Discriminative Tag Learning on YouTube Videos with Latent Sub-tags

Weilong Yang*
Simon Fraser University
Burnaby, BC, Canada
wya16@sfu.ca

George Toderici
Google Inc.
Mountain View, CA, USA
gtoderici@google.com

Abstract

We consider the problem of content-based automated tag learning. In particular, we address semantic variations (sub-tags) of the tag. Each video in the training set is assumed to be associated with a sub-tag label, and we treat this sub-tag label as latent information. A latent learning framework based on LogitBoost is proposed, which jointly considers both the tag label and the latent sub-tag label. The latent sub-tag information is exploited in our framework to assist the learning of our end goal, i.e.*, tag prediction. We use the cowatch information to initialize the learning process. In experiments, we show that the proposed method achieves significantly better results over baselines on a large-scale testing video set which contains about* 50 *million YouTube videos.*

1. Introduction

On the Internet, the term "tag" refers to keywords assigned to an article, image, or video. With the rapid development of social sharing websites (*i.e.*, Flickr, Picasa, and YouTube), the tags help organize, browse and search relevant items within these massive multimedia collections. In this paper, we are particularly interested in the problem of content-based automated tag learning/prediction. Given a video, an automated tag learning/prediction system would be able to predict the tags associated to the video based on its content. Such a system would ensure that videos are properly tagged, and therefore enforce uniformity in tagging. Uniformity is useful since users tend to not tag every possible aspect of the media, leading to a large number of undertagged objects. Having a uniform, automated tagging system would allow users to specify queries based on the known tag vocabulary and have an increased confidence that the results retrieved should be consistent, and more complete than otherwise possible, while allowing currently undertagged media to be considered for retrieval. Another application of the tag learning could be verifying that the users have typed a specific tag with the intent of annotating the video as opposed to simply spamming the index of the search engine.

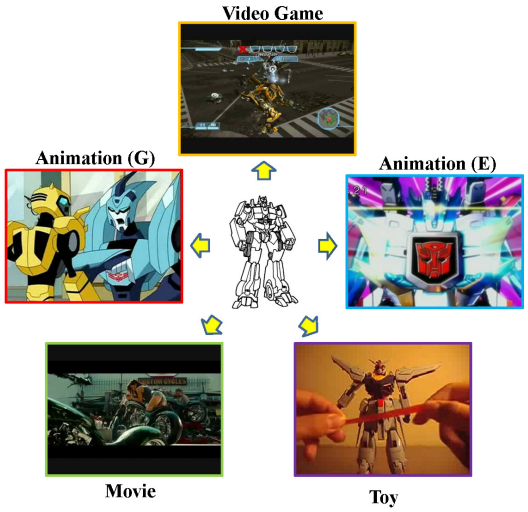


Figure 1. For the rather specific tag "transformers", the videos of this tag have large variations in video types and contexts. Those videos can be roughly grouped as video games, two different types of animations, toys, and movies. We use the notation "sub-tags" to refer to those semantic variants and treat them as latent information in our learning framework.

Our work is under a similar framework to [1]. We generate a set of tags by counting the unique user-supplied tags and n-grams in the title for each video on YouTube. For each given tag, a classifier model is trained over a positive training set containing the videos ($20K$-$100K$) whose user-provided metadata text contains the given tag. The negative training set is sampled randomly from videos which do not contain the tag in their metadata. In this way, both of the tag set and the training data are automatically generated from the online video corpora. In particular, since the tag set is generated from user-provided title/tags, the tags can better reflect the overall user tagging behaviors compared to a manually pre-defined tag set. In the tag set, we observe that some tags are too general (*e.g.*, "video", "music", "youtube"), and some others are too specific (*e.g.*, the name

*This work was done while Weilong Yang was an intern at Google, Inc.

of the video uploader). Those tags do not contain any useful information. Therefore we discard those tags which are either too frequent or too infrequent.

After examining the remaining tags, we observe that for most of tags, there is a large variation within the videos which have such a tag, though those tags are rather specific. Interestingly, for a particular tag, the variation of its associated videos is not trivial, and those videos can be clustered into a few groups which may have some semantic meanings. For example, as shown in Fig. 1, the videos with tag "transformers" consists of "video games", "animations", "toys" and "movies". Although all those videos contain the tag "transformers", the video types and contexts are different. Another example is the tag "bike", which is a simple object. As shown in Fig. 2, the "bike" videos may consist of videos about "mountain bike", "falling from bike", "pocket bike", and "motorbike". The variations among "bike" videos include scene variation (mountain *vs*. road), object variation (bicycle *vs*. motorbike *vs*. pocket bike), and event variation (falling from bike *vs*. riding a bike). In this paper, we use the notation of *sub-tag* to refer to the semantic variants of the given tag label. For example, the sub-tag set of "bike" is {mountain *bike*, falling from *bike*, pocket *bike*, motor*bike*}. In the context of video analysis, we suspect that off-the-shelf classifiers cannot handle these non-trivial variations. In this paper, we propose a novel framework that can jointly learn both tag label and its sub-tag labels in a unified framework. We believe introducing the sub-tag label to the learning process will help to capture more of the variations of the tag, thereby improving the quality of the final classifier.

