Random Forest for Image Annotation

Hao Fu, Qian Zhang, and Guoping Qiu

School of Computer Science,
University of Nottingham, Nottingham, UK
{hxf,qxz,qiu}@cs.nott.ac.uk
http://www.viplab.cs.nott.ac.uk/

Abstract. In this paper, we present a novel method for image annotation and made three contributions. Firstly, we propose to use the tags contained in the training images as the supervising information to guide the generation of random trees, thus enabling the retrieved nearest neighbor images not only visually alike but also semantically related. Secondly, different from conventional decision tree methods, which fuse the information contained at each leaf node individually, our method treats the random forest *as a whole*, and introduces the new concepts of semantic nearest neighbors (SNN) and semantic similarity measure (SSM). Thirdly, we annotate an image from the tags of its SNN based on SSM and have developed a novel learning to rank algorithm to systematically assign the optimal tags to the image. The new technique is intrinsically scalable and we will present experimental results to demonstrate that it is competitive to state of the art methods.

Keywords: Random Forest, Image Annotation, Semantic Nearest Neighbor.

1 Introduction

Nearest Neighbor (NN) based methods have been successfully applied to various problems in computer vision including, image classification [1], scene completion [2], image parsing [3], image annotation [4,5], etc. In this paper, we are particularly interested in using NN to deal with problems related to image annotation.

In order to extend nearest neighbor based methods to large scale settings, we need to carefully consider at least two significant issues. The first is how to design efficient data structures to retrieve the nearest neighbors. Some well known techniques including inverted file structure [6] or Locality Sensitive Hashing (LSH) [7] are usually adopted to deal with this problem. The second is the semantic gap problem where the nearest neighbors retrieved based on visual feature similarity do not necessarily share the same semantic concepts. This is a fundamental problem in computer vision and hundreds of different methods have been developed.

In the past, the two problems mentioned above were usually dealt with separately. In this paper, we propose to use random forest to tackle them simultaneously: using the tree structure of the random forest enables the efficient

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 86–99, 2012.

retrieval of nearest neighbors; using the tag distribution as supervising information to guide the generation of the random trees makes the images located at the same leaf node share similar semantic concepts. Furthermore, we exploit the specific structure of the random forest and propose a new concept called *semantic similarity measure* (SSM). Based on this SSM, we define *semantic nearest neighbors* (SNN) in contrast to the traditionally used visual feature space nearest neighbors.

We made three contributions in this paper. Firstly, we propose to use the tags contained in the training images as the supervising information to guide the generation of the random trees, thus enabling the retrieved nearest neighbors not only visually alike but also semantically similar. Secondly, different from conventional decision tree methods, which fuse the information contained at each leaf node individually, our method considers the random forest *as a whole*, and introduce the new concepts of semantic nearest neighbors (SNN) and semantic similarity measure (SSM). Thirdly, we annotate an image from the tags of its SNN and have developed a novel learning to rank algorithm to systematically assign tags to the image. We present experimental results to demonstrate that our technique is competitive to the state of the art.

2 Related Work

The task of image annotation is to automatically assign metadata, usually in the form of keywords or tags, to an image. Large quantity of literature exists in the image annotation area, and methods range from generative models [8] to discriminative models [9]. While generative models need a large quantity of training data to learn the joint probability of semantic concepts and image visual features, discriminative models treat each tag as a semantic class, and try to learn a different classifier for each tag. Therefore, it is not easy to model the correlations between tags and one will encounter difficulties when extending to large number of tags.

Recently, nearest neighbor based methods [4,5] have attracted much attention. Among them, TagProp [4] is perhaps the most successful method which shows superior performance on several benchmark image annotation datasets. It uses a weighted nearest neighbor model to predict the possible tags. Its superior performance relies on its sophisticated training procedure, and its optimization function is composed of items corresponding to every tag in every image, which will inevitably hinder its applicability to large scale datasets. The success of TagProp largely motivates the work presented in this paper which also adopts the nearest neighbor based models for image annotation but with much improved efficiency and flexibility.

On the other line of research, to accelerate the speed of retrieving nearest neighbors, some approximate nearest neighbor approaches are usually adopted. Among them, hashing [10] is generally believed to be the most prominent one. It uses a series of hashing functions to project the samples to different buckets, in the hope that similar samples will have a higher probability of falling

into the same bucket. Most of the existing hashing techniques are unsupervised [11,12], although supervised [13] or semi-supervised [7] hashing techniques exist and always show better performance in image retrieval applications. However, for those supervised hashing methods, the supervised information only takes the form of pairwise similarity [7,13]. They can not make full use of the tag information available in image annotation scenario. Besides this, each hashing function is just a linear projection. This simple form limits its discriminative power in dealing with highly non-linear data.

