Learning a Discriminative Model for Image Annotation

Jiwei Hu, Chensheng Sun, and Kin Man Lam
Centre for Signal Processing, Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hong Kong
E-mail: {07901714r, 07900270r, enkmlam} @ polyu.edu.hk

Abstract—**This paper introduces a new discriminative model for image annotation. To learn the discriminative model, our method divides each training image into patches, and embeds the patches into a hypergraph, so as to find the representative instances (also called exemplars) for every single class by solving the graph. Then, the feature differences between the training samples and the exemplars are used to form new feature vectors for the training process. We aim to prune the specific features for each single label and formalize the annotation task as a discriminative classification problem. The kernel methods are also employed to solve the problem. Experiments are performed using the Corel5K dataset, and provide a quite promising result when comparing with other existing methods.**

I. INTRODUCTION

Nowadays, an increasing number of people enjoy taking photos and browsing pictures on the Internet. With the huge development in computer science, it is more convenient to store and manage a large number of digital images, not only on our computer hard disk but also in cyberspace. However, it is not always easy to find a target image in a short time without the knowledge of the tag of that image. If we can successfully use some text keywords to reflect the content of every single image, image management and browsing will become an easy task. Image annotation is the right method to solve this problem, as it assigns relevant keywords to images, representing their content. However, automatic image annotation is a very difficult task because of the lack of a direct relationship between the keywords and the image content. The so-called semantic gap problem always results in a bad performance for image-to-word mapping.

In recent years, various learning methods have been proposed by many researchers for automatic image annotation. These methods have in common that they all rely on a set of labeled pictures to learn a model, which can then predict the labels for the unlabeled data. The literature can be grouped based on three models: generative models, discriminative models and nearest-neighbor-based models. Most generative models [17,18] construct a joint distribution over image contents and the keywords while finding a mapping between the image features and annotation keywords. These generative models aim to learn a single model for all the vocabulary terms, which yields a better modeling in terms of dependencies. Some methods treat the task of image annotation as several binary classification problems. This means that the joint distribution of the unobserved variables and the observed variables is not needed. In this situation, discriminative models [3,4] can generally yield a superior performance. These discriminative models learn a separate classifier for each single label, and use the classifier to judge whether the test image belongs to this class or not. Although the training process is complicated and time consuming, this approach can, with a smart design, achieve more promising performances than the generative models. The third model, as one of the oldest, simplest, and most effective methods for pattern classification, is the KNN-based model [19], which is accurate, especially with an increasing number of training data. Recently, a NN-based keyword transfer approach was proposed in [11]. In this method, the labels are transferred from neighbors to a given image after a simple distance calculation. The nearest neighbors are determined using the Joint Equal Contribution (JEC) only, which finds the average distance obtained from the differences in image features. The method was extended [13] to filter out most irrelevant labels, with a promising result obtained.

The representations of image features play an important role in the abovementioned models. In [1], a graph structure was proposed to describe the relations of the features. In this approach, a pair-wise graph is constructed, with each vertex representing a single image that may be labeled or unlabeled. Two similar images are connected by an edge, and the edge weight is calculated as an image-to-image distance. [2] extended the concept of a simple graph to a hypergraph. The main argument is that the simple graph cannot completely represent the relations among images. Actually, the hypergraph can contribute to a better representation of the relations among images by considering not only the local grouping information, but also the similarities between the hyperedges that involve more than two images.

This paper proposes a method to learn a discriminative model using the knowledge of KNN-based distance measurement for image annotation. In our algorithm, we divide each image instance into 5 windows (including 2 by 2 non-overlapping windows and a window of the same size located at the image center), and embed these windows or patches of the training samples into a hypergraph. Then, the exemplars of each class can be derived from the hypergraph.

The feature differences of the training samples and the exemplars are computed, and a similarity vector is then constructed. The kernel-based method is further introduced for training the classifier for each exemplar, and the combined classifiers are trained for each label. For a query image, the best few labels will be selected as the final results based on the trained classifiers.