Sub-tags are conceptually related to a hierarchy in which the parent tag would have a general meaning and the child sub-tag is more specific. However, unlike such hierarchies (*e.g*., ImageNet or WordNet), the sub-tag does not necessarily need to be related by a relationship which can be expressed in terms of rigid ontology. Besides, it is very difficult and costly to define (or name) the sub-tag set for all possible tag labels, especially for the YouTube videos. Naming sub-tags requires a comprehensive knowledge of video types, and user behaviors on YouTube. Moreover, in a dynamic video collection, more and more new tags may be invented and become popular, while the meanings of existing tags may vary as well.

The main contributions of this paper are two-fold. First, instead of explicitly defining the sub-tags, we propose to treat sub-tags as latent information and use it to assist the task of tag learning. Inspired by latent SVM [9], we propose a novel latent learning framework based on LogitBoost, which jointly considers both the latent sub-tag label and the tag label. In LogitBoost, we use decision stumps as the weak learner, which introduce the non-linearity into the model and also allow us to handle complex feature vectors that consist of different feature types. To deal with the noisy samples in our training data, we use a bootstrapping scheme which can be naturally combined into our learning framework. In each iteration, the model is only trained on a training subset that contains "trustworthy" positive samples. Secondly, likewise to other non-convex learning problems, a proper initialization of the latent sub-tag label is very crucial to our learning framework. We propose to use the cowatch-based clustering scheme for initialization. The similarity between two videos is measured by cowatch statistics. Conceptually, if two videos are watched one after the other in a short period of time (cowatched), they are usually related and likely to be similar. However, our learning framework is general, and it is not limited to this type of initialization scheme.

2. Related Work

In the computer vision literature, we can roughly categorize the "tag"-related work into two lines: (1) leveraging the tagged image/video content on the Internet to improve the performance of learning-based image or object recognition systems; (2) content-based image/video tag prediction.

With the help of image search engines and photo hosting websites, researchers built more and more computer vision datasets by downloading images or videos, *e.g*., ImageNet [6], and 80 Million Tiny Images [23]. To decrease the label noise, additional human annotations are employed in most datasets to prune out the noisy samples. It is interesting to note that in [23], a reasonable recognition performance is obtained despite the high labeling noise. Fan *et al*. [8] also show that more effective classifiers can be obtained after pruning out the noisy tags by an image clustering approach based on both visual and tagging similarities between images. Hwang and Grauman [11] assume that the prominence of an object in an image can be revealed by its order of mention in the tag list. They show that leveraging the information of tag ordering can improve the performance of object detection. In this paper, we also train our tag models on the videos collected from YouTube. For simplicity, we train a model for each tag and ignore the structure information among tags.

Recently, there has been an increased interest in automated image/video tagging. A variety of methods are proposed for image tagging, *e.g*., Kernel Canonical Correlation Analysis [21], hierarchical generative models [14], latent SVM [26, 12], *etc*. There is also a lot of research work about video tagging (*e.g*., [20]), and the proposed methods have been evaluated on the TRECVID dataset for the task of semantic indexing. A list of papers can be found in [16]. A few systems are built for the automated YouTube video tagging based on video content. Ulges *et al*. [24] predict the conceptual tags for the keyframes of the video by the combination of three different classifiers. Toderici *et al*. [22] and Aradhye *et al*. [1] learn the tags of a video using AdaBoost based on both video and audio features. In this paper, our work is based on [22, 1], but we are more interested in the

problem of semantic variations (sub-tags) in the YouTube videos that share the same tag. A latent learning framework is proposed to address this issue and we treat sub-tag labels as latent variables in our system.

3. Model Formulation

Given a tag label z and a training set $\mathcal{T}$ which consists of N training samples $\mathcal{T} = \{(V^n, y^n)\}_{n=1}^N$, our goal is to learn a scoring function for the tag label z: $f_z(V)$, which can return a confidence score of z being assigned to the video V. $y \in \{+1, -1\}$ takes value 1 if the video V has the tag label z, and -1 otherwise. In our model, each video is associated with a sub-tag label h, where $h \in \mathcal{H}_z$ and $\mathcal{H}_z$ is the set of sub-tags for the original tag z. We treat h as latent information since we do not have sub-tag labels during training. We assume the scoring function takes the following form:

$$f_z(V) = \max_{h \in \mathcal{H}_z} \Psi(V, h; \mathcal{F}_z);$$
$$\Psi(V, h; \mathcal{F}_z) = \sum_{b \in \mathcal{H}_z} \mathbb{1}_b(h) \cdot F_b(V), \quad (1)$$

where $\mathbb{1}_b(h)$ is an indicator that takes the value 1 if $h = b$, and 0 otherwise. $F_b(\cdot)$ is a LogitBoost [18] classifier learned from the training samples which share the same sub-tag label $h = b$. We denote $\mathcal{F}_z$ as a set of LogitBoost classifiers for the tag z, *i.e.*, $\mathcal{F}_z = \{F_b : b \in \mathcal{H}_z\}$. The details concerning training $\mathcal{F}_z$ will be discussed in Sec. 4.