Besides hashing, another popular Approximate Nearest Neighbor (ANN) search method is the tree structure based method [14,15,16]. Sometimes these tree structure methods exhibit a better performance than hashing [16]. It has also been observed that multiple random trees usually outperform a single tree [14,16]. As these tree structure models only aim for fast retrieving nearest neighbors, they are all unsupervised. Although there exists some work on utilizing supervised random trees for nearest neighbor search [17,18], they are mainly used as a fast alternative to k-means for deriving low-level feature representations and the random trees played no direct role in the final image classification which is usually done by classifiers such as SVMs.

The most related work to ours is in [19], who also propose to use random forest for image annotation. However, their usage of random forest is totally different from ours: they treat their random forest as a replacement of Gaussian mixture model embedded in a previous annotation model named Semantic Multiclass Model [9], whilst our new annotation model is based entirely on the random forest; what we stored at each leaf node is the training images that fall onto this leaf node, whilst theirs is the posterior probability of tags.

3 The Construction of the Random Forest

It has been a common practice [5,4] to use multiple kinds of features to represent one image. How to efficiently combine different features is a non-trivial task [20]. Existing methods include using an equal weight for different features [5], a distance specific weight for each feature [4], etc.

In our random tree scenario, the method of fusing different features is correlated with the choice of the split function of the random tree. Given the feature representation $\mathbf{F}$ for a sample, there are many possible ways of defining the split function at each node, such as a linear classifier [21,22]:

$$\begin{cases} \mathbf{w^T F} + b \geq 0 & \text{go to left child} \\ \text{otherwise,} & \text{go to right child} \end{cases} \tag{1}$$

or the feature difference between two dimensions [23]:

$$\begin{cases} F_i - F_j \geq thresh & \text{go to left child} \\ \text{otherwise,} & \text{go to right child} \end{cases} \tag{2}$$

However, the above two methods will both encounter problems when the feature dimension is high. As for a typical decision tree, we need to generate multiple hypothesized tests and pick the best one. If the feature dimension is high, then the valid search space for the above two methods would grow quickly as well, thus making it harder to pick up a good split. Although we can formulate it as an SVM problem [22] for choosing the appropriate linear classifier, it is only applicable for problems with explicit class labels.

To narrow down the search space for the split function, we prefer to use a simple split function while simultaneously trying to maintain its discriminative power. To achieve this, we propose to use dimension reduction methods to get a more compact representation for each feature channel. More specifically, we choose to use kernel PCA [24] for each feature channel. Although directly computing kernel PCA is time consuming and hard to extend to large scale, there exist fast approximate methods [25] which make it scalable to large scale datasets. In all our experiments, we use kernel PCA to reduce each feature dimension to a fixed low dimension, thus making each kind of feature have an equal probability to be chosen as the dimension on which the split function will operate. Denote the kernel PCA reduced feature as $\mathbf{F}'$, the split function is defined as:

$$\begin{cases} F'_i \geq thresh & \text{go to left child} \\ \text{otherwise,} & \text{go to right child} \end{cases} \tag{3}$$

Based on the split function defined above, we can split samples to the left child node or right child node accordingly. We can generate multiple splits by choosing different feature dimensions or different thresholds. We plan to use the tag information to guide the generation of the tree. The idea is straightforward: at each node, after splitting the samples to the left node or the right node, we can compute their corresponding tag histograms. A good split means the tag histogram of the left node should be quite different from the tag histogram of the right node. Thus we can use the widely adopted information gain criteria [21,26] as the score function:

$$Score(split) = \triangle E = -\frac{|I_l|}{|I_n|}E(I_l) - \frac{|I_r|}{|I_n|}E(I_r) \tag{4}$$

where $E(I)$ is the Shannon entropy of the tag distributions in the set of samples I. $|I|$ means the number of samples contained in I. I_n is the set of training sample in node n, while I_l and I_r represent the training images contained in node n's left and right child node respectively.

4 Tag Prediction Based on Semantic Neighbors

For a test image, we pass it through every random tree. It falls from the root node and keeps falling according to the split function until it reaches the leaf node. It is obvious that every node in its falling trajectory contains important information, but for now we only consider the samples stored in the leaf node

and call these samples the *semantic neighbors* of the test sample. The semantic neighbors obtained from different trees together make up the *semantic neighbor set*. Based on this semantic neighbor set, we can draw the important rationale behind this paper: the more often that two images fall into the same leaf node, the more likely they share similar tags. Therefore, the semantic similarity between two images should be monotonically increasing with the number of trees in which they fall into the same leaf node. Thus, we can count the number of trees that two images fall into the same leaf node and use this *count* value as the *semantic similarity measure* (SSM) of the two images (we will use *count* value and SSM interchangeably in the rest of the paper). Based on the SSM, we can sort all the images contained in the semantic neighbor set to retrieve its K semantic nearest neighbors. The concepts of semantic neighbor set and semantic neighbors are also illustrated in Fig.1.

