Graph-based Semi-Supervised Learning with Multi-Modality Propagation for Large-Scale Image Datasets☆

Wen-Yu Lee[a], Liang-Chi Hsieh[a], Guan-Long Wu[a], Winston Hsu[b,*]

[a]*Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 10617, Taiwan*
[b]*Graduate Institute of Networking and Multimedia and the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan*

Abstract

Semi-supervised learning (SSL) is widely-used to explore the vast amount of unlabeled data in the world. Over the decade, graph-based SSL becomes popular in automatic image annotation due to its power of learning globally based on local similarity. However, recent studies have shown that the emergence of large-scale datasets challenges the traditional methods. On the other hand, most previous works have concentrated on single-label annotation, which may not describe image contents well. To remedy the deficiencies, this paper proposes a new graph-based SSL technique with multi-label propagation, leveraging the distributed computing power of the MapReduce programming model. For high learning performance, the paper further presents both a multi-layer learning structure and a tag refinement approach, where the former unifies both visual and textual information of image data during learning, while the latter simultaneously suppresses noisy tags and emphasizes the other tags after learning. Experimental results based on a medium-scale and a large-scale image datasets show the effectiveness of the proposed methods.

Keywords: Image Retrieval, MapReduce, Semi-Supervised Learning, Large-Scale Data, Multi-Label, Image Annotation

1. Introduction

1.1. Automatic Image Annotation

With the popularization of mobile devices, uploading a photo to the Internet becomes more and more convenient. One may take a picture of anything anywhere at any moment, and then share the photo arbitrarily onto a social website. A social website, typically, allows and even suggests users to add tags to their photos. For users, tags can make their photos more informative and attractive. For the image database of a social website, adding tags may increase both the query accuracy and performance to its query related applications. However, adding tags is often considered as a hard work. For a general image database, there are relatively few images (i.e., photos) annotated with tags. As a result, automatic image annotation arises as one of the most

☆The preliminary work is presented in [1] as a 2-page poster paper.

*Corresponding author. Tel.: +886-2-3366-4888 ext. 512. Fax: +886-2-3366-4898.

Email addresses: majorrei@cmlab.csie.ntu.edu.tw (Wen-Yu Lee), viirya@gmail.com (Liang-Chi Hsieh), garywgl@csie.ntu.edu.tw (Guan-Long Wu), winston@csie.ntu.edu.tw (Winston Hsu)

popular research topics to automatically generate proper tags for all the images of an image database. Automatic image annotation is challenging mainly because of four reasons: (1) there are relatively few images having tags in a general image database, i.e., the resource is strictly limited, (2) an image database may have two or more similar images of the same landscape but with different tags from different interpretations, (3) the number of images continues to grow rapidly, and thus complicates the learning process, and (4) traditional supervised methods might not be practical because of the hard to form a model for every tag.

As a solution, semi-supervised learning (SSL) is widely-used to realize automatic image annotation. SSL is a learning technique to exploit lots of untagged images in the presence of a small amount of tagged images. Among the SSL methods, graph-based methods are quite popular due to their higher efficiencies in contrast to other methods [2]. A typical graph-based method models both tagged and untagged images as vertices followed by adding a weighted edge between each pair of vertices, where the weight of an edge is the similarity between its two terminal vertices (i.e., images). An edge weight is either positive or zero. In practical implementation, the zero-weight edges are often removed for complexity reduction. After that, a process called label propagation is activated to add tags to images (i.e., to label images) accordingly. Traditional label propagation implies *single-label* propagation, where each image considers only a single tag to reduce the complexity on label propagation. Recently, the study of multi-tag annotation arises [3]. It is desirable to consider multiple tags during label propagation. Although we may perform *single-label* propagation several times for multi-tag annotation, it might not be practical for a large-scale image dataset. In this paper, we focus on the development of a *multi-label* propagation approach for graph-based SSL for automatic image annotation. Note that in this paper the term, "label," is equivalent to "add tag(s)" or "tag," depending that it is a verb or a noun, respectively.

1.2. Previous Work

Previous works on automatic image annotation using graph-based SSL can be classified into two major technical components: (1) graph construction [4, 5, 6, 7, 8, 9], and (2) label propagation [10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

For graph construction, k-nearest neighbor (k-NN)-based and ε-neighborhood-based methods are commonly used [7], where the former adds edges between a vertex and the k-nearest neighbor(s) of the vertex, while the latter constructs an edge for two vertices if the distance (e.g., similarity) between them is within a pre-defined distance, i.e. ε. In contrast, Jebara *et al.* argued the importance to form a regular graph followed by presenting a regular-graph construction approach based on b-matching [5]. However, the use of b-matching may result in a high complexity which may not be practical to large-scale datasets. To remedy the deficiency, Elsayed *et al.* in [4] proposed to compute pairwise document similarity by MapReduce [20], which is a programming model famous in large-scale distributed computing. Then Lin investaged three algorithms for pairwise similarity comparisons with MapReduce, and showed empirically that the brute force algorithm is the most efficient when exact similarity is desired [6]. Recently, Liu *et al.* proposed to construct a scalable graph by coupling anchor-based label prediction and adjacency matrix design [8]. Luo *et al.* annotated multispectral images based on the author-topic model [9].

For label propagation, Zhu *et al.* proposed to use Gaussian random field model [10]. Later, Zhou *et al.* presented an iterative algorithm followed by deriving a simple while effective closed-form solution [11]. After that, He *et al.* improved the graph structure in [11] and indicated that the closed form solution in [11] may not be applicable for large-scale label propagation [12].