Given $\mathcal{F}_z$ and a testing video V, we need to solve an inference problem of finding the best sub-tag label h^* as follows:

$$h^* = \arg\max_{h \in \mathcal{H}_z} \Psi(V, h; \mathcal{F}_z). \quad (2)$$

This inference problem can be easily solved by enumerating all the possible sub-tag labels $h \in \mathcal{H}_z$. In this paper, we are more interested in improving the performance of video classification at tag level, though we do obtain the sub-tag label h^* as a by-product for each testing video.

4. Learning

Different from other learning algorithms with structured output, we simply assume each tag label is independent of each other and we train a model for each tag label. The optimal $\mathcal{F}_z^*$ could be learned as follows:

$$\mathcal{F}_z^* = \arg\min_{\mathcal{F}_z} \sum_n^N l(y^n, \max_{h \in \mathcal{H}_z} \Psi(V^n, h; \mathcal{F}_z)), \quad (3)$$

where $l(\cdot)$ is the loss function. We choose to use the convex logistic loss $l(y, \Psi) = \log(1 + \exp(-2y\Psi))$ in this paper. Similar to the latent SVM [9], this problem is not convex but we could use an iterative algorithm to solve it by alternating the estimation of latent variable h and the optimization of $\mathcal{F}_z$. This iterative procedure can be summarized as follows:

1. Holding $\mathcal{F}_z$ fixed, compute the latent variable h^n for each training sample as follows:

$$h^n = \arg\max_{h \in \mathcal{H}_z} \Psi(V, h; \mathcal{F}_z); \quad (4)$$

2. Holding h^n fixed, optimize $\mathcal{F}_z$ by solving the following problem:

$$\mathcal{F}_z^* = \arg\min_{\mathcal{F}_z} \sum_n^N l(y^n, \Psi(V^n, h^n; \mathcal{F}_z)). \quad (5)$$

Eqn. (4) can be easily solved by enumerating all possible $h \in \mathcal{H}_z$. Note that in Eqn. (5), the latent variable h has been fixed to a single choice. $\mathcal{F}_z$ is a set of classifiers $\mathcal{F}_z = \{F_b : b \in \mathcal{H}_z\}$. The optimization problem in Eqn. (5) can be written as

$$L = \min_{\mathcal{F}_z} \sum_{b \in \mathcal{H}_z} \sum_{n:h^n=b} l(y^n, \Psi(V^n, h^n; \mathcal{F}_z))$$
$$= \sum_{b \in \mathcal{H}_z} \min_{F_b} \sum_{n:h^n=b} l(y^n, F_b(V^n)). \quad (6)$$

Then, Eqn. (5) can be solved by minimizing $L(F_b) = \sum_{n:h^n=b} l(y^n, F_b(V^n))\ \forall b \in \mathcal{H}_z$ independently. In practice, this can be simply achieved by a regular LogitBoost solver which learns a classifier $F_b(\cdot)$ over the training samples whose latent sub-tag label $h^n = b$, *i.e.*, $\{(V^n, y^n)\}_{n:h^n=b}$.

4.1. Implementation Details

LogitBoost: The implementation of LogitBoost used in the optimization of $\mathcal{F}_z$ uses decision stumps as the weak learner. This allows it to handle complex feature vectors that consist of different feature categories, and also introduce the non-linearity to our model. However, one disadvantage of this boosting algorithm is that it takes a long time to train. It is particularly undesirable if we use boosting in an iterative training algorithm. To address this issue, in the training of $F_b(\cdot)$, we first run a linear SVM to filter out the feature dimensions that are assigned small weights. In other words, we use linear SVM as a feature selection step to select a subset of discriminative features. Then LogitBoost will be only used on this subset of features. This trick can significantly speed up the training process. Similar tricks are also introduced in [10]. To further improve the efficiency, for each LogitBoost classifier, we use only 256 decision stumps as weak learners.

Reweighting of classifiers: Due to the above feature selection trick and the early stopping scheme (only 256 stumps), the learned LogitBoost classifiers may have not

converged. For the same testing example, it is possible that a well converged classifier will output a higher decision score than a classifier which has not converged, though both classifiers would classify this example as positive. This may cause problems when estimating the latent variable (Eqn. (4)), since we need to compare the decision scores from different LogitBoost classifiers. To address this issue, we train a linear SVM to calibrate the decision scores from different classifiers with respect to the hinge-loss $l(y^n, \mathbf{x}^n) = \max(0, 1 - y^n(\mathbf{w}^T\mathbf{x}^n))$, where $\mathbf{x}^n$ is the feature vector of V^n and $\mathbf{w}$ is the model parameters. $\mathbf{x}^n$ is represented as a sparse vector, and $|\mathbf{x}^n| = |\mathcal{H}_z|$. If the latent variable of V^n is equal to b, then the b-th element in $\mathbf{x}^n$ is equal to the decision score $F_b(V^n)$, and the rest of the elements are all 0. The linear SVM is trained over all pairs of (V^n, y^n) in the training set. After the training, the scoring function in Eqn. (4) can be re-written as $\Psi(V, h; \mathcal{F}_z) = \sum_{b\in\mathcal{H}_z} w_b \cdot \mathbb{1}_b(h) \cdot F_b(V)$, where w_b is the b-th element in $\mathbf{w}$. This calibration step is motivated by the mixture-model representation in [9]. The intuition is to re-weight each classifier based on its discriminative ability over the tag label y.

