IMPROVING VIDEO CONCEPT DETECTION THROUGH LABEL SPACE PARTITIONING

Usman Niaz and Bernard Merialdo

Eurecom, France
niaz@eurecom.fr, merialdo@eurecom.fr

ABSTRACT

We present an approach to video concept detection by building binary trees partitioning the label space, using visual and semantic similarity for multi-label datasets. The technique overcomes sparse annotations problem by increasing the number of positive examples per concept with the number of classifiers per concept, though sub-optimal, augmented too. We draw similarities between the proposed tree generation approach and Error Correcting Output Codes (ECOC) for multi-label classification and build ranked lists of video shots using weighted decoding or weighted tree traversal. We build a set of different trees based on the presented criterion each partitioning the label space in its own specific way. Inspired by the work in [1] we amass information from ensemble of trees to build the final ranked list, but using a different criterion. The classification resulting in *ensemble error correction* is complementary to One-vs-All classification and increases concept detection performance significantly on the TRECVID 2010 and 2013 datasets.

Index Terms— Error correcting codes, multi-label classification, video concept detection

1. INTRODUCTION

General purpose video concept detection is a difficult task which is approached conventionally by building classifiers for each category using all the available annotations, or the One-vs-All (OVA) classification. The number of possible categories is limitless which leads to ever increasing video and image datasets. In addition trying to accommodate all the variations an object can be presented in a category makes the problem even harder. Nonetheless research is progressing and computer's vision is improving every year [2, 3].

Getting sane annotations for semantic concepts is expensive and some categories lack enough positive annotations, while on the other side there is no limit to selecting negatives which may lead to classification imbalance. However the video datasets are multi-label in nature which allows us to borrow annotations for associated categories. An image can contain many objects and a video usually depicts more than one item. The examples of "Car" are highly likely to contain the concept "Road" and can thus complement each other's training means.

We use this intrinsic multi-labeling with a twofold objective; to increase the training resources per label and to increase the number of classifiers per label which will in the end be combined to try to maximize the performance on the training set. Pursuing the work in [1] we divide the labels into partitions. A binary tree, or label tree [1], represents the partitioned label space reflecting the relationships between concepts based on the selected measure of similarity. The concepts that are very *similar* are kept together till deep into the tree, until a time comes eventually where they are placed in the opposite sides of the partition. This way each label embodies a unique combination of partitions.

Each node of the binary tree divides the receiving label set into two disjoint partitions. We divide the concepts or labels into partitions iteratively, until all the labels are divided. Then we merge all the annotations on either side of the partition and train a single label binary classifier for that partition. In the end we trace back classification score for each label based on appropriate combinations of the scores of the partitions to which that label belongs. The label partitioning is viewed in terms of Error Correcting Output Codes (ECOC) framework where a set of simple sub-optimal classifiers can achieve the performance of a good complex one [4, 5, 6, 7]. Then an effective weighting strategy is adopted while decoding to emphasize more on important partitions or dichotomies, generating list of scores for each label. Finally we build groups or ensembles of trees [1] by varying the similarity criterion and fuse the information from trees to get the final classification score. We have performed experiments on the TRECVID 2010 the TRECVID 2013 dataset and we show that the proposed approach is complementary to single label classification by showing significant improvement on the TRECVID Semantic INdexing (SIN) task. We also shed some light on the importance of the number of labels to start the partitioning with and argue that more labels induce a better partitioning and thus improved classification results.

The next section talks about some recent works in the domains of label partitioning and the use of ECOC for classification followed by the label partitioning criterion in the section 3. Experiments and results are presented in the section 4. Finally section 5 discusses future directions with conclusions.

Thanks to ANR, France for funding.

2. STATE OF THE ART

Wang and Forsyth [1] partition the label space of 1000 categories to generate binary trees for multi-class image categorization. They outperform random partitioning of label space with ensembles (forests) of label trees where for each category the highest score is selected among the trees of the forest as the classification result. In one ensemble the error from the previous tree is used to generate the next one. We rely directly on out similarity criterion to generate ensembles of trees. For combination we use weighted fusion to generate the final ranked list.