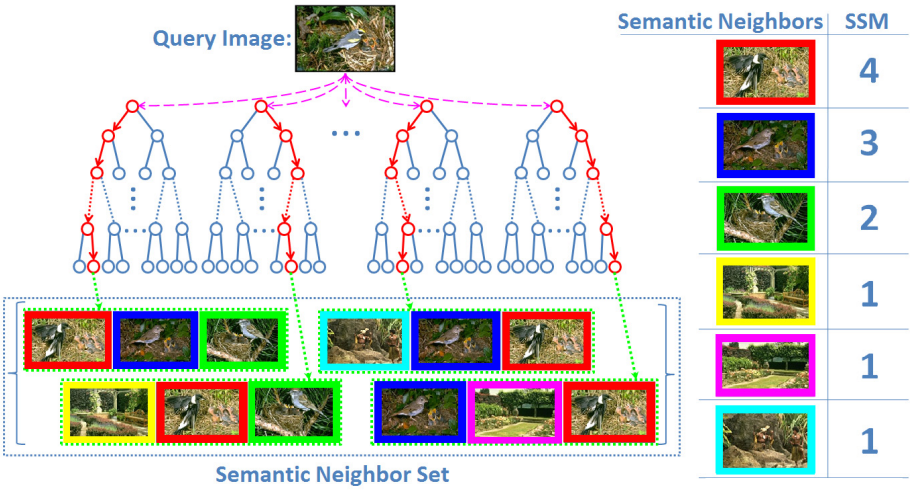

Fig. 1. An example showing the concepts of semantic neighbor set and semantic neighbors. A query image passes through all random trees. The training images stored at the leaf nodes on which the query image falls into form the semantic neighbor set. Based on this, the semantic similarity measure (SSM) between the query and a given training image is calculated as the number of times that the given image appears in the semantic neighbor set. A larger SSM indicates higher similarity.

Based on the semantic neighbor set, our semantic nearest neighbor (SNN)-based image annotation method is performed as follows: For a test image, we use the method described above to retrieve its K semantic nearest neighbors. The prediction of the tags for this image totally depends on these K semantic nearest neighbors. At this stage, we can use previously developed methods, like the label transfer method in [5] or the label filter method in [27]. In our case,

we can get additional help from the *count* value obtained from our random forest, and therefore we adopt a conventional tf-idf scheme [6].

Denote $\mathbf{I}$ the query image and $\mathbf{Q}$ the probabilities of assigning tags to annotate the image. Let $\mathbf{I_i}$ represents $\mathbf{I}$'s ith semantic neighbor returned by our random forest with its *count* value denoted as c_i, and $\mathbf{T_i}$ represents the ground truth tags of $\mathbf{I_i}$. Suppose there are M tags in total, thus $\mathbf{Q}$ and $\mathbf{T_i}$ can be represented as a M-vector: $\mathbf{Q} = (q_1, q_2, ..., q_M)^T$ and $\mathbf{T_i} = (t_{i1}, t_{i2}, ..., t_{iM})^T$. Here t_{ij} is an indicator function which is equal to 1 if tag j exists for the ith image. The prediction of $\mathbf{Q}$ totally depends on the $\mathbf{T_i}$ and c_i value:

$$q_j = \log \frac{R}{r_j} * \sum_{i=1}^{K} \left(\frac{t_{ij}}{Z} * f(c_i) \right), \ j \in \{1, 2, ..., M\} \tag{5}$$

where the term $\log \frac{R}{r_j}$ is the inverse document frequency [6] obtained from all the training images. R is the number of training images, and r_j is the number of training images who contain the jth tag. Z is a normalizing constant which is equal to $\sum_{i=1}^{K} \sum_{j=1}^{M} t_{ij}$. The term $f(c_i)$ represents a function which should be monotonically increasing with c_i. This term reflects our intention that the neighbor with a larger *count* value should contribute more to the predication of the tags. Based on the computed M-vector $(q_1, q_2, ..., q_M)^T$, we can predict l tags for the test image which correspond to the l largest q_j values.

Possible forms of $f(c_i)$ include $f(c_i) = c_i$, $f(c_i) = {c_i}^2$, etc. However, ad hoc choices of $f(c_i)$ is not fully convincing. In the next section, we introduce a systematic method to learn $f(c_i)$ from training data.

5 Image Annotation as Learning to Rank Semantic Neighbors

In conventional random forest literature [23], each tree is considered to be independent and they contribute equally to predict the posterior probability, which is equivalent to setting $f(c_i) = c_i$ in our scenario. However, we will show in our experimental section that sometimes if we choose $f(c_i) = {c_i}^2$, we can obtain a better performance. It is not difficult to understand this. As shown in Fig.2, the distribution of the *count* variable exhibits a heavy-tailed distribution. That's one of the reasons why the method of JEC [5] is successful by using only the first few nearest neighbors (which correspond to the larger *count* values in our scenario). It also stimulates our first intuition to use $f(c_i) = {c_i}^2$ to exaggerate the contribution of those large *count* values. However, choosing $f(c_i)$ in an ad hoc manner is not desirable and we need a more systematic approach.