Bootstrapping: In this paper, the tag labels of the training samples are extracted from user-provided meta-data, such as the video title, and user-provided tags. These tag labels are often irrelevant to the video content. In particular, we observe that the "hard" positive examples during learning are usually the videos with noisy training labels. Including those noisy video examples will likely deteriorate the learning performance. Instead, we would like to "remove" those positive samples that have the lowest decision scores and thus more likely to be outliers. In our learning algorithm, we maintain a set of training samples $\mathcal{S}$ that is a subset of the entire training set $\mathcal{T}$, *i.e.*, $\mathcal{S} \subset \mathcal{T}$. In each iteration of our learning algorithm, instead of using all training samples, we optimize the model $\mathcal{F}_z$ only on the training subset $\mathcal{S}$ which contains the most "trustworthy" positive examples. Since we have a very large number of negative samples ($100K$) that is difficult to fit into the memory, in each iteration, we only include the "hard" negative samples into the training subset $\mathcal{S}$.

Learning procedure: First, we initialize the training set $\mathcal{S}$ and the latent variable h^n of each training sample in $\mathcal{S}$ (Sec. 5). Based on the latent sub-tag label, we can denote $\mathcal{S} = \cup_{b\in\mathcal{H}_z}\mathcal{S}_b$, where $\mathcal{S}_b$ contains the training samples that have the sub-tag label b. Then, we repeat the following steps for a fixed number of iterations.

1. Train a LogitBoost classifier $F_b(\cdot)$ for each sub-tag label $b \in \mathcal{H}_z$ over the training subset $\mathcal{S}_b$, then obtaining the model $\mathcal{F}_z = \{F_b : b \in \mathcal{H}_z\}$. We train a linear SVM to re-weight those classifiers;

2. Holding $\mathcal{F}_z$ fixed, compute the latent variable for every training sample in the entire training set $\mathcal{T}$ by Eqn. (4). The decision score of each sample can be computed as $s^n = \Psi(V^n, h^n; \mathcal{F}_z)$. Similarly, we denote $\mathcal{T} = \cup_{b\in\mathcal{H}_z}\mathcal{T}_b$, where $\mathcal{T}_b$ contains the training samples which have the sub-tag label b;

3. Update the training subset $\mathcal{S} = \cup_{b\in\mathcal{H}_z}\mathcal{S}_b$, which we can rewrite as $\mathcal{S}_b = \mathcal{S}_b^{pos} \cup \mathcal{S}_b^{neg}$, where superscripts denote positive and negative subsets respectively. We re-construct $\mathcal{S}_b^{pos}$ by using the top k samples in $\mathcal{T}_b^{pos}$ with the largest decision scores. Similarly, we re-construct S_b^{neg} by using the top k samples (hard negatives) from $\mathcal{T}_b^{neg}$. k is a predefined parameter;

In step 3, due to the fact that the dimension of our features is relatively large, in order to feed every training sample into the memory in step 1, we tune the parameter k so that the size $|S| \leq 20K$ and we incrementally increase the size of S during the iterative learning process.

The learning procedure described above is an algorithmic bootstrapping approach for generating a clean positive training set. We run this procedure for a fixed number of iterations in the absence of convergence guarantees, but find it effective experimentally. With the help of the training subset $\mathcal{S}$, our approach learns $\mathcal{F}_z$ on a training subset which contains the most "trustworthy" positive samples and thus it is likely more tolerant to label noise.

5. Initialization by Cowatch Features

For a non-convex problem, the initialization is usually very important. In this paper we treat the sub-tag label of a video as latent information. Intuitively, we would like to assign the same sub-tag label to the videos which are not just visually similar but also have a very strong semantic similarity. In this paper, we use video cowatch statistics [2] for the sub-tag initialization. Cowatch statistics are generated by measuring the occurrence frequency of two videos in the same viewing session. In other words, if two videos are watched one after the other by users, there will be a high cowatch connection between them. Note that it is common that users will watch similar videos in a short period of time (*i.e.*, a viewing session). Cowatch statistics is a reliable tool to measure the video similarity since it combines the votes from millions of users. We believe the feature obtained from cowatch statistics is also a helpful cue to disambiguate different sub-tags. For example, users are more likely to watch a "mountain bike" video followed by another "mountain bike" video, and less likely followed by a "pocket bike" video.

