbf{B}'(i,s) := \mathbf{B}'(s,i)$; $\quad (1 \le i \le l, \quad i \ne s, t)$
remove $\mathbf{A}'(t,i)$, $\mathbf{A}'(i,t)$, $\mathbf{B}'(t,i)$, and $\mathbf{B}'(i,t)$; $(1 \le i \le l)$
(5) **re-index** the newly added points $P_1(\mathbf{B}'_{si}, \mathbf{A}'_{si})$ $(1 \le i \le l-1, \quad i \ne s)$
end

utilized this structure to improve retrieval. We are going to explore this direction too in the future work to investigate its efficiency for dimension reduction.

The proposed word-mergence approach can also be linked with constrained spectral clustering [29], [30]. Above all, both of them are clustering methods. Furthermore, the clustering process of both methods is restricted or guided by the pairwise relationship among sample points. The pairwise constrains in constrained spectral clustering are usually expressed as "must-link" or "cannot-link" and reflected in an affinity matrix. For the word-mergence method, this pairwise relationship is obtained from the class labels in the supervised case or a predefined similarity function in the unsupervised case. The pairwise relationship is reflected in the weight matrices $\mathbf{P}$ and $\mathbf{U}$, which can also be viewed as affinity matrices. In this sense, the affinity matrix in constrained spectral clustering can also be utilized by the word-mergence method.

On the other hand, the two clustering methods deal with different objects and have different goals. Constrained spectral clustering clusters *sample points* into a number of clusters to explore the underlying structure of data. In contrast, the proposed word-mergence method clusters *features* into a number of clusters to reduce the dimensions of the representation of sample points.

IV. EXPERIMENTAL RESULT

A. Experimental setting

For the supervised case, our algorithm is compared with a set of recently developed supervised word-merging algorithms using different merging criteria, including class separability (CSM) [8], mutual information loss (AIB) [2], [7], margin of separation (MME) and multinomial distribution with Bayesian estimation (MLT) [11], class conditional probability (UVD) [5] and divisive information-theoretic clustering (DIT) [16]. Like our algorithm, the first five ones also merge words hierarchically. The sixth algorithm, DIT, works in a divisive manner and clusters words into a pre-specified number of word-clusters. In addition, we include three more algorithms. The first one is a hierarchical clustering algorithm that treats the realization of each word in all the training samples as a long vector and clusters them based on the similarity of these vectors. This algorithm does not need class label information and is therefore used as a baseline for comparison. Since we implement this algorithm by using Matlab's linkage() function, it is called LKG in short. The second one is a feature selection algorithm mRmR [31] (MRM) widely used in computer vision. Comparison with this algorithm is used to verify the advantage of word-mergence over word-selection in this work. The third one is probabilistic latent semantic analysis (pLSA) based on topic models. It can perform efficient dimension reduction by exploring the latent topics that generate the words. At last, since our algorithm uses the nonparametric discriminant analysis (NDA) criterion to demonstrate its efficiency, it is called GE-NDA. It will be compared with the nine algorithms mentioned above.

For the unsupervised case, our algorithm uses locality preserving projections (LPP) to demonstrate its potential, and we call it GE-LPP. It will be compared with the unsupervised mutual-information-based merging criterion (MMI) [9] and the LKG algorithm mentioned above.

Linear Support Vector Machines (SVM) and a k-Nearest-Neighbor (k-NN) classifier with a Euclidean distance are used. For the supervised case (classification), both classifiers are used to evaluate the classification performance via classification error rate. For the unsupervised case (clustering), the k-NN classifier is used to check if the underlying cluster structure is well preserved (by using the class label information at the evaluation stage only). In our algorithm, when constructing the preferred and undesired structures, histogram intersection is used to determine the neighborhood in the original high-dimensional feature space by considering its histogram property. Our algorithm has two parameters, which are the neighborhood sizes, k and k', defined in Eq.(11), (14) and (17). Their setting will be reported in the experiments for each dataset. The regularization parameter of SVM is equally tuned for each algorithm via five-fold cross-validation. The code of AIB, DIT, MRM and pLSA is obtained from the corresponding authors of [7], [16], [32], while CSM, MME, MLT, UVD and LKG are implemented by ourselves.