The remainder of this paper is organized as follows. In Section Ⅱ, we give an overview of our method. Then, a brief introduction of related works will be described in Section Ⅲ. We present our proposed method in detail in Section Ⅳ. The experiment set-up and results, and a conclusion, are given in Sections Ⅴ and Ⅵ, respectively.

II. An Overview of our Framework

In this section, an overview of our algorithm will be given. Our algorithm is divided into a training stage and a testing stage, as shown in Fig. 1 and Fig. 2, respectively.

Training

Input: Training images I_n (training images are categorized as 1, 2, 3, …, n classes)

Step 1: Divide the training images into 5 sub-windows (including 2 by 2 non-overlapping windows and a window of the same size located at the image center), and extract the CPAM features [12] and other features of each patch (denoted as I_n^i).

Step 2: Embed all the patches into a hypergraph (see the details in Section 3), and obtain the representative exemplars for each label class.

Step 3: Compute the feature differences between the patches of the training images and the exemplars so as to form delta-vectors (25 delta-vectors generated for each image pair).

Step 4: Use a kernel-based method to train a classifier for each exemplar, such that

$$\hat{w} = \arg\min |wx - y|_2^2, \quad (1)$$

where x is the delta-vector and y is the observation.

Fig.1. The training procedures of our proposed algorithm.

III. Related Work on the Hypergraph and Feature Selection

A. Hypergraph

In the field of machine learning, the pairwise relationships among different objects are always considered. These relationships are often embedded into a graph. A vertex can represent an object, and the different vertices can be connected by edges directly or indirectly. Since the pairwise relationships can be represented quite well by a simple graph, graph-based representation has recently attracted more and more attentions. However, this simple graph representation will inevitably result in a loss of information if only pairwise

Testing

Input: Testing image I_q

Step 1: Use the same procedures as in the training stage to extract the delta-vector of I_q.

Step 2: Use the classifier to output the score of each patch of the query image with respect to the exemplars of the corresponding class, and then sum the total value of the 5 patches.

Step 3: Choose the highest 5 and 10 scores, and select the labels of these classes as the final labels of the query image.

Fig. 2. The testing procedures of our proposed algorithm.

information is used to describe the complicated relationships among the objects. [2] introduced the concept of the hypergraph, which can more completely represent the relationships among objects. In a hypergraph, an edge can connect more than two vertices. This edge is called a hyperedge, which can be considered as a set of several vertices, rather than two vertices only. In other words, a set of vertices is defined as a weighted hyperedge; and the weight of a hyperedge can, to a certain extent, represent the degree of the vertices in this hyperedge belonging to one cluster.

When a hypergraph has been constructed, the next problem is how to solve it. In [15], Zhou et al. employed a spectral clustering technique to partition the graph, similar to the normalized cut approach in [16]. Furthermore, the concept of hypergraph Laplacian has been introduced, which is derived by relaxation to approximately obtain the hypergraph normalized cuts.

Let V denote a finite set of image instances (also represented as a set of vertices of a hypergraph), and E denote a subset of V such that $\cup_{e\in E} = V$. A hypergraph is constructed as G= (V, E, W) with the vertices V, the hyperedge E, and the weight W for each hyperedge. A hypergraph can be represented by a $|V|\times|E|$ incidence matrix H with entries $h(v,e)$ = 1 if $v \in e$, and 0 otherwise. In this model, all the vertices in a hyperedge are treated equally; this inevitably results in the loss of information. [2] proposed a softmax probability model to replace the binary assignments to the entries. In the model, each vertex is taken as a 'centroid' node in turn, and the hyperedge is formed by the vertex concerned and its k nearest neighbors. The incidence matrix H is defined by a distance measurement $D(j, i)$ for the i^{th} vertex and the j^{th} hyperedge as follows:

$$h(v_i, e_j) = \begin{cases} D(j,i), & \text{if } v_i \in e_j \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