The procedure of using cowatch statistics for initialization is summarized as follows:

Step 1: From the positive training set of the tag label z, we randomly sample N videos with the tag z. In our experiments, we set $N = 3000$. We generate a cowatch video list C_l^n for each sampled video. We remove from C_l^n the videos that do not contain the tag label z. Then, we combine those

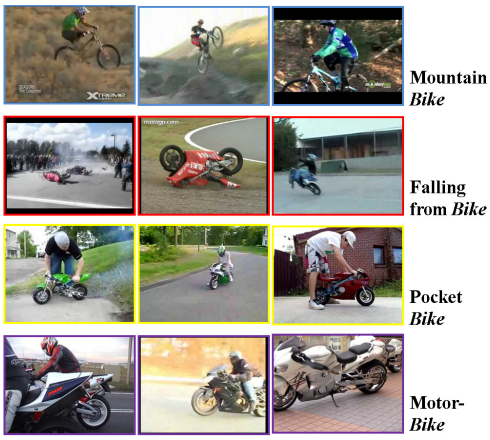


Figure 2. The sub-tag initialization results for the "bike" tag. In total, we generate four sub-tags: "mountain bike", "falling from bike", "pocket bike", and "motorbike". Note that our algorithm only cluster the candidate videos into four clusters and we manually assign a meaningful word label to each cluster.

cowatch lists as a video set $C_s = \cup_{n=1}^{N} C_l^n$. Each video can be represented as a sparse vector v^n, and $|v^n| = |C_s|$. The value of i-th element in v^n is 1 if the corresponding i-th video of C_s also exists in C_l^n, and 0 otherwise.

Step 2: The cowatch lists of similar videos are likely to overlap. To generate the initial sub-tag label set, we simply cluster the set $\{v^n\}_{n=1}^{N}$ into K clusters using *k-means*. The distance between sparse feature vectors is computed by using the L_1 distance. Clusters with too few samples will be discarded and we obtain $K' \leq K$ clusters in the end. Then, we merge the cowatch video list C_l^n of each sampled video to its corresponding cluster. The video set C_s is generated by combining the cowatch lists of N randomly sampled videos. It is possible that for some videos in C_s, they may appear in only one of the N co-watch lists. Those videos are more likely to be irrelevant to the tag label z, so we remove those videos from the K' clusters.

Step 3: We can build the sub-tag set for tag z as $\mathcal{H}_z = \{0, ..., K' - 1\}$. For the purposes of initialization, we use the cluster label of each video as its initial sub-tag label h^n. Note that the generation of this sub-tag set is an unsupervised process. Our algorithm cannot automatically assign a semantic meaningful label to each sub-tag. Then, the videos from cluster b will form the initial training subset $\mathcal{S}_b^{pos}$, and we randomly sample a large number of negative samples to form the $\mathcal{S}_b^{neg}$, for all $b \in \mathcal{H}_z$.

The example initialization results for the tag label "bike" are depicted in Fig. 2. Each initialization cluster has a very unique semantic meaning. Note that in this paper, the cowatch feature is only used for initialization purpose. It is also possible to consider the cowatch feature as a part of the video feature in our learning algorithm. However, the reliability of cowatch information depends on view counts of the video. In practice, most of testing videos would be the newly-uploaded videos, which have no viewing history.

6. Features

In order to tackle the problem of tag/concept/action learning many researchers [20, 7, 24, 3] have decided to use solely visual features. In the spirit of [22], we decide to use auditory features in addition to visual features. We believe that audio gives an important cue in video analysis, especially for tag prediction. The audio features contribute to detect concepts that are very difficult to define visually, *e.g.*, cat versus dog. Cats and dogs not only appear in similar contexts, but they are also very hard to disambiguate by only visual features. However, a dog's bark sound may help us distinguish between a dog video and a cat video.

We extract a variety of features from each video, which allow to capture various aspects of the video. The features can be categorized into three groups: frame features, motion features, and auditory features. For each type of feature, we use the standard bag-of-word representation. Each feature will be represented as a histogram by vector quantizing the feature descriptors. The histogram is normalized so that the sum of all the bin values is 1. The final feature vector of each video is the concatenation of the histograms from each feature. Its dimension is fixed for videos with different length, and the maximum number of non-zero dimensions for a video is 12439.

Frame features: This group of features consists of the histograms of oriented gradients (HOG) feature, color histogram, texton, and a face counter. For the HOG feature, at each frame pixel location, we extract a 1800-dimensional feature descriptor, which is the concatenation of HOG [5] in a 10×10 surrounding window. The raw descriptors are then collected into a bag-of-words representation by quantizing them using a randomized decision tree similar to [19]. In addition, we also compute a Hue-Saturation color histogram [13] and a Texton histogram [17] for every frame. Lastly, we run a face detector [25] over every frame and count the number of faces in each frame. This simple face counter provides an easy way to discriminate between videos containing human faces and those which do not. Note that all those frame-based features are represented as histograms, which are further pooled over the entire video using mean-pooling.