ECOC has recently attracted interest in the field of multi-label classification after conquering multi-class classification. Escalera et al. [7] discuss ternary ECOC for multi-class categorization problems which allows to increase the number of dichotomies thus encompassing better the data and label distribution. In ternary representation each label can either be a part of the classifier (dichotomy) with +1 or -1 bit or it can be omitted from a dichotomy with a 0 bit. Armano et al. [8] use ternary ECOCs to cater multi-label Text Categorization. The dichotomies are built in a straight forward manner by using all the present combinations in the training label set. Ferg and Lin [5] present an ECOC framework for multi-label classification problems and demonstrate its success on the Random K-labelset (RAKEL) algorithm. Contrarily [6] did not find ECOC to be suitable for RAKEL but found good classification results with Bode-Chauduri-Hocquenghem (BCH) ECOC. However they did not test ternary ECOC which is what is used in this paper. [9] improve over RAKEL by combining labels into groups using visual similarity and training group classifiers . We use a similar similarity measure combined with a label based distance to construct trees.

A close lying research area is the use of attributes for object detection that describe some visual or descriptive properties of the objects and group all the objects with those properties into a single classifier [10]. An attribute can be present or absent in an object and every object is a unique combination (binary code) of attribute classifiers. Whereas hashing assigns a unique binary code to each individual example [11] with two examples of same category having very similar codes.

3. PROPOSED APPROACH

We partition the label or concept space which is defined by projecting the labels into the space represented by the distance between them. This distance between labels can be calculated through different means. We use a *visual distance* and a *semantic distance* between labels to find this distance. The two measures are used simultaneously to split nodes and construct the label tree. The root node splits the whole label set into two partitions. Each node then splits the receiving partition into two subsets until each leaf contains only one label.

The partitioning criterion should be designed in a way to increase the learnability of the tree. **Learnability** as described by [1] (and the references there in) is the ability of the tree to learn from a particular partitioning of the labels and generalize on a test set. So a tree with poor or counter intuitive partitioning will have low learnability and thus will perform poor on the test set. As an example a tree partitioning the labels {*Car*, *Road*, *Anchorperson*, *Newsstudio*} into {*Car*, *Road*}, {*Anchorperson*, *Newsstudio*} is more learnable than a tree splitting the initial set into {*Car*, *Newsstudio*}, {*Anchorperson*, *Road*}. Our distance measures help build learnable partitionings as shown in the results.

Input L: a set of labels
if $|L| = 2$ **then**
  $L_l \leftarrow l_1$
  $L_r \leftarrow l_2$
else
  $\{i, j\} \leftarrow \underset{i,j}{\operatorname{argmax}}\, diss(i, j), i, j \in L$
  $L_l \leftarrow i \qquad L \leftarrow L \backslash L_l$
  $L_r \leftarrow j \qquad L \leftarrow L \backslash L_r$
  repeat
    $L_l \leftarrow L_l \cup \operatorname{argmin} diss(i, L_l), i \in L$
    $L_r \leftarrow L_r \cup \operatorname{argmin} diss(i, L_r), i \in L$
    $L \leftarrow L \backslash L_l$
    $L \leftarrow L \backslash L_r$
  until $L = \emptyset$
end if

Fig. 1. Node Splitting Algorithm

3.1. Tree Construction

To find the distance between two labels l_i and l_j we gather all the examples belonging to the two labels into two sets. Let v_i be the average feature vector for the examples having of l_i. Then $d_v(i, j)$ is the Euclidean distance between l_i and l_j which is referred to here as the visual distance. For each label we have a set of positive annotations A_i and since we are dealing with multi-label data the semantic similarity between two labels is given by:

$$sim(i, j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}$$

which is the ratio of common annotations between labels i and j. Here common annotations mean all the examples which are positive for the two cncepts in question. Since $sim(i, j)$ measures similarity between labels $d_s(i, j) = -sim(i, j)$ measures the semantic (label) dissimilarity. The total dissimilarity between labels is the weighted sum of the two measures