Recall that the *count* value c_i defined in (5) can only take discrete values. Therefore we can define $f(c_i) = w_{c_i}$, where w_{c_i} means this variable only depends on c_i value and we can consider it as a look-up table. Suppose we have trained N_T trees in total, then the inequality $c_i \leq N_T$ always holds, and w_{c_i} can have at most N_T different values. We use $\mathbf{W} = (w_1, w_2, ..., w_{N_T})^T$ to denote the vector form of w.

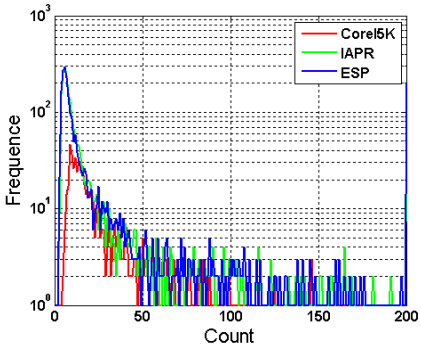

Fig. 2. Distribution of the *count* value across different datasets. Randomly sampled images from each dataset are fed into the random forest. The top K semantic nearest neighbors retrieved with their *count* values are gathered to plot this graph.

Denote $a_j = \log\frac{R}{r_j} * \frac{1}{Z}$, which is a tag specific constant, then (5) can be rewritten as:

$$q_j = a_j * \sum_{i=1}^{K}(t_{ij} * w_{c_i}), \quad j \in \{1, 2, ..., M\} \tag{6}$$

Here, we introduce another indicator variable $\mathbf{D} \in \{0, 1\}^{K \times N_T}$. Each of its row contains all zero but only one 1 which corresponds to the position of w_{c_i} in $\mathbf{W}$. Thus (6) can be written as:

$$q_j = a_j * [t_{1j}\ t_{2j}\ ...\ t_{Kj}] * \mathbf{D} * \mathbf{W} \tag{7}$$

Its matrix form is:

$$\begin{aligned} \mathbf{Q} &= \mathbf{A} .* ([\mathbf{T_1}\ \mathbf{T_2}\ ...\ \mathbf{T_K}] * \mathbf{D} * \mathbf{W}) \\ &= (\underbrace{[\mathbf{A}\ \mathbf{A}\ ...\ \mathbf{A}]}_{N_T} .* ([\mathbf{T_1}\ \mathbf{T_2}\ ...\ \mathbf{T_K}] * \mathbf{D})) * \mathbf{W} \end{aligned} \tag{8}$$

where $\mathbf{A} = [a_1, a_2, ...a_M]^T$, '.*' represents element-by-element multiplication.

Here, we can see that the prediction of tags can be casted as a linear prediction model. In the training procedure, we need to find the largest values in $\mathbf{Q}$, and make it equal to the ground truth annotation. We can cast it as a learning to rank problem [28].

Denote $\{(\mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{N})\}$ as the set of all possible tag pairs in the training set, where $\mathcal{T}_\mathcal{P}$ represents the ground truth tags and $\mathcal{T}_\mathcal{N}$ represents the rest of the tags. Let

$$\mathbf{\Psi_j} = a_j * [t_{1j}\ t_{2j}\ ...\ t_{Kj}] * \mathbf{D} \tag{9}$$

then (7) can be simplified as $q_j = \mathbf{\Psi_j} * \mathbf{W} = \langle \mathbf{\Psi_j}, \mathbf{W} \rangle$.

Then this learning to rank problem can be formulated as:

$$\min_{\mathbf{W},\xi_{ij}\geq 0} \quad \frac{1}{2}\mathbf{W^T W} + C \sum_{(i,j)\in\{(\mathcal{T}_\mathcal{P},\mathcal{T}_\mathcal{N})\}} \xi_{ij}$$

$$s.t.\ \forall (i,j) \in \{(\mathcal{T}_\mathcal{P},\mathcal{T}_\mathcal{N})\} : \langle \mathbf{\Psi_i}, \mathbf{W}\rangle \geq \langle \mathbf{\Psi_j}, \mathbf{W}\rangle + 1 - \xi_{ij}$$

$$\forall i :\ w_i \leq w_{i+1}, \quad \mathbf{W} \geq 0, \tag{10}$$

This problem differs from the conventional SVM problem [28] in that it has the additional constraints $\mathbf{W} \geq \mathbf{0}$ and $w_i \leq w_{i+1}$, but still it is a convex problem, and we can solve it using off-the-shelf convex problem solvers[1]. These additional constraints represent our belief that the contribution of the retrieved nearest neighbors should be monotonically increasing with the *count* value.