At last, for the kernelized version of the proposed algorithm, it will be tested for two commonly used additive kernels, Histogram intersection kernel and Hellinger kernel defined in

Table II. To rule out other effects, the simple class separability measure (CSM) is used as the word-merging criterion in this experiment. For each additive kernel, this experiment compares three settings: i) merging words with CSM and classifying samples with a linear SVM; ii) merging words with CSM and classifying with an additive kernel SVM; and iii) merging words with additive-kernel-incorporated CSM and classifying with the additive kernel SVM. It will be checked whether the last setting gives the best classification.

B. datasets, features and learning tasks

Four benchmark datasets including Caltech-256[5], PASCAL VOC2007[6], PASCAL VOC2012[7] and Scene-15[8] are used to compare the hierarchical word-merging performance.

Caltech-256 contains 256 object classes and one background clutter class. It is a significant extension of Caltech-101 by adding more object classes, increasing class sizes, reducing image artifacts and recollecting a more realistic background clutter class. To extract features, we densely sample 16×16 small local patches from each image at the step size of 8 pixels. The local patch is then characterized by the SIFT feature [20]. k-means clustering is applied to the feature descriptors to create a codebook of 1024 visual words. An image is then represented with a histogram of the number of occurrences of each word. Each histogram is ℓ_1-normalized, and square-rooting is applied to each bin to reduce noise. In the supervised case, following [8], [11], we treat each object class as the positive class and the background clutter class as the negative class to conduct a binary classification task. For each task, the goal of word mergence is to maximally maintain the classification performance while reducing feature dimensions. In the unsupervised case, the goal is to maximally maintain the cluster membership of each sample when reducing feature dimensions. The merging hierarchy in this case is learned without using class label information.

PASCAL VOC2007 and VOC2012 consist of 20 categories of objects from person, animal, vehicle and indoor objects. The same feature extraction process as in Caltech256 is applied, and a codebook of 4000 visual words is created. As previous, the histogram representation for each image is normalized and square-rooted. Following the setting of VOC Challenge, we predict the presence or absence of each object in an image. This results in 20 individual classification tasks, each of which distinguishes the images containing an object from those not containing it. Hierarchical word mergence is conducted for each classification task, respectively. The classification performance is evaluated by mean Average Precision (mAP) averaged over the 20 tasks. For VOC2007, the classifiers are trained with the union of training and validation sets and evaluated on the test set, by following the partition provided by the data set. For VOC2012, the classifiers are trained on the training set and evaluated on the validation set since the test set is not released.

[5]http://www.vision.caltech.edu/Image_Datasets/Caltech256/
[6]http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/
[7]http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/
[8]http://www-cvr.ai.uiuc.edu/ponce_grp/data/

As a traditional data set for scene recognition, Scene-15 contains 15 different scene classes. As in the literature [33], we randomly sample 100 images from each class for training and the remaining images are for test. Similar to the experimental setting and feature extraction procedure applied to Caltech-256, a codebook of 1000 visual words is created and the histograms for each image are obtained accordingly. In this experiment, each pair of the 15 scene classes is used to form a binary classification task. The word-merging algorithms are applied to all the 105 pairwise classification tasks, and their averaged performance is compared.

Considering the high computational load of the kernelized version of the proposed algorithm, a subset of Caltech-101[9] data set is used to evaluate its effectiveness. This subset contains the ten categories having the largest number of samples and is therefore called Caltech-10 in this experiment. Through the same settings on local feature extraction and codebook generation, 1000 visual words are generated. Also, ten binary classification tasks are created by treating each of the ten categories as the positive class and the "Background" category in Caltech-101 as the negative class.