Motion features: In order to compute motion features we employ the cuboid interest point detector [7]. We extract spatio-temporal volumes around all the detected interest points. From each cuboid we extract two types of descriptors: (1) We concatenate the normalized pixel values to a vector and apply PCA to reduce the dimensionality to 256. (2) We first split each slice of the cuboid into 2×2 cells. Then, we concatenate all HOG descriptors of cells in the cuboid to a vector. Similarly, PCA is applied to reduce the dimensionality to 256. Both descriptors are further quan-

Tag	Sub-tags
bike	mountain *bike*, falling from *bike*, pocket *bike*, motor*bike*
boat	building a *boat*, jet *boat*, *boat* accident
card	*Card* Captor Sakura (animation), *card* collection, making a greeting *card*, *card* trick
dog	*dog*1, *dog*2, *dog*3, *dog*4
explosion	bomb *explosion*, *explosion*2, building implosion
flower	*paper flower (howto)*, *flower (plant)*, *flower*3
helicopter	remote controlled (RC) *helicopter*1, RC *helicopter*2, *helicopter* crash, military *helicopter*
horse	*horse* jumping, *horse* reining, *horse*3
kitchen	*kitchen* remodeling, *kitchen*2, cooking show
mountain	*mountain* creek, *mountain* biking, *mountain* driving, *mountain* climbing
protest	riot police, *protest* in Iran, *protest*3, *protest* in Thailand
robot	industry robot, *robot*2, Super *Robot* Wars (video game)
running	free *running*, *running* tips, *running* back (football skill)
stadium	soccer *stadium*, football *stadium*, baseball *stadium*
transformers	video game, animation1, animation2, movie, toy

Table 1. The summary of the sub-tag labels for the 15 tags used in our experiments. Note that our algorithm cannot assign the semantic meaning label to each sub-tag. For the purpose of illustration, we manually assign a word label to each sub-tag by summarizing the video clusters obtained from cowatch initialization. For some sub-tags which are difficult to assign a meaningful word label, we simply use the cluster number to represent them.

tized using their corresponding codebooks.

Auditory features: We choose two widely-used audio features in addition to visual features: mel-frequency cepstral coefficients (MFCC) [4] and stabilized auditory images (SAI) [15].

7. Experiments

We evaluate our method on a large-scale video dataset which consists of about 50 million YouTube videos. We only use a very small portion for training, and remaining videos are used for testing. For ease of evaluation, we arbitrarily selected 15 tags: *"bike"*, *"boat"*, *"card"*, *"dog"*, *"explosion"*, *"flower"*, *"helicopter"*, *"horse"*, *"kitchen"*, *"mountain"*, *"protest"*, *"robot"*, *"running"*, *"stadium"*, *"transformers"*. In terms of categories, this tag set contains "animals", "objects", "actions", "scenes", and "events". For each tag, we consider a set of $20K$-$100K$ videos which contains the given tag as the potential positive training set, and we randomly select around $100K$ videos as the negative training set. The rest of videos are used for testing. For each tag, we create a sub-tag set following the method described in Sec. 5. For illustration purposes, we summarize the sub-tag set of each tag in Table 1 and manually assign a semantic word to each sub-tag.