In this formulation, the vertex v_i is softly assigned to the hyperedge e_j based on the similarity measurement $D(j, i)$. In this hyperedge, v_j is the centroid and v_i is the k-nearest neighbor of v_j. This probability model not only considers local grouping information, but also presents the probability that a vertex belongs to a hyperedge. The correlations among

different vertices can be well presented using this model. This also means that the structure of the vertices can be explored conveniently by referring to the probabilistic hypergraph model. For each hyperedge, we have a hyperedge weight computed as follows:

$$w(e_i) = \sum_{v_j \in e_i} D(i, j). \quad (3)$$

Unlike [2], our aim is not to solve the hypergraph for image retrieval. In other words, we need not partition the hypergraph for training. We simply embed the feature instances into the graph, and then we make use of the spatial structure to find the most representative patches for each label class. The hyperedge is the correlation between the centroid vertex and its *k* nearest neighbors. More specifically, assume that we have two vertices v_a and v_b belonging to the same class *m*. If the hyperedge weight related to the centroid vertex v_a is greater than that related to the centroid vertex v_b, we can say that the vertex v_a has more representative power than the vertex v_b has for the class *m*, to some extent. From this point of view, we can use the knowledge of the hyperedge weight to identify the most representative instances for each class, and the so-called exemplars can be found for each class by solving the following formulation:

$$i^* = \arg\max_i w(e_i). \quad (4)$$

This formulation means that, for each class, we indentify the *k*-nearest neighbors of each class sample that will form the most compact cluster. The compactness is measured by the sum of the similarity between the centroid sample and its neighbors.

B. Exemplar-derived Features

Although many image features have been proposed for image annotation, the properties of the different features and the combination of the features have not been well investigated. Moreover, the selected features may contribute unequally or even negatively to the performance. In [1], Zhang et al. presented a group-sparsity-based method to solve the feature related problems in the image-annotation tasks. The features used in our paper are inspired by [1], but the features are used in a different way.

In the Section Ⅲ.A , we have described how to construct the exemplars of each class. An exemplar represents the properties of the label class concerned, and we generate a new feature vector by computing the difference between the training samples and the exemplars. This new feature vector is called a *delta*-vector (Δ-vector). We assign the labels *y* of Δ (*i*, *j*) as a binary number:

$$y(i, j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to same class} \\ -1 & \text{otherwise.} \end{cases} \quad (5)$$

IV. An Image-Annotation Framework Based on a Discrimiative Model

A. Formalizing the annotation problem

In the image-annotation problem, an annotation system receiving a query image I_q will output its corresponding labels $l(q, t)$, where $l(q, t)$ refers to the set of tags *t* related to the query image *q*. Assume that the label set *L* contains *n* classes, and $l(q, t)$ is a subset *L*. An exemplar image is denoted as I_e^n, where *n* is the class index. The aim of the system is to find the tags t^* that maximize the conditional distributions $p(t| I_q)$.

In our algorithm, the Δ-vector is used instead of the traditional feature vector that directly describes an image instance. The Δ-vector is obtained by comparing the training images to the exemplars as follows:

$$\Delta = \left| \Upsilon(I^m) - \Upsilon(I_e^n) \right| \quad (6)$$

where I^m represents the m^{th} image in the dataset, and $\gamma(I)$ represents the feature set of image *I*. Now, the following formulation is used to obtain the output tags.

$$\begin{aligned} t^* &= \arg\max_t p(t \mid I_q) \\ &= \arg\max_t p(t \mid \Delta) \\ &= \arg\max_t \sum_i p(t \mid \Delta_i). \end{aligned} \quad (7)$$

Thus the annotation task is performed in a new feature space composed of the difference vectors Δ_i instead of the original image feature space.