C. Result of the supervised case

Caltech-256. With the initial codebook, each image is represented as a 1024-dimensional histogram. For each of the 256 classification tasks, 30 positive and 30 negative training samples are used. Through hierarchical word mergence, the dimensions of the histograms are gradually reduced from 1024 to two only. Classification is performed with these dimension-reduced histograms, respectively. As previously mentioned, the proposed GE-NDA has two parameters, the neighborhood sizes k and k'. To show their impact, we conduct GE-NDA by setting $k = k'$ with various values. The result averaged on the 256 tasks is in Figure 3, with CSM included as a reference. As seen, GDA consistently outperforms CSM when k (and k') varies in a large range from 20 to 50. In the following, we simply set $k = k' = 40$. Certainly, a more rigorous multi-fold cross-validation on k and k' could be used to produce even better classification result.

With the above setting, Figure 4(a) and (b) compare GE-NDA and all the other algorithms. The sub-figures (a) and (b) plot the classification performance obtained by a linear SVM and a 5-NN classifier. As seen from Figure 4(a), the proposed GE-NDA consistently achieves the lowest error rates once the dimensions are reduced to be lower than 700. Especially, GE-NDA is *the only one that is able to obtain the classification performance better than the case using the original* 1024*-dimensional histograms*, with the improvement as large as 5%. This is desirable because classification performance is not sacrificed while feature dimensions are reduced. Note that the error rates of GE-NDA and CSM show an increase and then decrease at the very early stage (between 1024 and 800). This is due to the following property of CSM (as previously mentioned, NDA can also be regarded as a localized version of CSM). On the Caltech-256 tasks, the CSM criterion tends

[9]http://www.vision.caltech.edu/Image_Datasets/Caltech101/

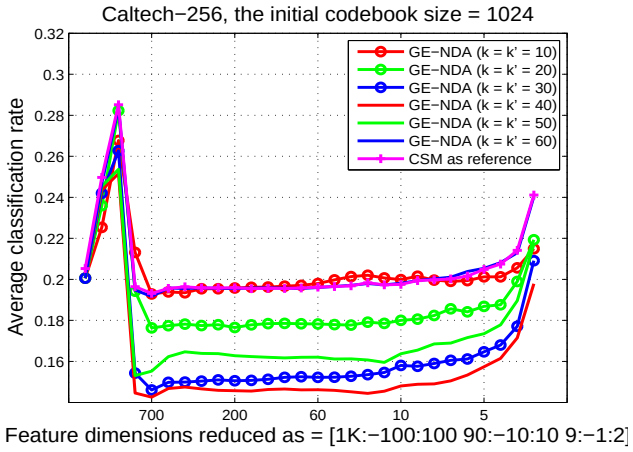


Fig. 3. The impact of k and k' to the classification error rate of GE-NDA, where k' is set as k for simplicity. k changes from 10 to 60, with the step of 10. Higher values are unnecessary because the size of training set is 60.

to first merge some dimensions that have the largest class-mean-difference values into a single dimension. This makes the value of this merged dimension increase quickly and nonproportionally dominate classification. After that, some dimensions having the smallest class-mean-difference values will be merged into a single dimension too. This restores the balance and reduces the error rate to be normal. More detailed analysis can be found from our previous work [14]. This peak happens at the very early stage and does not affect the classification at the later stage, for which hierarchical word mergence algorithms care more. For the other algorithms, the baseline LKG deteriorates with the merging process. MME, DIT and AIB show very similar performance on this task. UVD and MLT give relatively poorer results. Also, MRM degrades with the decreasing number of selected features and is outperformed by GE-NDA. This result shows the advantage of word-mergence over word-selection in this situation and is consistent with the literature [4]. Our explanation is that in the bag-of-words model, each word usually contributes its bit of discriminative power. It is the joint efforts of a sufficient number of words that achieve excellent classification. In this case, merging words could become a better option because selecting words may lead to substantial information loss when the number of selected words is small. In addition, pLSA shows reasonably good performance when the number of topics (i.e., the reduced dimensions) is set properly. However, when the number of topics is overly large or small, its performance will gradually degrade. Also, its best performance in this course is still not as good as that achieved by GE-NDA.