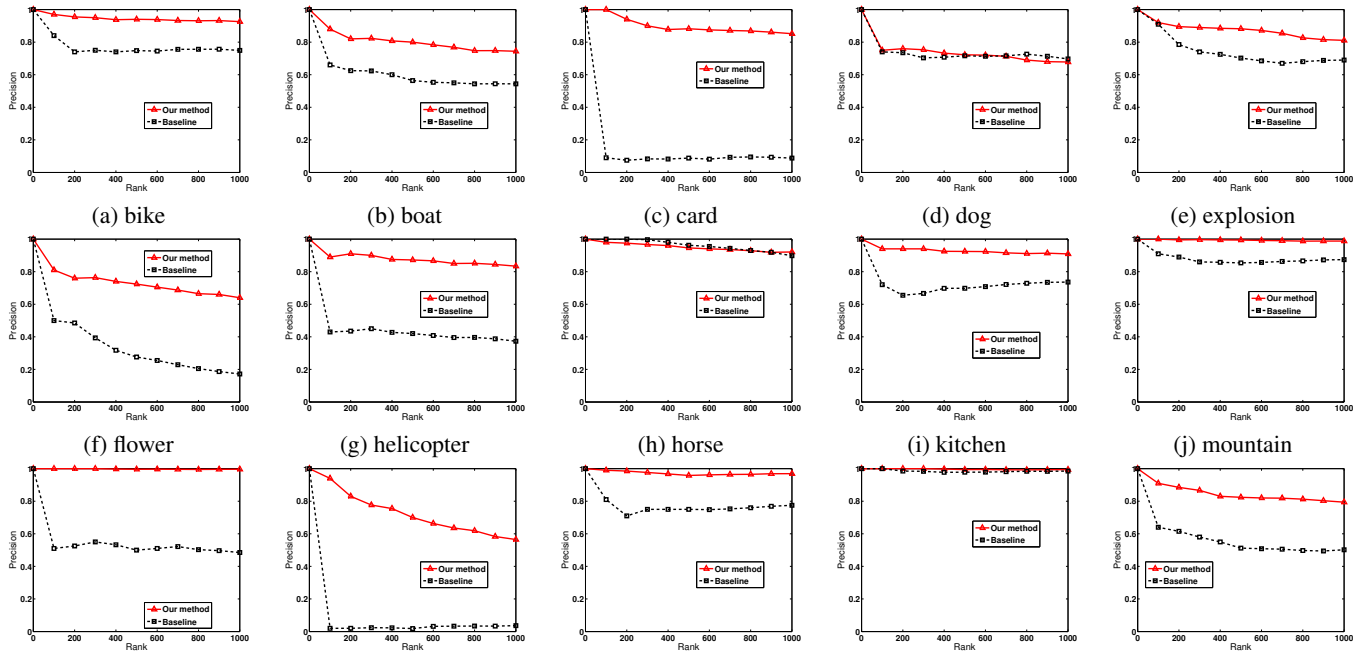
7.1. Evaluation Measures

For every given tag, we train a classifier model, which we apply on each video in the testing set (none of the videos in this set were used during training). A decision score is computed for each video using Eqn. (1). If we had the ground-truth tag label of the 50 million testing videos, we could use the decision scores to compute the ROC or precision-recall curves. However, it is tremendously costly to accurately annotate that many videos, even when using low-cost online crowdsourced marketplaces (*e.g.*, Amazon's Mechanical Turk). An alternative way is to randomly sample a relatively small number of testing videos then manually annotate those videos and only use them in the evaluation. We argue that this scheme is not fair, because the randomly sampled set would either be too small to fairly represent all the video categories on YouTube, or it would be too large to be practical.

In this paper, we choose the precision at K (precision@K) measure, which has been widely adopted in information retrieval. In practice, video retrieval and ranking are also important applications of automated video tagging system. For each tag, we first apply the trained model onto whole testing set (50M). We rank the videos in the testing set by their decision scores. Then, we only annotate the videos with top K decision scores. The precision at K is computed as precision@K $= |\{\text{relevant videos}\}|/K$. In this way, *for each tag*, the evaluation only requires the annotations of K videos. We choose $K = 1000$, giving us $15,000$ videos to annotate. Compared to 50 million videos, this number is negligible but it is far more acceptable in terms of annotation cost. This measurement is particularly suitable for the situation in which the testing set is dynamic, as in the case in which new videos are added to the collection over time. For each update of the testing set, we only need to annotate the new videos that appear in the top-K rank list.

7.2. Results

We compare our method to the video tag learning approach described in [22]. In order to make the comparison fair, we use the exactly the same features (Sec. 6) for both the baseline and our method. The average precision@1000 of the baseline is 57.36%, and our method is 84.14%. We observed an improvement of 46% on the average precision@1000. Fig. 3 shows the precision at K curves of both our method and the baseline. As we can see, our method significantly outperforms the baseline on 12 tags, and we achieve similar results on the tags "dog", "horse", and "stadium". Both our method and the baseline achieve very good performance on the tags of "stadium" and "horse". Most of "dog" videos uploaded to YouTube are usually shot from consumer level hand-held cameras. These videos have very limited variation, and thus our method performs similarly to the baseline on this tag. After manually examining the cowatch initialization results, as shown in Table 1, the sub-tags of "dog" do not have any semantic meaning. This observation demonstrates a limitation of our approach: our approach can barely improve the perfor-



(a) bike (b) boat (c) card (d) dog (e) explosion

(f) flower (g) helicopter (h) horse (i) kitchen (j) mountain

(k) protest (l) robot (m) running (n) stadium (o) transformers

Figure 3. Precision at K curves for both baseline and our method on the 50 million YouTube video dataset. We incremental increase K from 100 to 1000.

mance for the unambiguous tags . This is what we expect since for the unambiguous tags, the latent sub-tag information has little semantic meaning and thus its contribution is very limited.

As shown in Table 1, most of sub-tag sets generated from cowatch initialization are semantically meaningful. One interesting question is whether the iterative process of our algorithm really contribute to the performance improvement. To answer this question, we compare our approach with a method that is only trained on the initial training subset obtained by the cowatch initialization. This method essentially only runs the first step of our learning procedure (Sec. 4.1) once. Due to the cost of annotation, we only run this experiment on two tags: "transformers" and "bike". For comparison purposes, we compute precision@1000 as shown in Table 2. We can see that our method outperforms the method using only initial training subset. This is reasonable because in the initial training subset, the latent sub-tags are generated from the cowatch information and are not tied with our end goal of tag learning. Interestingly, on the "transformers", the method using only initialization is worse than the baseline. We believe it is due to the high complexity of the "transformers" videos, and the initialization of the sub-tag set cannot properly capture the variance of the "transformers" tag.