B. Model Parameterization

In our algorithm, an image-annotation model is to be learned as discriminatively as possible. In order to achieve this, a separate classifier is learned for each keyword, and is used to decide whether a query image belongs to the class or not. In our case, we change the traditional image-to-tags problem to a new feature-weighting problem. An image is described by dividing it into regions, with each region represented by a Δ-vector. Then, a classifier is trained for each label class; the input to the classifier is the Δ-vectors, and the output is a probability value ranging from 0 to 1. In other words, a scoring function that measures the match between the visual differences and the high-level concept is designed. Specifically, we design the classifier thus:

$$F(\Delta_q) = \Psi\left(\left\{ f_i\left(\Delta_{q,i}\right) \right\}_i \right), \quad (8)$$

where $\Delta_{q,i}$ represents the differences between the query image *q* and the i^{th} exemplar, the function *f* is the classifier to be designed (details will be shown in Section Ⅳ.C), and the function Ψ is used to replace the simple summation of all the outputs of the classifier in (7).

In the training steps, for any training image, the Δ-vector is firstly obtained by comparing the image *I* and the exemplars of that class, and then we compute the softmax probability of this vector belonging to the m^{th} class. However, the differences between the image *I* and the different exemplars may contribute unequally to the final Δ-vector. More specifically, assume that we have a number of *k* exemplars for each class; the Δ-vectors obtained from the image *I* and the *k*

exemplars form a Δ-matrix, $[|\Delta_1|, |\Delta_2|, \ldots |\Delta_k|]^T$. Then, we explore the relationship among the different Δ-vectors. In other words, we want to find a mapping from each Δ-vector space to the tag space, i.e. minimizing the differences between the positive samples in a class and its corresponding exemplars with respect the observations. Now, we have converted the mapping problem into a metric-learning approach.

The parameterization of the mapping is inspired by [3]:

$$\Phi : T \times P \rightarrow \{0,1\}, \text{ where } \Phi_i = \varphi(\Delta, \theta). \qquad (9)$$

In (9), P is the image feature space, and T is the space of tags. The function Φ is a number of 1-vs-all classifiers that determine if an input image has the corresponding label. The function Φ is employed for certain mappings and θ is a parameter in the function φ. In this case, we only assign a binary output to the function φ, and we learn a weighting function for each exemplar.

Having determined the mapping function for each exemplar, we can formalize the Δ-matrix as $[\Phi_1(\Delta_1), \Phi_2(\Delta_2),\ldots, \Phi_m(\Delta_m)]^T$. Then, we can use a single SVM classifier to train a classifier for each class. At this stage, we know that the key point is to find a mapping for each single exemplar of each class. Next, we will show how to determine the parameter θ, and then how to learn an individual classifier for each exemplar.

C. *The Rank-SVM-Based Learning Algorithm*

Our target is to learn an optimal value of θ such that the distance between the exemplar and all positive samples in the corresponding class are as small as possible, while the distance between the exemplar and all negative samples in the other classes is as large as possible, i.e.

$$\theta^* = \arg\min_{\theta}[\frac{1}{p}\sum_{p}\theta(\Delta_{pe_i})^2 - \frac{1}{q}\sum_{q}\theta(\Delta_{ne_i})^2], \qquad (10)$$

where p is the number of positive samples, and q is the number of negative samples corresponding to the exemplar e_i. Δ_{pe_i} represents the Δ-vector between the positive samples and the exemplars, and Δ_{ne_i} is the Δ-vector between the negative samples and the exemplars.

To solve θ to minimize (10), we can transform this formulation into a ranking problem. This means that we can interpret the formulation as the projection of a Δ-vector onto a vector θ. After the projection, the ranking of Δ_{pe_i} should always be higher than that of Δ_{ne_i}. Specifically, we convert the problem into:

$$f^* = \arg\min_{f \in F}[\frac{1}{pq}\sum_{i=1}^{p}\sum_{j=1}^{q} l(f, \Delta_{pe_i}, \Delta_{ne_i}) + \lambda N(f)], \qquad (11)$$

where $l(\,)$ is the convex upper bound on 1 ($\Delta_{pe_i} < \Delta_{ne_i}$), $N(\,)$ is a regularizer, $\lambda > 0$ is the regularization parameter, and F is a ranking function. There are many learning algorithms that can solve (11).