Figure 4(b) shows the result of the 5-NN classifier. Again, GE-NDA consistently shows the lowest error rates in most time and outperforms the second best algorithm, CSM, by 4%. This result is a good example to demonstrate the advantage of our graph-embedding-based approach. Note that for GE-NDA, it incurs a high error rate (around 0.30) at the initial. This is because the Euclidean-distance-based k-NN classifier cannot effectively measure the similarity of histogram-based image representation. With the process of merging words, the histograms are gradually embedded into Euclidean spaces. This makes the dimension-reduced histograms work with the Euclidean distance better and better. Cross-referencing the sub-figure (a), it can be found that after the word-merging process, a k-NN classifier with merely 5-dimensional features (or even lower) has been able to achieve better classification performance (0.16 vs. 0.20) than a linear SVM with the original 1024-dimensional histogram. In addition, the second-best classification performance is achieved by CSM. This well supports our analysis in Section III-B that CSM is essentially a Euclidean embedding of the original histogram representation too. GE-NDA achieves better embedding than CSM by employing the NDA criterion. Other word-merging algorithms do not work as effectively as GE-NDA and CSM in this situation, because they handle hierarchical word mergence from a perspective other than Euclidean distance based graph-embedding.

In addition, GE-NDA is further compared with the other algorithms by using a nonlinear SVM. A kernel commonly used in the BoW model, histogram intersection kernel, is employed[10]. As seen in Figure 4(c), when using a nonlinear SVM, the classification performance of all the algorithms slightly improves. UVD becomes better and even achieves the lowest error rate when the dimensions are reduced to around 400. Nevertheless, GE-NDA still demonstrates excellent overall performance, especially when the dimensions are reduced to small values.

PASCAL data sets. For the 20 classification tasks on PASCAL VOC2007, the average numbers of positive and negative training samples are 365 and 4615, respectively. For all the tasks, the neighborhood sizes k and k' are empirically set as 500 and 4500. Figure 4(d) and (e) plot the result averaged on the 20 tasks. As seen, GE-NDA shows the second-highest mean Average Precision and it is only slightly lower than MME. However, MME needs to solve an SVM-alike optimization problem at each level of the hierarchy. It is much more time-consuming and cannot handle a large number of visual words. For example, in this experiment it averagely takes MME about 9891 seconds to hierarchically merge the 4000 words, while GE-NDA only needs about 1376 seconds on a Linux platform with 2.3GHz CPU and 16GB memory. Via the proposed graph-embedding approach, we can readily incorporate the criterion like NDA to well outperform CSM and become close to MME. This again demonstrates the benefit of the proposed approach. The sub-figure (e) plots the case of k-NN classification. The mean Average Precision is obtained based on the class posteriori probability estimated from the density of the samples from different classes in the k-sized nearest neighborhood. GE-NDA, MME and CSM attain very similar performance and the performance is well maintained during the word-merging process. Among these algorithms, LKG and MLT show the worst performance. In addition, note that for MRM, only the results for dimensions lower than 1000 are plotted, because the code of MRM only selects up to 1000 features in its default setting.

[10]Let $\mathbf{x}$ and $\mathbf{y}$ denote two samples. Histogram intersection kernel is defined as $k(\mathbf{x}, \mathbf{y}) = \sum_i \min(x_i, y_i)$.

TABLE IV
AVERAGE TIME TAKEN BY THE HIERARCHICAL MERGING PROCESS (INCLUDING INDEXING AND MERGING STEPS; IN SECOND)

Strategy in merging process	Caltech256	PASCAL07 / 12	Scene-15
Fast search strategy	0.5	6.3 / 6.7	0.5
Exhaustive search	3.7	192.3 / 197.4	2.7

For the 20 classification tasks on PASCAL VOC2012, the average numbers of positive and negative training samples are 417 and 5274. The neighborhood sizes k and k' are empirically set as 1000 and 5200. As seen from the sub-figures (f) and (g), GE-NDA still shows better performance than CSM, although it is inferior to MME. However, considering the computational efficiency, GE-NDA is more promising than MME to handle tasks with a large number of visual words.