Given a tag label z and a testing video, besides a decision score of tag z being assigned to the testing video, we can also infer the sub-tag label for the testing video by Eqn. (2). We visualize the testing videos with top scores from each sub-tag in Figs. 4,5. Two interesting observations can be made. In Fig. 4 ("transformers"), some videos under sub-tag h_0 (video game) are classified as h_4 (movie). In the cowatch initialization of tag "bike", most videos of sub-tag h_2 are about "pocket bike". However, as shown in Fig. 5, the testing videos under sub-tag h_2 are mostly "motorbike". Those videos seem to be "misclassified" with respect to sub-tag label, but they are all related videos to the tag label "bike", which is exactly what we expect. As we pointed out in Sec. 3, the latent sub-tag label is only a by-product of our model. Our model is only optimized for tag-level classification, so we do not aim to obtain good sub-tag level classification results. Therefore, although some video do not have an "accurate" sub-tag label, they are assigned the correct tag labels. Note that the "misclassified" and "accurate" are determined by comparing the testing videos with initialization results. Sub-tag information is latent so we cannot measure the performance of sub-tag level classification.

Tag	Baseline	Initialization only	Our approach
transformers	50.2%	33.1%	**79.4%**
bike	74.8%	87.4%	**92.5%**

Table 2. Precision@1000 for tags "transformers" and "bike". We compare our method to the baseline, and a method that runs the first step of our learning procedure once.

8. Conclusion

We have studied the problem of semantic variations in the videos which share the same tag. We have named those semantic variants as sub-tags, which were treated as latent

(a) h_0 (video game)

(b) h_1 (animation1) (c) h_2 (animation2)

(d) h_3 (toy) (e) h_4 (movie)

Figure 4. Visualizations of the sub-tag labels of the testing videos for tag "transformers". Please refer to supplementary materials for more examples.

(a) h_0 (mountain bike) (b) h_1 (falling from bike)

(c) h_2 (pocket bike) (d) h_3 (motorbike)

Figure 5. Visualizations of the sub-tag labels of the testing videos for tag "bike". Please refer to supplementary materials for more examples.

variables and used to assist the task of tag learning. A general latent learning framework was proposed to jointly model the tag label and its related latent sub-tags. We have presented a clustering approach based on cowatch information to initialize the latent sub-tag labels in our learning framework. By running experiments on the testing set which consists of about 50 million YouTube videos, we have demonstrated that our approach significantly outperformed the baselines, with an improvement of over 46% on the average precision@1000.

Acknowledgements: We would like to thank Mehmet Emre Sargin, Ullas Gargi, Dennis Strelow, Hrishikesh Aradhye, Jay Yagnik and Greg Mori for their help on this paper.

References

[1] H. Aradhye, G. Toderici, and J. Yagnik. Video2text: Learning to annotate video content. In *ICDM workshop*, 2009.

[2] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for YouTube: taking random walks through the view graph. In *WWW*, 2008.

[3] D. Borth, A. Ulges, and T. Breuel. Relevance filtering meets active learning: improving web-based concept detectors. In *ACM MIR*, 2010.

[4] G. Chechik, E. Ie, M. Rehn, S. Bengio, and D. Lyon. Large-scale content-based audio retrieval from text queries. In *ACM MIR*, 2008.

[5] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. In *CVPR*, 2005.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[7] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV-VS*, 2005.

[8] J. Fan, Y. Shen, N. Zhou, and Y. Gao. Harvesting large-scale weakly-tagged image databases from the web. In *CVPR*, 2010.

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE PAMI*, 2009.

[10] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 2003.

[11] S. Hwang and K. Grauman. Reading between the lines: Object localization using implicit cues from image tags. In *CVPR*, 2010.

[12] L. Jie and F. Orabona. Learning from Candidate Labeling. In *NIPS*, 2010.

[13] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 2001.

[14] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR*, 2009.

[15] R. Lyon, M. Rehn, S. Bengio, T. Walters, and G. Chechik. Sound retrieval and ranking using sparse auditory representations. *Neural computation*, 2010.

[16] Trecvid notebook papers. http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html, 2010.

[17] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[18] R. Schapire. The boosting approach to machine learning: An overview. In *MSRI workshop*, 2002.

[19] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[20] C. Snoek, K. van de Sande, O. de Rooij, B. Huurnink, J. van Gemert, J. Uijlings, J. He, X. Li, I. Everts, V. Nedovic, et al. The MediaMill TRECVID 2008 semantic video search engine. In *TRECVID Workshop*, 2008.

[21] R. Socher and L. Fei Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*, 2010.

[22] G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, and J. Yagnik. Finding meaning on youtube: Tag recommendation and category discovery. In *CVPR*, 2010.

[23] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *PAMI*, 2008.

[24] A. Ulges, M. Koch, D. Borth, and T. M. Breuel. Tubetagger - youtube-based concept detection. In *ICDM Workshop*, 2009.

[25] P. Viola and M. Jones. Robust real-time object detection. In *Workshop on SCTV*, 2001.

[26] Y. Wang and G. Mori. A discriminative latent model of image region and object tag correspondence. In *NIPS*, 2010.