However, the problem defined in (10) is still different from our needs. The aim of the learning to rank problem, as defined in (10), is to make the rank of positive samples as high as possible, but in our scenario, we only need to predict the top l tags. This means there is no difference between ranking one positive tag as rank $l+1$ or rank $l+100$. Therefore, we introduce a slack variable ξ_i for each positive tag instead of ξ_{ij} for each positive-negative pair. This slack variable motivates the algorithm to make the rank of positive tags outperform all the negative tags. Our new learning to rank problem is defined as:

$$\min_{\mathbf{W},\xi_i\geq 0} \quad \frac{1}{2}\mathbf{W^T W} + C \sum_{i\in\mathcal{T}_\mathcal{P}} \xi_i$$

$$s.t.\ \forall (i,j) \in \{(\mathcal{T}_\mathcal{P},\mathcal{T}_\mathcal{N})\} : \langle \mathbf{\Psi_i}, \mathbf{W}\rangle \geq \langle \mathbf{\Psi_j}, \mathbf{W}\rangle + 1 - \xi_i$$

$$\forall i :\ w_i \leq w_{i+1}, \quad \mathbf{W} \geq \mathbf{0}, \tag{11}$$

One obstacle to directly solving (11) is that it will generate huge number of constraints. For example, if a dataset contains 5000 images for training, each image is annotated with 4 tags on average and the size of the tag set is 260, then one image will generate 4*256≈1000 constraints, and the whole dataset will generate about 5 million constraints! However, with the help of the additional constraints $\mathbf{W} \geq \mathbf{0}$ and $w_i < w_{i+1}$, we will show that most of the constraints are redundant and we can reduce the size of the constraints by almost two orders of magnitude.

Definition 1. *Tag m is **superior** over tag n if $q_m = \langle \mathbf{\Psi_m}, \mathbf{W}\rangle \geq q_n = \langle \mathbf{\Psi_n}, \mathbf{W}\rangle$ for every $\mathbf{W} \in \{\mathbf{W} \geq 0, w_i \leq w_{i+1}\}$.*

From the above definition, we can see that if tag m is superior to tag n, then we will always prefer tag m over tag n in predicting the tags. If tag m is the ground truth annotation, then we will always be right. On the contrary, there will be no

[1] `http://cvxr.com/cvx/`

hope to remedy the mistake. Therefore, such constraints can be considered as redundant and there is no need to add them to our optimization problem. Using the following proposition, we can quickly judge if tag m is superior over tag n.

Proposition 1. *Denote* $\mathbf{\Psi_j}$ *which is defined in (9) as* $[\psi_{j1}\ \psi_{j2}\ ...\ \psi_{jN_T}]$. *Tag* m *is superior over tag* n *if and only if* $\sum_{i=N}^{N_T}\psi_{mi} \geq \sum_{i=N}^{N_T}\psi_{ni}$ *for every* $N \in \{1, 2, ..., N_T\}$. *[The proof of this proposition can be found in the supplementary material* [2]*.]*

Based on this proposition, we can use the following procedure to find the redundant pairs and remove them from the constraint set:

1). Let $\mathbf{\Psi} = [\mathbf{\Psi_1}; \mathbf{\Psi_2}; ...; \mathbf{\Psi_M}]$. Rearrange the rows of $\mathbf{\Psi}$ into the **R**ight-**O**rdered form $\mathbf{\Psi_{RO}}$;
2). Calculate the cumulative sum of each row vector in the reverse order $\mathbf{\Psi_{cum}}$;
3). We can judge that tag m is **not** superior to tag n if either of the following two conditions hold: the position of $\mathbf{\Psi_n}$ in $\mathbf{\Psi_{RO}}$ is higher than $\mathbf{\Psi_m}$ or any items of $\mathbf{\Psi_n}$ in $\mathbf{\Psi_{cum}}$ is bigger than $\mathbf{\Psi_m}$. This procedure is also illustrated in Fig.3.

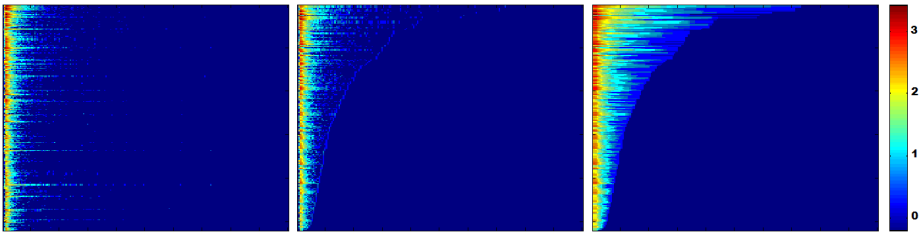

Fig. 3. From left to right: the matrix $\mathbf{\Psi} = [\mathbf{\Psi_1}; \mathbf{\Psi_2}; ...; \mathbf{\Psi_M}]$, where $\mathbf{\Psi_j}$ is defined in (9); $\mathbf{\Psi_{RO}}$ obtained by rearranging $\mathbf{\Psi}$ in its Right-Ordered form; the cumulative sum of each row of $\mathbf{\Psi_{RO}}$ in the reverse order

6 Experiments

Corel5K [8] is one of the most well known image annotation datasets. It contains 5000 images and 373 different tags in total. It is usually split into 4500 images for training and the remaining 500 for testing, resulting in 260 tags in both the training set and testing set. IAPR-TC12 [29] and a subset of ESP-game [4] are another two image annotation datasets which contain approximately 20000 images. More specifically, IAPR-TC12 is usually divided into a fixed number of 17665 images for training and the rest 1962 images for testing, while 291 tags exist both in the training set and testing set; ESP-game contains 18689 images for training and 2081 images for testing. The training set and testing set contain 268 different tags in common.