$$diss(i, j) = \lambda * d_v(i, j) + (1 - \lambda) * d_s(i, j) \qquad (1)$$

Figure 1 shows the algorithm for splitting one node of the tree which receive a certain set of labels as input. The algorithm simply selects the two farthest labels as seeds and then builds two clusters (left and right) around those seeds. At each iteration the label closest to either cluster is assigned to that cluster until no more label is left to be assigned. Note that the new dissimilarity is calculated between the label and the cluster center but not the initial seed. Two partitions are generated each of which is further passed through the same procedure to complete the tree. Since it is a binary tree we call the two partitions *left* and *right*.

For training each node is considered one binary classifier since it partitions the labelset into two sets. All the examples belonging to the labels in the left partition are treated as positive examples and the examples belonging to labels in the right partition are considered negatives. So the positive and negative examples for each classifier are actually only the positive examples of concepts on either side. We borrow notation from [1] to build the two sets of examples as follows:

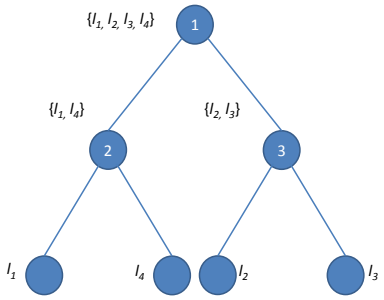
$$\begin{aligned} S^+ &= \{x_i : y_i \in L_l\} \\ S^- &= \{x_i : y_i \in L_r\} \end{aligned} \tag{2}$$

S^+ and S^- are used to train the binary SVM classifiers. Since we are dealing with mutli-label data as opposed to multi-class data, in case of contention the example is assigned to the partition containing the minority class. I.e. if an example belongs to more than one labels, and the two labels are divided by a node, the example will belong to the node with fewer total examples. This makes sense as in case of a multi-label example belonging to a general class, *Vehicle* and a specific class, *Car* the example should be for *Car* since it differentiates *Car* from the other *Vehicle*s. For the other case of simple majority vs. minority it is better not to reduce the already limited number of examples for the minority class. But this kind of contention normally occurs near the leaves since labels close to each other are kept in the same partitions until it is necessary to separate them to tell them apart.

3.2. Ternary Codes and Error Correction

The tree construction bears close resemblance to the popular muli-class classification approach Error Correcting Output Codes (ECOC) [6, 8]. To look at the benefits of the repetitive codes for classification we first explain the resemblance. ECOC builds a set of sub-optimal binary classifiers for each label where final prediction is obtained by simple pooling of the participating classifiers for that label. Each classifier epitomizes a dichotomy which partitions the label space just like a node of the tree built in the previous subsection.

The technique presented here is essentially ternary ECOC where a label can be either part of a dichotomy (-1,+1) or not (0). Ternary ECOCs are natural extension of ECOC methods for multi-label classification [6, 8]. The nodes or dichotomizers are built incrementally optimizing criterion in equation 1



	n_1	n_2	n_3
l_1	1	1	0
l_2	-1	0	1
l_3	-1	0	-1
l_4	1	-1	0

Fig. 2. A binary tree which partitions the label-set and the corresponding M-matrix with 3 dichotomies. Each node of the tree represents a column of the M-matrix

to divide most profiting class labels. Each class label encodes a unique code and the (hamming) distance between two close classes is low satisfying conditions in [1, 9]. This code is also called the codeword for that category and the codewords for all the categories make up the rows of the ECOC M matrix, where $M \in \{-1, 0, 1\}^{|L|,N}$ [7, 8]. $|L|$ is the number of categories or labels and N is the length of each codeword which is the number of classifiers or the number of nodes of the label tree. Figure 2 depicts a multi-label tree and the corresponding M matrix [4].