Moreover, He *et al.* combined relevance feedback with label propagation. As a counterpart of [12], which focused on the scenario of query by example, Tong *et al.* focused on the scenario of query by keyword and proposed the idea of multi-label propagation [14]. Unlike the prior works, which are deterministic methods, Pan *et al.* proposed a probabilistic solution based on random walk with restart [13]. On the other hand, Tong *et al.* presented a linear fusion and a sequential fusion schemes to fuse multi-modal features for learning quality enhancement [15]. In addition, Wang *et al.* argued to integrate *visual* and *textual* information for label propagation [16]. Liu *et al.* presented to integrate similarities immediately on different graphs for label propagation [17]. Recently, Liu *et al.* observed that most label-propagation approaches have to perform at least an inverse operation on a n-by-n graph Laplacian, where n is the number of vertices. The inverse operation often requires a cubic time (i.e., $O(n^3)$) complexity, which becomes prohibitive as the number of vertices (i.e., images) increases [8]. For the addressed problem, Rao and Yarowsky focused on text data to develop a parallel label-propagation algorithm [18]; unfortunately, such a text-oriented algorithm may not handle more complicated data (e.g., images) well. More recently, Kang *et al.* presented a matrix-vector multiplication approach by MapReduce [19].

1.3. Our Contributions

This paper focuses on label propagation for graph-based SSL, where two important issues raised in recently years. One is the large-scale issue: the proposed algorithm has to be useful to high-dimensional data and large-scale datasets. The other is the multi-labeling issue: the proposed algorithm has to handle several tags on an image, so as to explore (improve) multiple tags for each untagged (tagged) image, while most previous works focused on single tag issue. Consequently, we unify the promising ideas of (a) graph-based SSL [11, 12, 13], (b) multi-label propagation [14], (c) linear fusion [15, 17], and (d) matrix-vector multiplication [19], to develop a multi-layer multi-label propagation framework for large-scale image datasets. Note that for multi-label propagation, we generalize the matrix-vector multiplication in [19] to a matrix-matrix multiplication, i.e., matrix multiplication.

Overall, in this paper, (1) we implement a flexible multi-layer learning structure, which unifies the visual and textual features of given images by linear fusion to enhance the learning performance; (2) we consider multi-label propagation to enable the addition and refinement of multiple tags to untagged and tagged images, respectively; (3) we introduce a technique to set the convergence criterion of our multi-label propagation approach automatically instead of using an ad-hoc assignment; (4) we present a practical implementation of our multi-layer multi-label propagation approach by MapReduce, which is capable of handling large-scale image datasets; (5) we conduct experiments on a medium-scale and a large-scale image datasets to evaluate the effectiveness of the proposed methods. The rest of this paper is organized as follows. Section 2 presents the overview of our graph-based SSL system. Section 3 introduces the proposed algorithms for label propagation. Section 4 presents the experimental evaluation. Finally, Section 5 concludes this paper.

2. Learning System Overview

Before turning to the details of our label-propagation algorithm, this section gives an overview of our graph-based SSL system. The overall system flow is illustrated in Figure 1. Given a set of tagged and untagged images, for each image we extract both the visual and textual features. Note

that the textual features might be missing. With the extracted features, we then construct a sparse visual graph and a sparse textual graph accordingly. Each vertex of the visual (textual) graph is associated with an image, while the weight of an edge between two images is the visual (textual) similarity of the two image contents. Subsequently, our label-propagation algorithm is activated to add (improve) tags to the given untagged (tagged) images. Finally, we suppress noisy tags by tag refinement and output the results.

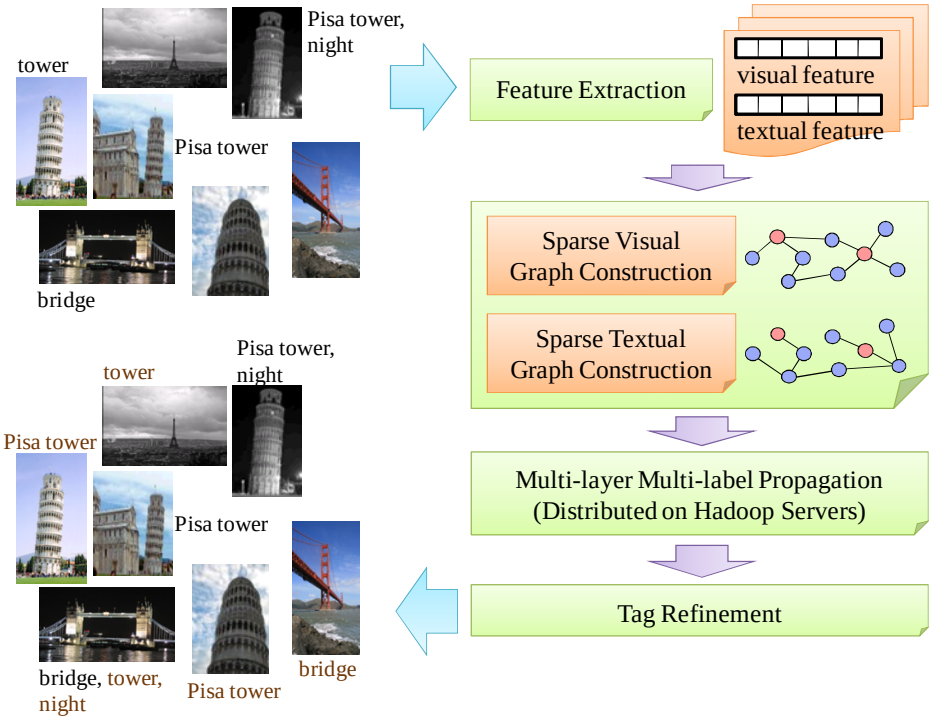

Figure 1: An illustration of our graph-based SSL system. Given an image dataset, we first perform feature extraction to obtain both visual and textual features. Then a sparse visual graph and a sparse textual graph are constructed to enable multi-layer label propagation. Based on the two graphs, we then perform multi-layer multi-label propagation. Finally, we apply tag refinement to improve the tag results.