Scene-15 data set. Figure 4(h) and (i) compare the ten algorithms on the Scene-15 data set in further. For all the 105 tasks, the neighborhood sizes k and k' are empirically set as 100 and 200. As seen in both sub-figures, GE-NDA, CSM and MME achieve similar overall-best performance, and their advantage over the other algorithms becomes more and more pronounced with the process of dimension reduction. Among all the algorithms, LKG and MRM give the worst performance. pLSA can produce competitive performance when an appropriate number of topics is set. Nevertheless, its performance degrades quickly when the number of topics is decreased. The above results can be expected because i) LKG and pLSA are unsupervised methods. They focus on data representation rather than extracting discriminative information through class labels. When the dimensions are significantly reduced, they cannot effectively represent the data anymore and this causes the degradation on classification performance; ii) As aforementioned, although MRM utilizes class label information to select discriminative features, the effectiveness of feature selection will diminish with decreasing number of selected features.

At last, we demonstrate that by integrating with the fast search strategy [8], GE-NDA has higher computational efficiency than an exhaustive search. By using the two search strategies respectively, running time of the hierarchical merging process (including the merging and indexing steps at each level) is obtained for each classification task of the four data sets, and the averaged result is compared in Table IV. As seen, by using the fast search strategy, the merging process can be significantly shortened, especially on the PASCAL data sets which have higher feature dimensions. This result justifies the integration of the proposed graph-embedding approach with the fast search strategy.

D. Result of the unsupervised case

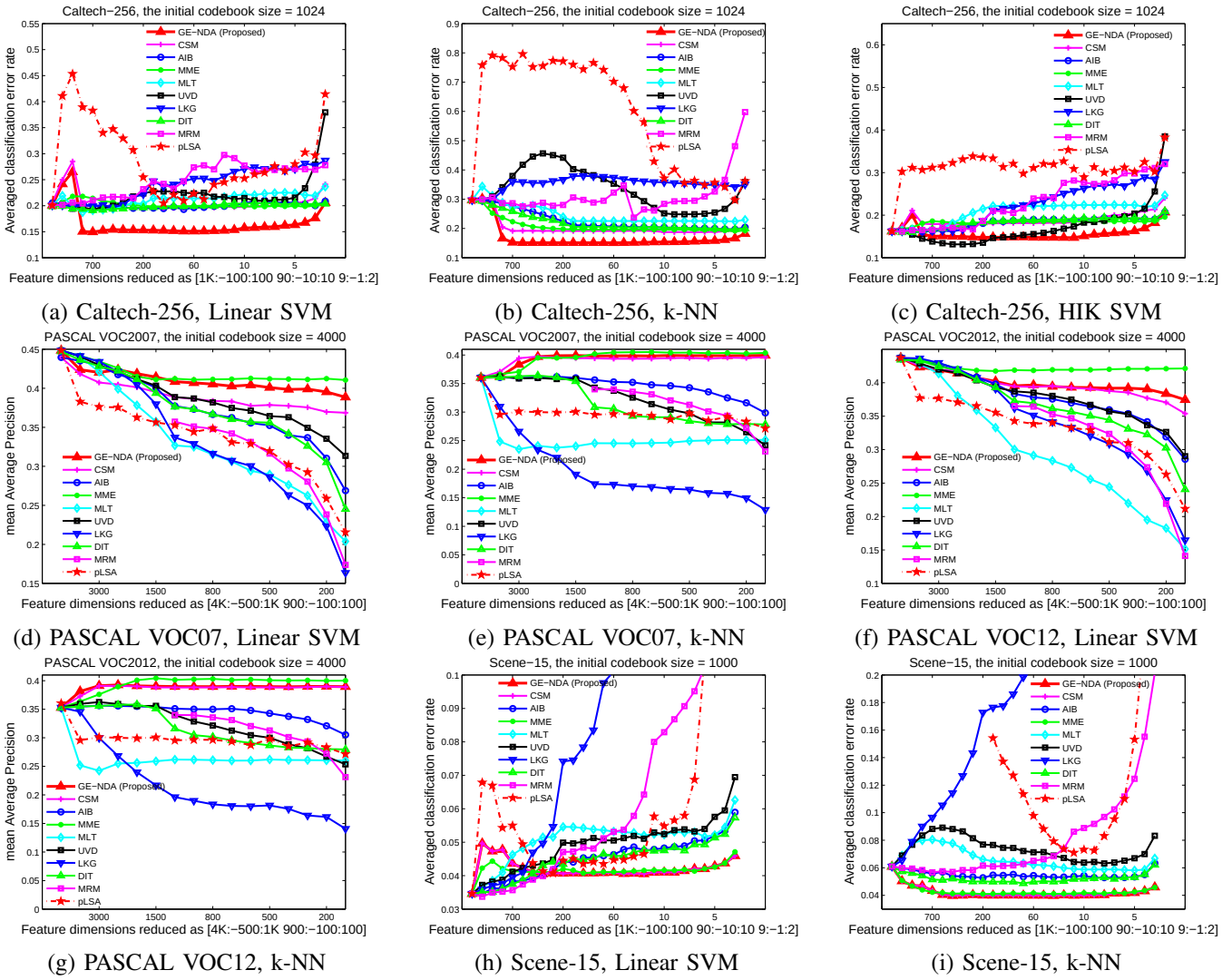
We conduct an initial investigation of the potential of the proposed graph-embedding approach for unsupervised case. As planned, our algorithm GE-LPP is compared with MMI and LKG on the 256 tasks of Caltech-256, both of which are unsupervised hierarchical word-merging algorithms. The neighborhood size k used by the LPP algorithm (defined in Eq.(17)) is empirically set as five. The average result obtained by the 5-NN classifier is plotted in Figure 5(a). As shown, our algorithm using LPP demonstrates lower classification error rates than MMI and LKG in most of the merging process and well maintains its superiority till the end. This shows that compared with LKG and MMI, our approach has higher potential to preserve the cluster membership in a lower-dimensional Euclidean space. Certainly, unsupervised cases often involve much higher dimensions and a larger number of samples. Further improving the computational efficiency of our approach to better handle that situation will be one of the central tasks in our future work.