We introduce some slack variables and take the Lagrangian dual results in the following convex quadratic program (QP) over pq variables $\{\alpha_{ij}: 1 \le i \le p, 1 \le j \le q\}$:

$$\min_{\alpha}[\frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{p}\sum_{l=1}^{q}\alpha_{ij}\alpha_{kl}\phi(\Delta_{pe_i}, \Delta_{ne_i}, \Delta_{ke_i}, \Delta_{le_i}) - \sum_{i=1}^{p}\sum_{j=1}^{q}\alpha_{ij}] \qquad (12)$$

$$\text{subject to } 0 \le \alpha_{ij} \le C \quad \forall i, j.$$

where $C = 1/\lambda pq$, and

$$\phi(\Delta_{pe_i}, \Delta_{ne_i}, \Delta_{ke_i}, \Delta_{le_i}) = (K(\Delta_{pe_i}, \Delta_{ke_i}) - K(\Delta_{pe_i}, \Delta_{le_i}) - K(\Delta_{ne_i}, \Delta_{ke_i}) + K(\Delta_{ne_i}, \Delta_{le_i}))$$

This formulation can be solved using a standard QP solver, or other more efficient methods [20].

V. Experiments and Results

The experiments are divided into two parts. First, we will construct the exemplars for each class and show the image patches that are chosen as the exemplars of some classes. Then, we will evaluate the performance of our image-annotation framework using the Corel 5K dataset.

Corel 5K contains 5,000 images comprising 4,500 training and 500 testing samples. Each image in the dataset is annotated with about 3.5 keywords on average, and the dictionary has a total of 374 words or labels.

As described previously, different types of features are first subtracted from 5 sub-windows. Fig. 3 shows examples of the patches/sub-windows used.

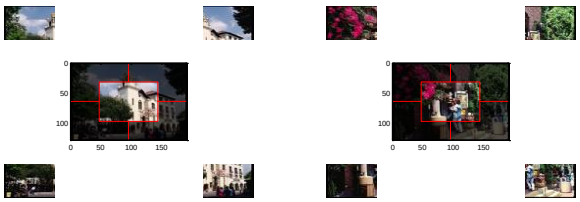


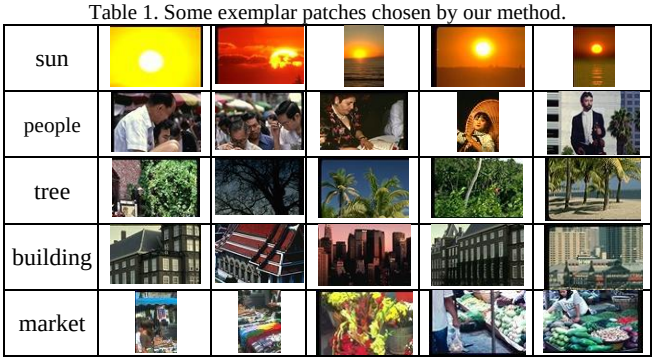
Fig.3. Five sub-windows chosen as image patches for feature extraction.

The first experiment will show the kinds of image patches to be chosen as exemplars for different classes. The images in Corel 5K have significant differences for different classes, but also have large variations within images in the same class. Thus, it is useful to find the "good exemplars" for each class. In this experiment, we have employed the Colored Pattern Appearance Model (CPAM) feature only, which has been proven to be simple and efficient [11].

Table 1 illustrates some of the patches chosen as exemplars for some classes. We can see that some classes have very good and representative exemplars, even though every image instance is simply divided into 5 sub-windows. However, some patches may not clearly reflect some of the classes. Indeed, a big semantic gap obviously exists between the low-level features and the class labels like "market". Nevertheless, our method can identify good exemplars in most cases. We

believe that identifying good exemplars plays an important role in the research of image retrieval.

Table 1. Some exemplar patches chosen by our method.



We have evaluated the performance of our algorithm with an increase in the number of exemplars used. By considering the average number of examplars for each class, we have experimented with the mean precision (which is the key measurement for image annotation) for the 500 testing images based on 1, 3, 5, 7, and 9 exemplars. Table 2 shows that using five exemplars can result in the best performance in our method. So we choose 5 as the number of exemplars for each class in the experments that follow.