As aforementioned, we construct two sparse graphs before the activation of label propagation. In fact, our label-propagation algorithm can also work on a dense graph, e.g., a complete graph. We do not work on a dense graph but on a sparse graph because it has been shown in [21] that fully-connected dense graphs performed worse than sparse graphs empirically. A dense graph may contain many spurious edges between dissimilar vertices, and therefore misleads label propagation to a low-quality result. Since graph construction is not the main focus of this paper, we shall not give the details here, but a brief description instead, as follows. Overall, our in-house tool for graph construction consists of two stages. For large-scale issues, both the two stages were implemented based on the MapReduce programming model [20]. In the first stage, we partition the given images into overlapped groups called image pools by MinHash [22], while in the second stage, for each image pool we compute the pairwise similarities and remove image-pool boundaries afterwards. Note that we do not build any zero-weight edge to reduce the graph

complexity. The resultant graph is our sparse similarity graph. For simplicity, in the rest of the paper, we will use the terms "visual graph" and "textual graph" to be as shorthand for "sparse visual graph" and "sparse textual graph," respectively.

3. Methodologies

This subsection presents our proposed methods for multi-layer multi-label propagation. Note that all of the distributed algorithms mentioned in the rest of paper are developed based on the MapReduce model. Section 3.1 introduces our multi-layer learning structure followed by the overview of our proposed algorithm. Section 3.2 gives a formal description of our multi-layer multi-label propagation approach. Section 3.3 presents the convergence criterion setting of our label propagation approach. Finally, Section 3.4 describes the tag refinement approach.

3.1. Multi-Layer Learning Structure

Given a visual and a textual graph, we construct a multi-layer learning structure, which is sketched in Figure 2. In this structure, the top layer is a visual graph, the bottom layer is a textual graph, and the middle layer shows an abstract fusion layer for the visual and textual graphs to communicate with each other during the learning process. The reasonable behind is that empirically there is often a high correlation between the similarity of two images in the visual graph and that of the two images in the textual graph. With the fusion layer, the two graphs can supervise and guide each other to achieve a high-quality learning result.

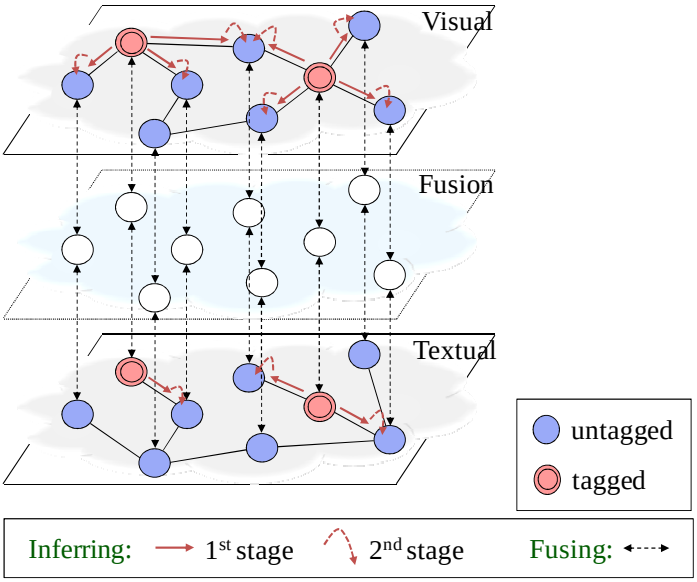

Figure 2: Multi-layer learning structure. The top layer is a visual graph, the bottom layer is a textual graph, and the middle layer shows an abstract fusion layer for the communication of the visual and textual graphs. Note that initial labels are shown in double circles (light red). Graph structure can also benefit from the MapReduce model by only propagating and receiving information from the neighbors.

Our multi-layer multi-label propagation algorithm is an iterative process. We assign some initial tags as ground truth in our algorithm. Each iteration consists of two inferring processes

and one fusion process (linear fusion, see, e.g., [15]), where an inferring process is a two-stage process, which is composed of two separate distributed algorithms; the fusion process is a distributed algorithm, too. Note that the actions of the two inferring processes are the same but applied on different graphs: one is on the visual graph, and the other is on the textual graph. An inferring process is activated to propagate the label information (e.g., a value) from each labeled vertex to its adjacent vertices. More precisely, the first stage of an inferring process weights the adjacent edges of each vertex using the label information, while the second stage collects the weighted value(s) to the vertices pointed to (*cf.* Figure 2). After the inferring processes, the fusion process is activated. The fusion process fuses every two vertices in the different graphs but referring to the same image, in a weighted fashion with a user-specified parameter $\beta \in (0,1)$ (*cf.* Eq. (2), e.g., $\beta = 0.5$ for averaging) so that the two graphs can communicate with each other. Finally, both visual and textual graphs are updated by assigning each fused value back to the two corresponding vertices. The overall learning process is repeated until a certain number of iterations have been reached.