E. Result of the additive kernel version

In this experiment we investigate the effectiveness of the additive kernel version of the proposed algorithm. To effectively demonstrate the benefit brought by additive kernels for hierarchical word mergence, we work with original histograms, that is, we do not apply the square-rooting operation as previous (which has the effect of using an additive (Hellinger) kernel) to reduce noise before merging words. Two additive kernels are tested in this experiment. One is the histogram intersection kernel (HIK in short) and the other is the Hellinger kernel (HEL in short). As previously mentioned, for each of them three settings are compared: i) merging words with CSM and classifying samples with a linear SVM; ii) merging words with CSM and classifying with an SVM using the additive kernel; and iii) merging words with additive-kernel-incorporated CSM and classifying with an SVM using the additive kernel. The difference between settings i) and ii) is whether a linear or additive-kernel SVM is applied, while the difference between ii) and iii) is whether an additive kernel is incorporated into the CSM word-merging criterion. This experiment is to check whether the classification performance will increase with moving from the first setting to the last setting. The result is shown in Figure 5(b) and (c), where the sub-figure(b) is for incorporating the additive kernel HIK while the sub-figure(c) is for incorporating the HEL. As seen, incorporating the additive kernels consistently achieves the best classification performance for all the number of clustered words. In particular, the effectiveness of incorporating additive kernels for word-merging process is well confirmed by comparing the graphs of "CSM (HIK-SVM)" (circle, in blue) and "HIK-incorporated CSM (HIK-SVM)" (square, in black) in the sub-figure(b). The better performance of the latter indicates that using the HIK additive kernel indeed improves the quality of word mergence. The same conclusion can be drawn by comparing the graphs of "CSM (HEL-SVM)" (circle, in blue) and "HEL-incorporated CSM (HEL-SVM) (square, in black)" in the sub-figure(c). These results again show the efficiency of the proposed approach.