Much work has been done on these three datasets. In terms of accuracy, TagProp [4] is the best technique in the literature. The authors of TagProp have also

[2] `http://www.viplab.cs.nott.ac.uk/publications/Papers/ECCV12-supp.pdf`

released the features they used on these three dataset. We directly did experiments based on these features. There are 15 different kinds of features in total, including two kinds of global features: Gist features and color histograms. Local features include SIFT and hue descriptor which are extracted either densely or at the Harris-Laplacian interest points. For all these kinds of features, we use kernel PCA to reduce each of them to a fixed 100 dimensions. We use $k = \exp(-\gamma^{-1} \cdot d)$ to compute the kernel matrix, where γ is the mean of the distance matrix, and d is the distance between samples. Following [4], we use L_2 norm as the base metric for Gist, L_1 for global color histograms, and χ^2 for all the other features.

For all of these three datasets, we generate N_T random trees. Based on the *count* value, we retain K nearest neighbors for each test sample, then the tags are predicted using (5) based on these K nearest neighbors. As in other works, five tags are predicted for each image. Average precision, average recall and the number of tags whose recall is above zero ($N+$) are used to evaluate the performance.

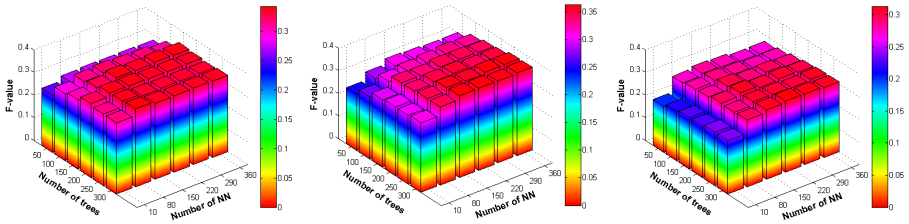

Fig. 4. The relation between the system performance F_value ($F\_value = 2 * Precision * Recall/(Precision + Recall)$), and the parameters of N_T (number of trees) and K (number of Nearest Neighbors used for prediction). From left to right: the results on Corel5K, results on IAPR-TC12 and results on ESP game.

The relation between the performance, and the number of trees N_T and the number of nearest neighbors K are shown in Fig.4. From there, we can see that about 200 trees will saturate the performance and we only need to retain about 100 nearest neighbors, and the three dataset exhibit a similar trend. This proves that our algorithm is robust to these parameters. Two stopping criterions are set in generating the random trees. The first one is that the number of images contained in a node should be larger than $thresh_1$, and a second criterion is that the depth of the random tree should not exceed $thresh_2$. We empirically set $thresh_1$ to be 8 across all the three datasets. $thresh_2$ is set to be 10 for Corel5K and 12 for the other two datasets.

6.1 Semantic Nearest Neighbor Search

A well known problem of traditional nearest neighbor search is the semantic gap issue where two images that are similar in the feature space bear little semantic

relations. Here we show that our random forest based semantic nearest neighbors can better overcome this issue. Fig.5 shows visual examples of several query images and their semantic nearest neighbors (SNN) returned by our method and the visual feature based nearest neighbors returned by the JEC method [5]. It can be seen that, whilst the visual appearances of the nearest neighbor images returned by JEC do resemble those of the query images, their semantics differ significantly. On the other hand, the SNN images returned by our method are not only visually similar to the query images but also share common semantics with them. More examples can be found in the supplementary material.

These examples demonstrate that our new concept of semantic nearest neighbors and semantic similarity measure can indeed be successfully used to perform nearest neighbor search and reduce the semantic gap of traditional nearest neighbor search. It is this capability to retrieve semantically similar nearest neighbors that has enabled our method to achieve good performances in image annotation.