3.2. Multi-Layer Multi-label Propagation Algorithm

This subsection details our multi-layer multi-label propagation algorithm. Our algorithm is extended from [11], targeting at multi-label propagation for large-scale image datasets. For simplicity, we shall concentrate on the operations in the visual graph, and shall not mention those in the textual graph, except when we explain the fusion process. Note that each vertex may have tags or no tag at all.

In the sequel, let G be the visual graph with vertex set $\mathscr{X} = \{x_1, ..., x_n\}$, where n is the vertex number; let $W \in \mathbb{R}^{n \times n}$ be the similarity matrix of G, where W_{ij} denotes the edge weight between vertices x_i and x_j; let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix with D_{ii} equaling the inverse square root of the sum of the ith-row of W; let $\mathscr{L} = \{1, ..., c\}$ be the label set of G; let $Y \in \mathbb{R}^{n \times c}$ be a matrix with $Y_{ij} = 1$ if x_i is initially labeled as j, and $Y_{ij} = 0$ otherwise; let $F \in \mathbb{R}^{n \times c}$ be a classification matrix, where eventually each vertex x_i will be assigned tag(s) by tag refinement (see Section 3.4) based on the i-row of F; let $\alpha \in (0,1)$ be a user-specified parameter. Without loss of generality, assuming that initially the first l (typically, $l \ll n$) vertices $\{x_1, ..., x_l\}$ are individually labeled, and the other $(n-l)$ vertices $\{x_{(l+1)}, ..., x_n\}$ are unlabeled. Our objective is to infer the labels of the unlabeled vertices. Note that in general both S and Y are sparse.

The propagation process is as follows. Initially, we assign Y to F. Then we create a normalized weight matrix S, which equals the product of D times W times D (i.e., $S = D \times W \times D$). Next, we activate an iterative process, iteratively performing the following two operations.

$$F^{((i+1),vis)} = \alpha S \times F^{(i,vis)} + (1-\alpha)Y, \tag{1}$$

$$F^{((i+1),vis)} = \beta F^{((i+1),vis)} + (1-\beta)F^{((i+1),txt)}, \tag{2}$$

where the meaning of a matrix with the superscript "(i, vis)" is twofold: obtained in the i-th iteration, and used for learning on the *visual* graph (i.e., $F^{(0,vis)} = F$). Similarly, $F^{((i+1),txt)}$ is the classification matrix for the *textual* graph obtained in the $(i+1)$-th iteration. Note that β is the fused parameter (*cf.* Section 3.1). As aforementioned, the physical meanings of the two are to infer labels and then to fuse with the textual graph. Note that fusion (Eq. (2)) in the i-th iteration can only be done after inferring labels (Eq. (1)) in both of the visual graph and textual graphs in the i-th iteration. The iterative process is repeated until a certain number of iterations is reached. We will show how to decide the iteration number in Section 3.3.

Now we present our implementation for Eq. (1), i.e., the inferring process. Note that both S and Y are invariant during the whole propagation process. We substitute S' for αS and Y' for $(1-\alpha)Y$, i.e., $S' = \alpha S$ and $Y' = (1-\alpha)Y$. Clearly, S' (Y') is sparse if S (Y) is sparse. Therefore, we rewrite Eq. (1) as

$$F^{((t+1),vis)} = S' \times F^{(t,vis)} + Y', \tag{3}$$

which contains a matrix multiplication and a matrix addition. As mentioned earlier, we implemented such an inferring process by a two-stage distributed process: the first stage performs the multiplication part of the matrix multiplication, while the second stage performs the addition concurrently for the products from the first stage and for the matrix addition. The two stages are described in Figure 3 and Figure 4, sequentially. Note that we store each matrix based on a coordinate list representation: each line is associated to a row-column index pair and a nonzero weight.

The rationale behind Figure 3 is that given an operation, S' times F, each entry of the i-th column of S' should and would be multiplied by each entry of the i-th row of F exactly once. This can be done by assigning proper key values in the *map* procedure followed by performing pairwise multiplications in the *reduce* procedure. Subsequently, in Figure 4 we add the resultant entries from Figure 3 and the entries of Y' together according to their row and column indexes. (Two entries can only be added if they have the same row and column indexes.) Note that in practical implementation, the *reduce* procedure can also work as a combiner of the MapReduce programming model. In addition, it is instructive to note that if in Figure 4 we consider only the resultant entries from Figure 3, the two-stage process will become a generalized version of the matrix-vector multiplication approach [19] for matrix-matrix multiplication. So far we have presented the implementation of the inferring process. In particular, the fusion process can also be achieved in a similar way as in Figure 3, where the fusion parameter β is injected in the *map* procedure.