V. CONCLUSION

This paper develops a graph-embedding-based approach to hierarchical word mergence. It can uniformly accommodate various criteria used in the well-developed graph-embedding technique and exploit them for merging words. By taking

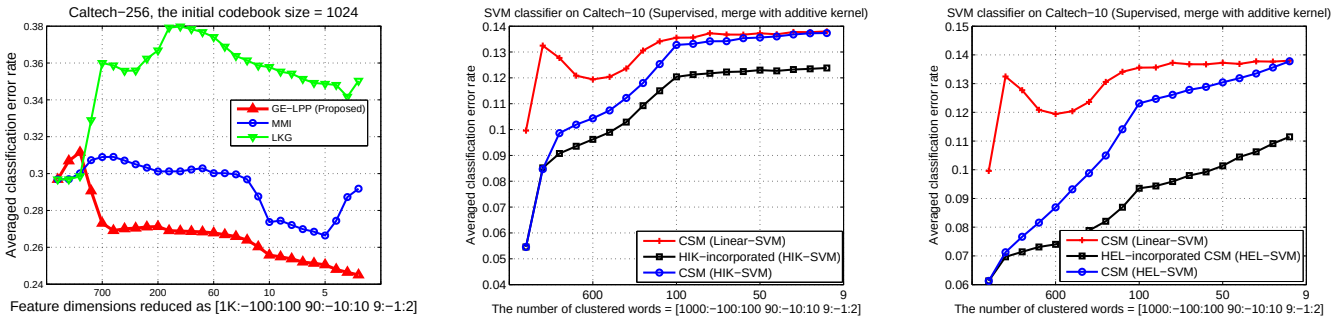


(a) Caltech-256, Linear SVM (b) Caltech-256, k-NN (c) Caltech-256, HIK SVM

(d) PASCAL VOC07, Linear SVM (e) PASCAL VOC07, k-NN (f) PASCAL VOC12, Linear SVM

(g) PASCAL VOC12, k-NN (h) Scene-15, Linear SVM (i) Scene-15, k-NN

Fig. 4. Comparison of classification performance obtained by different supervised word-merging algorithms on Caltech256, PASCAL VOC07, PASCAL VOC12 and Scene-15. The number of initial visual words for the classification tasks on these four data sets is 1024, 4000, 4000 and 1000, respectively. By hierarchical word mergence, the dimensions of image representation are gradually reduced to different lower dimensions, as shown in the horizontal axis of each sub-figure. The vertical axis shows the average classification error rate or mean Average Precision obtained by an SVM or k-NN classifier.



(a) (b) (c)

Fig. 5. (a) Comparison of the performance obtained by different unsupervised hierarchical word-merging algorithms; (b) and (c) Comparison of classification error rates obtained with or without the incorporation of an additive kernel.

advantage of a fast search strategy developed in our previous work, the proposed approach can efficiently handle thousands of visual words, making it attractive for practical applications. Experimental result demonstrates its superior performance for hierarchical word mergence. The significance of this work lies at that it provides an efficient, open and flexible platform for

applying, evaluating and developing graph-based dimension reduction criteria for the task of hierarchically merging words. A number of extensions can be developed in the future work. In particular, we will develop computationally more efficient variants of the proposed approach to deal with larger scale data sets and higher-dimensional features. Also, we will explore the potential applications of the proposed approach to other domains, for example, microarray classification [34] in which samples are represented by the expression level of different kinds of genes.

REFERENCES

[1] F. Pereira, N. Tishby, and L. Lee, "Distributional clustering of english words," in *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, Columbus, OH, Jun. 1993, pp. 183–190.

[2] N. Slonim and N. Tishby, "Agglomerative information bottleneck," in *Proceedings of the 12th Advances in Neural Information Processing Systems*, Denver, CO, Dec. 1999, pp. 617–623.

[3] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text classification," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, Melbourne, Australia, Aug. 1998, pp. 96–103.

[4] I. S. Dhillon, S. Mallela, and R. Kumar, "A divisive information-theoretic feature clustering algorithm for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1265–1287, 2003.

[5] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Proceedings of the 10th IEEE International Conference on Computer Vision*, Beijing, China, Oct. 2005, pp. 1800–1807.

[6] J. Liu and M. Shah, "Scene modeling using co-clustering," in *Proceedings of the 11th IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–7.