Table 2. The mean precision rates (*P%*) for using different numbers of exemplars.

No. of exemplars	1	3	5	7	9
(*P%*)	14.1	21.3	32	26.5	23.3

Next, we will evaluate the performance of our algorithm and compare it with other state-of-the-art annotation algorithms. After choosing the exemplars for each class, our algorithm computes the Δ-vectors and then performs classification.

Similar to [14], different feature descriptors, and a combination of these features, were used in our experiments. The following features are used for each image patch. We compute the Δ-vectors as the differences between the input patches and each of the exemplars.

(1) Color feature: RGB color moment (3×3 grid, color mean, variance, skewness for R, G, B),
(2) Edge histogram (edge-orientation histogram, Canny edge detector),
(3) Gabor wavelets transform (5 scales and 8 orientations, 3 moments for each sub-image),
(4) Local binary pattern (a 59-d LBP histogram), and
(5) GIST (a complex and popular feature descriptor).

We choose three measurements to evaluate the performances of our proposed method and other methods. Three performance indices are measured: the mean precision rates (*P%*), the mean recall rates (*R%*), and the number of total keywords recalled (N^+).

Table 3. Performances based on the Corel5K dataset for some existing methods and our proposed method.

Methods	*P%*	*R%*	N^+
CRM [5]	16	19	107
InfNet [6]	17	24	112
NPDE [7]	18	21	114
MBRM[10]	24	25	122
SML [8]	23	29	137
TGLM [9]	25	29	131
JEC [11]	27	32	139
LASSO[11]	24	29	127
TagProp[4]	33	42	160
Proposed	32	38	151

Table 3 shows the performances of our proposed kernel-based discriminative model versus some existing methods using Corel 5K. Our method outperforms most of the other methods, and is comparable to TagProp; in particular, in terms of the mean precision rate, which is the most important measurement.

Learning an individual classifier for each class is complicated and time consuming work for image annotation. However, this work presents an efficient algorithm that formalizes a kernel mapping task into a ranking problem. We make use of the hypergraph, and extract a new feature vector after obtaining the exemplars for each class. After the classifiers have been learnt for each exemplar, a simple sigmoid function is used to combine the results of various exemplar classifiers to make a decision about the image label. The experiments on Corel 5K provide quite promising results.

Fig. 4 shows that our algorithm has a higher probability than TagProp of containing at least one correct label than TagProp. Although TagProp has a slight better performance in terms of the mean precision rates, our method achieves a higher efficiency in finding the correct labels. That means, TagProp sacrifice more time in finding the correct labels comparing with our work.

Table 4 shows some results for the labels obained from our algirothm and the ground-truth. We see that our algorithm can achieve at least one correct label in most cases.

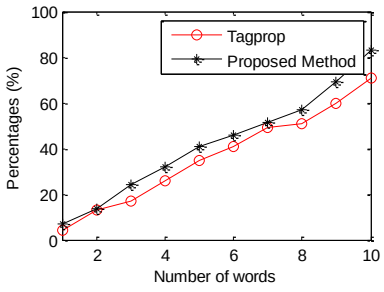


Fig.4. Percentages of number of testing images correctly annotated by at least one word among the top *n* words using the discriminative model "TagProp" and our proposed method.

Table 4. Predicted labels versus ground-truth (difference are marked in *italic*)

				
Predicted labels	tree sky sun clouds *shadow*	tree beach sky *sea clouds*	snow *cat* tree *cliff stone*	sea *sky waves* water boats
Ground Truth	clouds sky sun tree	beach palm people tree	fox snow tree wood	Beach Water boats
				
Predicted labels	building *sky* tree street *people*	tree *sky* water house *building*	*building* house water *tree market*	horses tree *bush* foals *fields*
Ground Truth	building sculpture street tree	boats house tree water	canal house people water	foals horses mare tree

VI. CONCLUSIONS

This paper presents an efficient algorithm for learning a kernel-based discrimantive model for image annotation. We define a new feature repesentation by taking the exemplars of each class label into consideration. Based on that, we convert the feature-to-label mapping task into a ranking problem. A rank-SVM-based learning algorithm is introduced, and the convex optimization method is used to solve the aglorithm. Our method is very straightforward, and the experiments have proven that our new discrimantive model can achieve comparable performances with the state-of-the-art approach but consumes less time in finding the correct labels.