3.3. Convergence Criterion of Label Propagation

The issue remains in label propagation is to decide the iteration number as the criterion of convergence. This is important for a distributed environment because it is hard to inspect each subgraph individually. We will also show experimentally in Section 4.5 the necessity to determine the iteration number automatically and globally. Intuitively, this number must be large enough such that each image can propagate its label information to those images related to it. That is, a vertex u should propagate its label information to every vertex v during label propagation if there is a path from u to v. To this end, we first replace each edge weight by one. Note that our graph did not reserve the zero-weight edges before; the edges considered here are all positive-weight edges. By doing so, the original problem can then be transformed into the finding of maximum path length from the shortest paths between all pairs of vertices in the graph, where each missing edge represents an infinite distance. Since the inferring process introduced before in nature realizes a matrix multiplication, we thus find the all-pairs shortest paths using dynamic programming, i.e., the matrix multiplication-based approach [23]. Once the all-pairs shortest paths have been identified, we can find the maximum path length accordingly. Note that the processes described above can be implemented following the ideas of Figure 3 and Figure 4 with small modifications.

```
1:  Procedure Map(String key, String value)
2:  Input:
    key: file name of S', file name of F;
    value: a matrix entry;
3:  Begin
4:      (rowId, colId, weight) = ParseLine(value);
5:      if IsSbarFile(key) is true
6:          newKey = colId;
7:          newValue.setTriple('S', rowId, weight);
8:      else
9:          newKey = rowId;
10:         newValue.setTriple('F', colId, weight);
11:     end if
12:     Emit(newKey, newValue);
13: End

14: Procedure Reduce(String key, Iterator values)
15: Begin
16:     ArrayList arrayListS, arrayListF;
17:     foreach val in values
18:         if val.first is 'S'
19:             arrayListS.addPair(val.second, val.third);
20:         else
21:             arrayListF.addPair(val.second, val.third);
22:         end if
23:     end foreach
24:     foreach dataS in arrayListS
25:         (rowId, weightS) = dataS.getPairData();
26:         foreach dataF in arrayListF
27:             (colId, weightF) = dataF.getPairData();
28:             Emit(rowId+" "+colId, weightS×weightF);
29:         end foreach
30:     end foreach
31: End
```

Figure 3: The first stage of an inferring process with MapReduce.

3.4. Tag Refinement

In this subsection, we present the tag refinement approach used in this paper, which is applied after label propagation. Note that at this moment, the classification matrices in both the visual graph and the textual graph are the same. We take any of them as the resultant classification matrix. In the sequel, we define the transpose matrix of the ith-row matrix of the resultant classification matrix as the tag matrix $T_i \in \mathbb{R}^{c\times 1}$, where $(T_i)_{j1}$ is the score of image i to tag j of $\mathscr{L}$ for $1 \leq j \leq c$. Before proceeding, let us explain the reason for using tag refinement. We need tag refinement to improve image tags learned because the original images crawled from an image

```
1:  Procedure Map(String key, String value)
2:  Input:
    key: file names;
    value: a matrix entry;
3:  Begin
4:      (rowId, colId, weight) = ParseLine(value);
5:      Emit(key, weight);
6:  End

7:  Procedure Reduce(String key, Iterator values)
8:  Begin
9:      sum=0.0;
10:     foreach val in values
11:         sum += val;
12:     end foreach
13:     Emit(key, sum);
14: End
```

Figure 4: The second stage of an inferring process with MapReduce.

dataset, such as Flicker, without any pre-processing may have some noisy tags. In other words, some tags may not properly (or even not correctly) interpret the images. Undoubtedly, such noisy tags may introduce noises into the learning results after label propagation. As a solution, the main objective of tag refinement is to effectively suppress the noisy tags while emphasize the others. That is, given an image with a tag list, we would like to to reorder the list such that tag i precedes (i.e., gets a higher rank) tag j if tag i is more relevant to the image than tag j. Figure 5 gives two examples. Each image contains two list of tags with different orders, where the upper (lower) list is the tag list before (after) tag refinement. Consider the left image. We can see that after tag refinement, the tag list is sorted according to their relevance to the image. For example, tag "Tower of Pisa" is placed at the first in the tag list. This means that "Tower of Pisa" is the most-relevant tag to the image here. The right image is the same. The most-relevant tag, "Golden Gate Locale," in the tag list is placed at the first in the tag list after tag refinement.

Recently, a promising work for tag refinement was presented in [24], where Zhu *et al.* proposed to decompose a user-provided tag matrix into a low-rank refined matrix and a sparse error matrix. The work in [24] is effective for a global tag refinement; however, it may not be appropriate to our problem here as each entry in our tag matrices represents a score and our objective is to tune the scores locally (a global tag refinement may obscure the experimental results of our proposed approaches). As a result, we modified the approach of learning class-specific weights for the searching of visually similar images in [25] to fit our needs. Without loss of generality, assume there is a union set of vertices $\{x_1, x_2, ..., x_{(k-1)}\}$ which are adjacent to a target vertex x_k in the visual or textual graphs. Let $\texttt{adjVis}(x_k)$ and $\texttt{adjTxt}(x_k)$ be the vertices adjacent to x_k in the visual and textual graph, respectively, i.e., $\texttt{adjVis}(x_k) \bigcup \texttt{adjTxt}(x_k) = \{x_1, x_2, ..., x_{(k-1)}\}$. Let $A_1, A_2, ..., A_k$ be the weight vectors (i.e., column matrices), where $A_i \in \mathbb{R}^{c\times 1}, 1 \leq i \leq k$. We consider the proximity between x_i and x_j by the similarity between them, for $1 \leq i, j \leq k$ and $i \neq j$, to preserve the inter-image relationship. Finally, our formulation is as follows (*cf.* Eqs. (2)–(5)

Tags (before):
Italy, tower, Tower of Pisa, Pisa, Roman, Pisa tower,

Tags (after):
Tower of Pisa, Pisa tower, Pisa, tower, Roman, Italy

Tags (before):
California and Marin County, traveling, tourists, Golden Gate Locale, SFO, bridge, California, San Francisco, vacation

Tags (after):
Golden Gate Locale, San Francisco, SFO, bridge, California and Marin County, California, traveling, tourists, vacation

Figure 5: An illustration of tag refinement. After tag refinement, the tag order of a tag list is changed while the number of tags is the same. A tag in the tag list of an image precedes if it is more relevant to the image than the tag(s) placed behind.