[7] B. Fulkerson, A. Vedaldi, and S. Soatto, "Localizing objects with smart dictionaries," in *Proceedings of the 10th European Conference on Computer Vision*, Marseille, France, Oct. 2008, pp. 179–192.

[8] L. Wang, L. Zhou, and C. Shen, "A fast algorithm for creating a compact and discriminative visual codebook," in *Proceedings of the 10th European Conference on Computer Vision*, Marseille, France, Oct. 2008, pp. 719–732.

[9] J. Liu and M. Shah, "Learning human actions via information maximization," in *Proceedings of the 21st IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, Jun. 2008, pp. 1–8.

[10] J. Liu, Y. Yang, and M. Shah, "Learning semantic visual vocabularies using diffusion distance," in *Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, Jun. 2009, pp. 461–468.

[11] L. Liu, L. Wang, and C. Shen, "A generalized probabilistic framework for compact codebook creation," in *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, Jun. 2011, pp. 1537–1544.

[12] L. Liu and L. Wang, "A scalable unsupervised feature merging approach to efficient dimensionality reduction of high-dimensional visual data," in *Proceedings of the 14th IEEE International Conference on Computer Vision*, Sydney, Australia, Dec. 2013, pp. 3008–3015.

[13] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, pp. 40–51, Jan. 2007.

[14] L. Wang, L. Zhou, C. Shen, L. Liu, and H. Liu, "A hierarchical word-merging algorithm with class separability measure," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 36, pp. 417–435, Mar. 2014.

[15] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, pp. 480–492, Mar. 2012.

[16] N. M. Elfiky, F. S. Khan, J. van de Weijer, and J. Gonzàlez, "Discriminative compact pyramids for object and scene recognition," *Pattern Recognition*, vol. 45, no. 4, pp. 1627–1636, 2012.

[17] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley Press, 2001.

[18] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, Jun. 2010, pp. 3360–3367.

[19] S. Gao, I. W. Tsang, L. Chia, and P. Zhao, "Local features are not lonely - Laplacian sparse coding for image classification," in *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, Jun. 2010, pp. 3555–3561.

[20] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision*, Kerkyra, Greece, Sep. 1999, pp. 1150–1157.

[21] F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, "Trace ratio criterion for feature selection," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, Chicago, IL, Jul. 2008, pp. 671–676.

[22] F. Chung, *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics (Book 92), American Mathematical Society, 1996.

[23] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proceedings of the 18th Advances in Neural Information Processing Systems*, British Columbia, Canada, Dec. 2005, pp. 507–514.

[24] K. Fukunaga and J. M. Mantock, "Nonparametric discriminant analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, pp. 671–678, Nov. 1983.

[25] H.-T. Chen, H.-W. Chang, and T.-L. Liu, "Local discriminant embedding and its variants," in *Proceedings of the 18th IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, Jun. 2005, pp. 846–853.

[26] X. He and P. Niyogi, "Locality preserving projections," in *Proceedings of the 16th Advances in Neural Information Processing Systems*, British Columbia, Canada, Dec. 2003, pp. 153–160.

[27] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, Dallas, TX, May 1998, pp. 604–613.

[28] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of the 19th IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, Jun. 2006, pp. 2161–2168.

[29] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Proceedings of the 18th International Conference on Machine Learning*, Williamstown, MA, Jul. 2001, pp. 577–584.

[30] Z. Lu and M. Á. Carreira-Perpiñán, "Constrained spectral clustering through affinity propagation," in *Proceedings of the 21st IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, Jun. 2008, pp. 1–8.

[31] H. Peng, F. Long, and C. H. Q. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1226–1238, Aug. 2005.

[32] J. Verbeek. (2007) Software. [Online]. Available: http://lear.inrialpes.fr/~verbeek/software.php

[33] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the 19th IEEE Conference on Computer Vision and Pattern Recognition*, New York, NY, Jun. 2006, pp. 2169–2178.

[34] M. Bicego, P. Lovato, B. Oliboni, and A. Perina, "Expression microarray classification using topic models," in *Proceedings of the 25th ACM Symposium on Applied Computing*, Sierre, Switzerland, Mar. 2010, pp. 1516–1520.