Acknowledgement: The work described in this paper was fully supported by a grant from the Research Grants Council of the HKSAR, China (Project No. PolyU 5192/07E).

REFERENCES

[1] Shaoting Zhang, Junzhou Huang and etc, "Automatic Image Annotation Using Group Sparisty," CVPR 2010, pp.3312-3373.

[2] Yuchi Huang, Qingshan Liu and etc, "Image Retrieval via Probabilistic Hypergraph Ranking," CVPR 2010, pp.3376-3383.

[3] David Grangier and Samy Bengio, "A Discriminative Kernel-Based Model to Rank Images from Text Queries," *IEEE TPAMI*, vol.30, no.8, pp. 1371-1384, Aug. 2008.

[4] M.Grubinger, T.Mensink, J.Verbeek, and C.Schmid "Tagprop: Discriminative Metric Learning In Nearest Neighbor Models for Image Auto-Annotations," ICCV 2009, pp. 309-314.

[5] V.Lavrenko, R.Manmatha and J.Jeon. "A Model For Learning the Semantic of Pictures," NIPS 2003

[6] D. Metzler and R. Manmatha, "An Inference Network Approach to Image," CIVR 2004, pp. 42-50.

[7] A. Yavlinsky, E. Schofield, and S. Ruger. "Automatic Image Annotation Using Global Features and Robust Nonparametric Density Estimation," CIVR2005, pp.507-517.

[8] G. Carneiro, A.B. Chan et al., "Supervised Learning of Semantic Classes For Image Annotation and Retrieval," *IEEE TPAMI*, vol.29, no 3, pp. 394-410, 2007;

[9] J. Liu, M. Li, Q. Liu, et al., "Image Annotation via Graph Learning," *Pattern Recogn*, vol.42, no2, pp.218-228, 2009

[10] S.L. Feng, R. Manmatha, and V. Lavrenko, "Multiple Bernoulli Relevance Models for Image and Video Annotation." CVPR 2004, vol.2, pp.1002-1009.

[11] A. Makadia, V. Pavlovic, and S. Kumar, "A New Baseline for Image Annotation," ECCV 2008.

[12] G. Qiu, "Image indexing using a coloured pattern appearance model," *Proc. Storage and Retrieval for Media Databases* 2001.

[13] Jiwei Hu, Kin-Man Lam, and Guoping Qiu, "A Hierarchical Algorithm for Image Multi-labeling," ICIP 2010.

[14] Jianke Zhu, Steven C.H.Hoi, Michael R. Lyu and Shuicheng Yan, "Near-Duplicate Keyframe Retrieval by Nonrigid Image Matching," ACM Multimedia 2008,

[15] D. Zhou, J.Huang, and B.Scholkopf, "Learning with Hypergraphs: Clustering, Classification, and Embedding," NIPS2006.

[16] Jianbo Shi and Jitendra Malik, "Normalized Cuts and Image Segmentation," *IEEE TPAMI*, vol.22, no.8, pp.888-905 Aug. 2000.

[17] K.Barnard, P.Duygulu and etc, "Matching Words and Pictures," *Journal of Machine Learning Research*, vol.3, pp.1107-1135, 2003.

[18] F.Monay and D.Gatica-Perez, "Plsa-Based Image Auto-Annotation: Constraining the Latent Space," *ACM Multi-media*, pp. 348-351, 2004

[19] H.Zhang, A.Berg, M.Maire, and J.Mailik, "SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition" CVPR 2006, pp.2126-2136.

[20] Le, Q. V., Smola, A., Chapelle, O., & Teo, C. H., "Optimization of Ranking Measures," accepted by *Journal of Machine Learning Research*, 2010