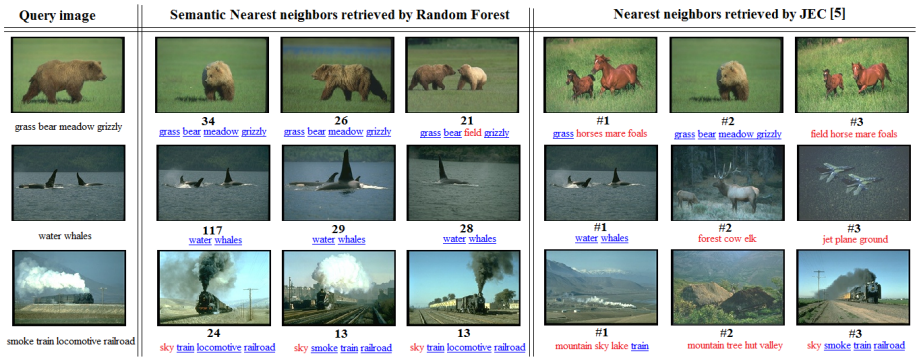

Fig. 5. Some examples of the semantic nearest neighbor images retrieved by our random forest method and by JEC method. The tags associated with each image are also shown beneath each image. The tags which are in accordance with the test image are colored in blue and underlined, while the false tags are colored in red. The numbers underneath the SNN images are the values of SSM.

6.2 Image Annotation Performances

Table.1 shows the average precision and recall performances of our new random forest based image annotation technique and comparisons with state of the art techniques. From Table.1, we can see that our methods outperform all previous methods only except TagProp [4]. However, as mentioned before, the success of TagProp relies on its sophisticated training procedure and per image per tag optimization, which hinders its extension to large scale datasets. On the contrary, our method can be easily extended to large scale datasets. Besides this, our method is as straightforward as a nearest neighbor method.

Table 1. Image Annotation Performances on Corel5K, IAPR-TC12 and ESP game. RF represents our Random Forest method. RF_*count* denote $f(c_i) = c_i$, RF_$count^2$ denote $f(c_i) = {c_i}^2$ and RF_*optimize* denote $f(c_i)$ is learned based on our optimization framework.

	Corel5K			**IAPR-TC12**			**ESP game**		
method	Prec	Recall	N+	Prec	Recall	N+	Prec	Recall	N+
MBRM [30]	0.24	0.25	122/260	0.24	0.23	223/291	0.18	0.19	209/268
JEC [5]	0.27	0.32	139/260	0.28	0.29	250/291	0.22	0.25	224/268
MSC [31]	0.25	0.32	136/260	-	-	-	-	-	-
HPM [32]	0.25	0.28	136/260	-	-	-	-	-	-
M-E Graph [33]	0.25	0.31	-	-	-	-	-	-	-
TagProp [4]	0.33	0.42	160/260	0.46	0.35	266/291	0.39	0.27	239/268
GS [34]	0.30	0.33	146/260	0.32	0.29	252/291	-	-	-
SML+RF [19]	0.36	0.33	135/260	0.27	0.30	266/291	-	-	-
RF_*count*	0.26	0.36	143/260	0.47	0.22	220/291	0.45	0.24	233/268
RF_$count^2$	0.29	0.41	165/260	0.45	0.31	253/291	0.34	0.27	239/268
RF_*optimize*	0.29	0.40	157/260	0.44	0.31	253/291	0.41	0.26	235/268

In this sense, JEC [5] is the most comparable to our method. However, our method shows a clear performance gain over JEC because JEC only relies on the nearest neighbors of visual features, whilst our method finds semantically similar nearest neighbors (also see Fig. 5).

6.3 Learning to Rank vs. Ad Hoc Annotation Functions

Comparing the results obtained from RF_*count*, RF_$count^2$ and RF_*optimize* in Table.1, we could see that RF_$count^2$ performs best on Corel5K and IAPR-TC12 but performs worst on ESP-game, while RF_*count* performs slightly better than RF_$count^2$ on ESP-game but worst on the other two datasets. This shows that these ad hoc annotation functions although sometimes can work well but as can be expected, lack consistency. In comparison, the systematic method RF_*optimize* performs consistently well across all the three datasets. The unpredictable performances of the ad hoc annotation functions across different datasets and the highly consistent good performances of the novel systematic learning to rank algorithm clearly highlighted the value and usefulness of our learning algorithm.

As can be seen the optimization objective of our learning to rank algorithm (11) treats each tag equally important. Another useful measurement of performances is to count the total number of correctly predicted tags. Table.2 lists the total number of correctly predicted tags for the two ad hoc annotation functions and the systematic learning to rank method. It is seen that the systematic method consistently predicted more correct tags.

Table 2. The number of correctly predicted tags on each dataset. RF_*optimize* consistently outperforms the ad-hoc functions.

	Corel5K	IAPR-TC12	ESP game
RF_*count*	993/1756	3957/11053	3778/9774
RF_$count^2$	1004/1756	4242/11053	3724/9774
RF_*optimize*	**1010/1756**	**4254/11053**	**3797/9774**

7 Concluding Remarks

In this paper, we have developed a novel random forest based framework for automatic annotation. Our new contributions include the use of tag information to guide the generation of the random trees, the introduction of the concept of semantic neighbors and a novel learning to rank framework for systematically learning image annotation models from the semantic neighbor sets. We have presented experimental results demonstrating that our new method is competitive to state of the art. We have recently shown that the idea of semantic neighbors is also beneficial for image retrieval. For more details, please refer to [35].