of [25]).

$$\begin{aligned}\text{Minimize} \quad & \gamma \sum_{x_i \in \texttt{adjVis}(x_k)} \left(W_{ik}^{vis} \|A_i \circ T_i - A_k \circ T_k\|^2\right) + \\ & (1-\gamma) \sum_{x_i \in \texttt{adjTxt}(x_k)} \left(W_{ik}^{txt} \|A_i \circ T_i - A_k \circ T_k\|^2\right)\end{aligned}$$

$$\text{subject to} \quad A_i^\top \times \mathbf{1} = 1, i = 1, ..., k, \tag{4}$$

$$A_i \geq \mathbf{0}, i = 1, ..., k, \tag{5}$$

where $\gamma \in (0,1)$ is a parameter that controls the balance of the visual and textual graphs, W_{ik}^{vis} (W_{ik}^{txt}) is the edge weight between x_i and x_k in the visual (textual) graph, the symbol "$\circ$" denotes element-wise (Hadamard) product, the superscript "$\top$" denotes the transpose operation, and $\mathbf{1}$ ($\mathbf{0}$) denotes a $c \times 1$ matrix filled with ones (zeros). For the objective function to one graph, the term $\|.\|^2$ shows a weighted distance between two images, which is further weighted by the similarity between the two corresponded images. For the constraints, Constraint (4) enforces the summation of the entries of each weight vector to be one, while Constraint (5) enforces every entry of a weight vector to be nonnegative. As a result, we can learn k weight vectors, in which $A^{(k)}$ is used to concurrently suppress the noisy tags and emphasize the other tags of our target image, x_k, by weighting $T^{(k)}$ in element-wise. Note that if a normalized result is preferable, we may rewrite Constraint (4) as follows:

$$A_i^\top \times T_i = 1, i = 1, ..., k. \tag{6}$$

4. Experiments

This section evaluates the effectiveness of the proposed multi-label graph-based SSL approach and the necessity of tag refinement. We implemented our approaches using the Java

programming language. The experiments were conducted on a middle Hadoop cluster (version 0.20.1) consisting of 24 commodity machines. The rest of this section is organized as follows. Section 4.1 presents the input datasets and explains the basic experiment setup. Section 4.2 introduces the evaluation criteria. Section 4.3 compares our multi-layer multi-label propagation with four traditional approaches. Section 4.4 investigates the effectiveness of tag refinement. Section 4.5 evaluates the proposed convergence criterion for label propagation. Finally, Section 4.6 performs the sensitivity tests for the three parameters used in this paper.

4.1. Experiment Setup

All of the experiments were based on the following two image datasets.

Flickr550K: Flickr550K contains $540,321$ images with $9,360$ manually-annotated ground truth images in 21 query categories [26]. (Note that in [26] this dataset was named Flickr550. We use Flickr550K instead to be consistent with Flickr11K introduced in the following.)

Flickr11K: Flickr11K is made as a specific subset of Flickr550K [27]. It contains $11,277$ medium resolution (500×360) images with $1,282$ ground truth images in seven query categories.

For both Flickr11K and Flickr550K, each image is presented in visual and textual high-dimensional features. For each dataset, we consider the $1,282$ ground truth images in the seven query categories, which are colosseum, eiffel tower, golden, starbucks, torre pendente di pisa, tower bridge, and triomphe. Note that although Flickr550K is with 21 query categories, we focus on the seven out of the 21 query categories.

We extracted the visual and textual features in the following ways. We took *visual word* as visual features for similarity computation. For visual word generation, we adopted the difference-of-Gaussian (DoG) approach to detect feature points followed by describing them with scale invariant feature transform (SIFT). The SIFT descriptors were then quantized into 10k clusters using k-means clustering, where each cluster defined a visual word containing the feature descriptors (feature points) in this cluster. For the textural features, we adopted the expanded Google snippet from the Google search engine to perform query expansion to represent associated (noisy) tags in textual features in $91,004$ dimensions.

To be close to the real world image data (i.e., relatively few tagged images), for all of the textual graphs used, we reserved only a portion of tagged images by randomly sampling, while removed the tags from the other tagged images. For graph construction, we reserved only 30% tagged images by randomly sampling from each dataset to fit the real world condition. For label propagation, the focus of this paper, we further reduced the number to 100 to show the effectiveness of our propagation method. That is, Flickr11K and Flickr550K reserved only 0.89% and 0.02% tagged images (randomly sampled), respectively.

Since general graph-based SSL is sensitive to the initial labels, for each experiment we repeated the learning process ten times and average the results for a more accurate test result.

4.2. Evaluation Criteria

We evaluate the performance by *Mean Average Precision (MAP)*, which is the mean of the average precision of the tags for each image. That is, a higher MAP indicates a better retrieval results. For tag refinement, we focus on the precision of the top-n labels of the ranked label set of each image, i.e., P@n. Empirically, parameters α, β, and γ were set to 0.9, 0.5, and 0.6, respectively. Note that the sensitivity test of each of them can be found in Section 4.6.

Table 1: Compare the visual graph, textual graph, early-fusion [16], late-fusion [16], and multi-layer (ours) -based learning methods in MAP, where the percentages (%'s) are the improvement ratios between the visual graph only and the multi-layer learning methods. We can see that the multi-label method can achieve better results than the others.