Since we are generating ranked lists for test frames, every classifier $n \in N$ gives us a score $s_n(+1|f)$ for the test frame f. The score for each label is calculated as [8]

$$s(l|f) = \sum_{n=1}^{N} s_n(+1|f) M(l, n)$$

Since the dichotomies turn from general to more discriminative as the depth of the tree increases we have found that more weight assigned to the specific dichotomies deeper in the tree results in better performance. So instead of the uniform weighting used in [8] we have used the following

$$s(l|f) = \sum_{n=1}^{N} \frac{n}{|M(l, .)|} s_n(+1|f) M(l, n)$$

with $s(l|f)$ finally normalized for a tree. $|M(l, .)|$ is the number of classifiers for label l. Each leaf of the label tree is a single label so $s(l|f)$ can also be regarded as the score at leaf corresponding to label l.

The first and foremost advantage of any technique augmenting labels is the increase in the number of examples per category which affects significantly performance of the categories with fewer positive examples. In ECOC this advantage is not limited to just the increase in training resources per category but also increases the error correcting capabilities of the system due to repeated classifiers [4, 8, 5]. Even if some dichotomies partition badly the label space others are expected

to correct the mistakes due to repetition and provide reliable final results. This advantage is further strengthened when we use ensemble of trees for a single problem as described in the next subsection.

For good error correction we argue that the label set should be dense in relation with the number of examples. Since we have established that related labels add examples for each other, a dense multi-label dataset induces a tree which in which partitions are more learnable. Thus if the label set is very sparse, unrelated labels are always forced to group together in partitions, and we will have fewer training examples per dichotomy and a poor per label classification in the end. We support these claims with experimentation in section 4.

3.3. Ensemble of Trees, Forest

The label partitioning tree uses pre-computed information and is quick to build. We take the liberty to build multiple trees by varying our similarity criterion. This is done by changing the value of λ in equation 1. The trees can be combined by averaging the scores of the corresponding leaves, chosing the highest score [1] or finding a weighted combination of the corresponding outputs from the trees. For label (leaf) l the weighted combination from two trees t_1 and t_2 is learned as follows on the validation data:

$$s(l|f) = w_l^{t_1} * s^{t_1}(l|f) + w_l^{t_2} * s^{t_2}(l|f)$$

We have built ensembles of upto 6 trees using linear weighted fusion of trees. For each ensemble the parameter λ in equation 1 is varied from 0 to 1 uniformly. E.g. for an ensemble of 3 trees λ is 0, 0.5 and 1 for each of them.

4. RESULTS AND EXPERIMENTS

4.1. Datasets and Setup

We have performed experiments on the TRECVID 2010 (TV2010) and TRECVID 2013 (TV2013) [2, 3] dataset.

For the TV2010 containing 400 hours of 11644 internet videos with 119,685 keyframes for development and 146,788 test keyframes we generate ranked lists for 50 concepts from the TRECVID 2011 Light Semantic Indexing task [2]. The visual features used are 128 dimensional SIFT features [12] which are densely extracted [13] and are used to build a 500 word visual dictionary using k-means. For classification we have used linear SVM to learn from a suitable feature map (homogeneous kernel map) built by the histogram intersection kernel [14].

For the TV2013 dataset, which engulfs the TRECVID 2010, 2011 and 2012 datasets, consisting of 800 and 600 hours of videos for training and test respectively we extract dense SIFT features and generate a dictionary of 1000 visual words. Considering the huge amount of data the classifier used is a linear SVM trained on the homogeneous kernel map built on the input features where Pegasos training [15] is used to minimize the training time. The development is done using the list of 60 concepts while 38 concepts were evaluated by NIST in the year 2013, for which results are presented.

Methods	TV 2010	TV 2013
OVA	5.27	6.91
GS [9]	5.37	-
LFM [1]	4.52	3.45
LFA	5.28	4.72
LFR	5.14	4.52
LFO	**6.29**	5.13
GS-OVA [9]	6.04	-
LFO-OVA	**7.05**	**8.38**

Table 1. MAP scores for TRECVID 2010 and 2013

4.2. Results

The video concept detection performance is judged by calculating Average Precision (AP) on the first 2000 shots returned for each label (concept) on the test datasets.