Acknowledgments. This work is partially supported by EPSRC grant EP/J020257/1 and China Scholarship Council - Nottingham Joint PhD scholarship.

References

1. Boiman, O., Shechtman, E., Irani, M.: In defense of Nearest-Neighbor based image classification. In: CVPR (June 2008)
2. Hays, J., Efros, A.A.: Scene completion using millions of photographs. In: SIGGRAPH, vol. 26 (July 2007)
3. Tighe, J., Lazebnik, S.: SuperParsing: Scalable Nonparametric Image Parsing with Superpixels. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 352–365. Springer, Heidelberg (2010)
4. Guillaumin, M., Mensink, T., Verbeek, J., Schmid, C.: TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation. In: ICCV (September 2009)
5. Makadia, A., Pavlovic, V., Kumar, S.: A New Baseline for Image Annotation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 316–329. Springer, Heidelberg (2008)
6. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV, vol. 2 (2003)
7. Wang, J., Kumar, S., Chang, S.F.: Semi-Supervised Hashing for Scalable Image Retrieval. In: CVPR (2010)
8. Duygulu, P., Barnard, K., de Freitas, J.F.G., Forsyth, D.: Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 97–112. Springer, Heidelberg (2002)
9. Carneiro, G., Vasconcelos, N.: Formulating Semantic Image Annotation as a Supervised Learning Problem. In: CVPR (2005)
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Symposium on Theory of Computing (1998)

11. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV (September 2009)
12. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: NIPS, vol. (1) (2008)
13. Jain, P., Kulis, B., Grauman, K.: Fast Image Search for Learned Metrics. In: CVPR (June 2008)
14. Jia, Y., Wang, J., Zeng, G., Zha, H., Hua, X.S.: Optimizing kd-trees for scalable visual descriptor indexing. In: CVPR (2010)
15. Kumar, N., Zhang, L., Nayar, S.: What Is a Good Nearest Neighbors Algorithm for Finding Similar Patches in Images? In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 364–378. Springer, Heidelberg (2008)
16. Muja, M., Lowe, D.G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: VISAPP (2009)
17. Uijlings, J., Smeulders, A., Scha, R.: Real-time Bag of Words, Approximately. In: CIVR (2009)
18. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS (2006)
19. Fukui, M., Kato, N., Qi, W.: Multi-Class Labeling Improved by Random Forest for Automatic Image Annotation. In: IAPR Conference on Machine Vision Applications, pp. 202–205 (2011)
20. Fu, H., Qiu, G., He, H.: Feature Combination beyond Basic Arithmetics. In: British Machine Vision Conference (BMVC). BMVA (2011)
21. Bosch, A., Zisserman, A., Munoz, X.: Image Classification using Random Forests and Ferns. In: ICCV (October 2007)
22. Yao, B., Khosla, A., Fei-Fei, L.: Combining Randomization and Discrimination for Fine-Grained Image Categorization. In: CVPR (2011)
23. Yu, G., Yuan, J., Liu, Z.: Unsupervised Random Forest Indexing for Fast Action Search. In: CVPR (2011)
24. Schölkopf, B., Smola, A., Müller, K.R.: Kernel Principal Component Analysis. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997)
25. Zhang, K., Tsang, I.W., Kwok, J.T.: Improved Nystrom Low-Rank Approximation and Error Analysis. In: ICML (2008)
26. Criminisi, A., Shotton, J., Konukoglu, E.: Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Foundations and Trends in Computer Graphics and Vision 7(2-3), 81–227 (2012)
27. Hu, J., Lam, K.M., Qiu, G.: A Hierarchical Algorithm for Image Multi-labeling. In: ICIP (2010)
28. Joachims, T.: Training Linear SVMs in Linear Time. In: ACM KDD (2006)
29. Escalante, H.J., Hernández, C.A., Gonzalez, J.A.: The segmented and annotated IAPR TC-12 benchmark. Computer Vision and Image Understanding (April 2010)
30. Feng, S., Manmatha, R., Lavrenko, V.: Multiple Bernoulli Relevance Models for Image and Video Annotation. In: CVPR (2004)
31. Wang, C., Yan, S., Zhang, L., Zhang, H.J.: Multi-label sparse coding for automatic image annotation. In: CVPR (June 2009)
32. Zhou, N., Cheung, W., Qiu, G., Xue, X.: A Hybrid Probabilistic Model for Unified Collaborative and Content-Based Image Tagging. IEEE TPAMI 33, 1281–1294 (2011)
33. Liu, D., Yan, S., Rui, Y., Zhang, H.J.: Unified Tag Analysis With Multi-Edge Graph. In: ACM MM (2010)
34. Zhang, S., Huang, J., Huang, Y., Yu, Y., Li, H., Metaxas, D.: Automatic Image Annotation Using Group Sparsity. In: CVPR (2010)
35. Fu, H., Qiu, G.: Fast Semantic Image Retrieval Based on Random Forest. In: ACM MM (2012)