	Visual Only	Textual Only	Early Fusion	Late Fusion	Multi-layer (Ours)
Flickr11K	0.2859	0.3083	0.3114	0.3418	**0.3611 (26%)**
Flickr550K	0.1632	0.1994	0.2149	0.2342	**0.2883 (77%)**

Table 2: Compare the use of tag refinement and not in MAP, where the resultant top-1, 2, 3, 4, 5, and 10 labels are considered.

Flick11K	P@1	P@2	P@3	P@4	P@5	P@10
Multi-label w/o Refinement	0.0016	0.0059	0.1059	0.1809	0.3029	0.3548
Multi-label w/ Refinement	0.0842	0.1033	0.1348	0.2948	0.3541	0.3611
Flick550K	P@1	P@2	P@3	P@4	P@5	P@10
Multi-label w/o Refinement	0.0003	0.0189	0.1093	0.1777	0.2574	0.2680
Multi-label w/ Refinement	0.0546	0.1098	0.1357	0.1867	0.2883	0.3271

4.3. Multi-Layer Multi-Label Propagation

This subsection compares our multi-layer learning with visual graph, textual graph, early fusion and late fusion -based learning, where visual (textual) graph-based learning uses no textual (visual) graph, and early (late) fusion-based learning considers both visual and textual graph but fuses only at the beginning (end). All of the label propagations here were implemented based on our multi-label propagation approach described in Section 3.2 to tackle the large-scale dataset, i.e., Flickr550K. The results are shown in Table 1. As revealed in the table, our multi-layer learning can significantly improve the baselines by aggregating visual and textual contexts during message passing among the graphs. Moreover, the results also verify that our multi-label propagation approach is practical and useful to a large-scale image dataset.

After multi-label propagation, each image contains a set of tags, where each tag is with a score. With this information, we then do tag refinement to suppress noisy tags.

4.4. Tag Refinement

As mentioned in Section 4.2, here we investigate the precision of the top-n labels of the ranked label set of each image, i.e., P@n. Table 2 shows the experimental results. We consider the top-1, 2, 3, 4, 5, and 10 labels. As can be seen, the results of using tag refinement are consistently better than using no tag refinement. Based on the results, we further believe that the noisy tags indeed may lower the learning quality a lot. With tag refinement, such noises are effectively suppressed by considering the neighbor information of an image graph.

4.5. Convergence Criterion - Shortest Path

As described in Section 3.2, we set a maximum number of iterations as the criterion of convergence for label propagation. Such a number is defined as the maximum path length from the shortest paths between all pairs of vertices in the graph such that each image can propagate its label information to any images related to it. At this moment, we compare the shortest path-based approach and an ad-hoc approach which arbitrarily sets the maximum number of iterations. The experimental results are shown in Table 3. Each of the numbers, $1, 5, 10, 50$, and 100, denotes the maximum number of iterations adopted. Note that the maximum numbers of iterations to Flickr11K and Flickr550K are 13 and 15, respectively. As we can see, our shortest path-based approach can achieve the best results. From the results, it is interesting to notice that using more than enough iteration may degrade the performance. This may be because of the magnification of the influence of the potential noisy tags.

Table 3: Compare the shortest path-based approach and an ad-hoc approach in MAP. Based on the results, we can see that the shortest path-based results are comparable to those by the ad-hoc approach.

	Ad-Hoc Approach					Shortest Path-Based Approach
	1	5	10	50	100	
Flickr11K	0.3297	0.3330	0.3504	0.3473	0.3455	**0.3611**
Flickr550K	0.2544	0.2629	0.2680	0.2643	0.2697	**0.2883**

4.6. Sensitivity Test

In this subsection, we conduct experiments to decide the three user-specified parameters used in this work, i.e., α, β, and γ. They are for the inferring process, the fusion process of label propagation, and the balance control of tag refinement, respectively. The sensitivity tests of them are as follows. Table 4 investigates the use of different α in the inferring process. As can be seen, when α equals to 0.9, we can achieve the best MAP. Thus we set α to 0.9. Table 5 investigates the use of different β in the fusion process. As can be seen, the best setting of β is 0.5. This result also reflects the necessity of fusing the visual and textual graphs to achieve a better MAP. They are both important. No matter we bias toward any of them, the resultant MAP would get worse. Table 6 investigates the use of different γ in balance control. As revealed in the table, the best setting of γ is 0.6. Consequently, parameters α, β, and γ were set to 0.9, 0.5, and 0.6, respectively.

Table 4: Consider the setting of α (*cf.* Eq. (1)), i.e., the inferring parameter in the graph-based multi-layer multi-label propagation in MAP.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	**0.9**
Flickr11K	0.0097	0.1044	0.1504	0.1695	0.1766	0.2112	0.2373	0.2501	**0.2554**
Flickr550K	0.0999	0.1203	0.1400	0.1504	0.1655	0.1799	0.1867	0.2003	**0.2237**

Table 5: Consider the setting of β (*cf.* Eq. (2)), i.e., the fusion parameter in the graph-based multi-layer multi-label propagation in MAP.

	0.1	0.2	0.3	0.4	**0.5**	0.6	0.7	0.8	0.9
Flickr11K	0.2113	0.2367	0.2548	0.2481	**0.2677**	0.2499	0.2410	0.2158	0.2035
Flickr550K	0.2044	0.2130	0.2205	0.2311	**0.2359**	0.2245	0.2139	0.2008	0.1917

Table 6: Consider the setting of γ (*cf.* Eq. (4)), i.e., the balance control parameter in tag refinement in MAP.