4.2.1. Overall Results

Table 1 shows the Mean AP (MAP) scores of the proposed label forest (LF) technique and compares them to the one-vs-all (OVA) classification and groupsvm (GS) [9]. For the LF the scores from individual trees are maximized (LFM) for each concept as in [1], averaged (LFA) and fused with optimal weights (LFO). Trees are also build by partitioning the labels randomly dropping the visual and semantic closeness criterion, which are then fused by finding optimal weights (LFR). Furthermore LFO and GS are fused with OVA using linear weighted fusion which further improves performance. For table 1 all the forests used contain 6 trees and for GS the best results are shown containing 100 groups for 50 concepts from [9]. Results on GS for TV2013 were not available.

4.2.2. Examples per Classifier

Using binary dichotomies to train classifiers the number of labeled examples are considerably reduced when compared to the one-vs-all approach since the negative examples provided with the dataset are discarded and only positive examples are used for training. Table 2 shows the average number of examples per classifier for the two datasets, rounded to the closest hundred, acquired from the development sets using all concepts (50 for TV2010 and 60 for TV2013). This removal of negatives comes with a cost which is visible in the performance of LFO for TV2013 where each classifier has almost 10 times few examples.

However if we look in terms of number positive examples per label there is a considerable increase for both datasets.

	OVA (pos+neg)	OVA (pos)	One-Tree
TV2010	45,000	1,600	9,500
TV2013	78,000	1,700	9,100

Table 2. Average number of examples per classifier

Since all the examples on either side of the dichotomy for the label tree are actually positive examples the numbers in the third column of the table 2 are positive examples for our approach. The increase is manifold from the original number of positives annotations per category on average. For the almost quadrupled TV2013 the number of labels is lower with respect to the dataset size so the label partitioning does not entirely capture the relationships between concepts as some counter intuitive partitions may be formed.

4.2.3. Reducing the set of labels

To further make our point we have performed an experiment with rather smaller label sets on the similar datasets. That is to say we make label partitioning trees for fewer labels. To achieve that we divided the set of 50 concepts randomly into 5 sets of 10 concepts each for TV2010 and then repeated the similar experiments. Similarly for TV2013 the 60 concepts set was divided into 6 sets of 10 and trees were generated separately for each of the 6 sets of labels. For both the datasets for every set of 10 concepts we generated 6 trees making one ensemble, finally making 5 ensembles for TV2010 and 6 ensembles for TV2013. Final classification scores were calculated separately for each ensemble and in the end we have AP scores for 50 and (38 evaluated out of 60) concepts for TV2010 and TV2013. Table 3 shows the MAP for the two datasets with divided label sets, which is considerably less than the MAP acquired using the full set of labels for a 6-tree ensemble, table 1. The number of total examples on average per classifier is also reduced, as understandable, to 6,400 for TV 2010 and 5,700 for the TV2013 dataset. Thus the performance of the label forests approach is critical to the number of labels and increases with the increase in the label-set size.

4.2.4. Concepts with Few Positives

To see the effect of detection performance on the concepts that have very few positive annotations we select 10 concepts from each dataset with fewest positive examples. We compare performances of LFO with 6 trees and fusion of LFO and OVA and plot the performances on the log linear graphs in figures 3 and 4.

	LFO	LFO-OVA
TV2010	5.21	6.00
TV2013	3.12	7.01

Table 3. MAP scores for all concepts using subsets of concepts for tree generation