	0.1	0.2	0.3	0.4	0.5	**0.6**	0.7	0.8	0.9
Flickr11K	0.2572	0.2628	0.2848	0.3066	0.3244	**0.3620**	0.3502	0.3115	0.2753
Flickr550K	0.1977	0.2046	0.2283	0.2380	0.2508	**0.2737**	0.2494	0.2229	0.2002

5. Conclusion

We have presented a graph-based multi-layer multi-label SSL method which can effectively unify the visual and textual information for multi-label learning. Our framework does not require to pre-processing the given large-scale dataset but is capable of performing large-scale multi-label propagation. On the other hand, we have also presented a tag refinement technique which can simultaneously suppress noisy tags and emphasize the other tags. Experimental results have shown that our algorithm can operate on a large-scale image dataset while effectively infer the image labels. Future work includes the development of learning on large-scale video datasets.

References

[1] W.-Y. Lee, L.-C. Hsieh, G.-L. Wu, W. Hsu, Y.-F. Su, Multi-layer graph-based semi-supervised learning for large-scale image datasets using mapreduce, in: Proc. of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2011, pp. 1121–1122.

[2] X. Zhu, Semi-supervised learning literature survey, Doctoral Dissertation of Carnegie Mellon University (2006).

[3] F. Wu, Y. Han, Q. Tian, Y. Zhuang, Multi-label boosting for image annotation by structural grouping sparsity, in: Proc. of the International Conference on Multimedia, 2010, pp. 15–24.

[4] T. Elsayed, J. Lin, D. W. Oard, Pairwise document similarity in large collection with mapreduce, in: Proc. of the Association for Computational Linguistics on Human Language Technologies, 2008, pp. 265–268.

[5] T. Jebara, J. Wang, S.-F. Chang, Graph construction and *b*-matching for semi-supervised learning, in: Proc. of the International Conference on Machine Learning, 2009, pp. 441–448.

[6] J. Lin, Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce, in: Proc. of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, pp. 155–162.

[7] P. P. Talukdar, Topics in graph construction for semi-supervised learning, Technical Report of University of Pennsylvania (2009).

[8] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, in: Proc. of the International Conference on Machine Learning, 2010, pp. 679–686.

[9] W. Luo, H. Li, G. Liu, Automatic Annotation of Multispectral Satellite Images Using Author-Topic Model, IEEE Geoscience and Remote Sensing Letters 9 (2012) pp. 634–638.

[10] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proc. of the International Conference on Machine Learning, 2003, pp. 912–919.

[11] D. Zhou, Q. Bousquet, T. N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: Proc. of Advances in Neural Information Processing Systems, 2004, pp. 321–328.

[12] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, Manifold-ranking based image retrieval, in: Proc. of the International Conference on Multimedia, 2004, pp. 9–16.
[13] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, Automatic multimedia cross-modal correlation discovery, in: Proc. of the International Conference on Knowledge Discovery and Data Mining, 2004, pp. 653–658.
[14] H. Tong, J. He, M. Li, W.-Y. Ma, H.-J. Zhang, and C. Zhang, Manifold-ranking-based keyword propagation for image retrieval, EURASIP Journal on Applied Signal Processing 2006 (2006) 1–10.
[15] H. Tong, J. He, M. Li, C. Zhang, and W.-Y. Ma, Graph based multi-modality learning, in: Proc. of the International Conference on Multimedia, 2005, pp. 862–871.
[16] M. Wang, X.-S. Hua, X. Yuan, Y. Song, L.-R. Dai, Optimizing multi-graph learning: towards a unified video annotation scheme, in: Proc. of the International Conference on Multimedia, 2007, pp. 862–871.
[17] Y. Liu, T. Mei, X. Wu, and X.-S. Hua, Multigraph-based query-independent learning for video search, IEEE Transaction on Circuits and Systems for Video Technology 19 (2009) 1841–1850.
[18] D. Rao, D. Yarowsky, Ranking and semi-supervised classification on large scale graphs using map-reduce, in: Proc. of the Workshop on Graph-based Methods for Natural Language Processing, 2009, pp. 58–65.
[19] U Kang, C. E. Tsourakakis, C. Faloutsos, PEGASUS: mining peta-scale graphs, Knowledge and Information Systems 27 (2011) 303–325.
[20] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, ACM Communications of the ACM 51 (2008) 107–113.
[21] X. Zhu, Semi-supervied learning literature survey, Technical Report of University of Wisconsin Madison (2008).
[22] A. Broder, On the resemblance and containment of documents, in: Proc. of the Compression and Complexity of Sequences, 1997, pp. 21–29.
[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, All-pairs shortest paths, Introduction to Algorithms, The MIT Press, 2009, pp. 684–707.
[24] G. Zhu, S. Yan, Y. Ma, Image tag refinement towards low-rank, content-tag prior and error sparsity, in: Proc. of the International Conference on Multimedia, 2010, pp. 461–470.
[25] Y.-G. Jiang, J. Wang, S.-F. Chang, Lost in binarization: query-adaptive ranking for similar image search with compact codes, in: Proc. of the International Conference on Multimedia Retrieval, 2011.
[26] Y.-H. Yang, P.-T. Wu, C.-W. Lee, K.-H. Lin, W. H. Hsu, H. Chen, ContextSeer: context search and recommendation at query time for shared consumer photos, in: Proc. of the International Conference on Multimedia, 2008, pp. 199–208.
[27] Y.-H. Kuo, K.-T. Chen, C.-H. Chiang, W. H. Hsu, Query expansion for hash-based image object retrieval, in: Proc. of the International Conference on Multimedia, 2009, pp. 65–74.