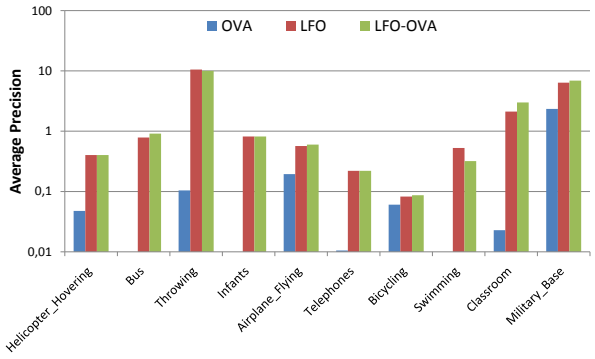


Fig. 3. AP scores for 10 concepts with fewest positive annotations in TV2010

Figure 3 shows AP scores for 10 concepts from TV2010 dataset with an average 82 positive examples per concept in the development set. All the concepts benefit from the addition of examples, even those that had an AP of 0 with OVA. Fusing OVA and LFO further improves performance except for "Throwing" and "Swimming" with a negligible drop.

For TV2013 the number of positive examples for the 10 concepts with fewest examples is 610 on average. Results are shown in figure 4 and here LFO is not as effective but the fusion tries to recover some of the loss. However LFO does bring more shots in the first 2000 for the concepts "Telephones" and "Bus" for which there was none or only a few previously with OVA.

4.2.5. Groups of Trees

Figure 5 shows performance of various groups of trees for both the datasets. We build forests of 2 to 6 trees using the proposed method which is compared to random label partitioning for the same number of trees. Both methods are then

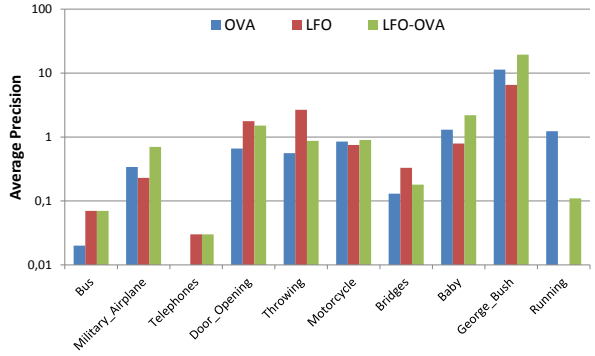


Fig. 4. AP scores for 10 concepts with fewest positive annotations in TV2013

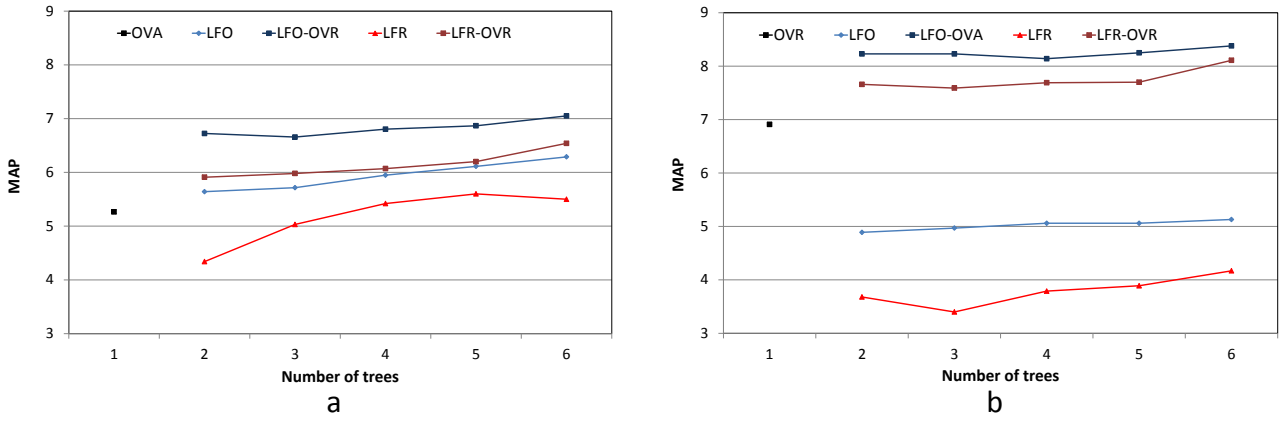


Fig. 5. Performance (MAP) comparison of proposed label partitioning to random partitioning and single label classification, for various groups of trees. (a) TRECVID 2010 (50 concepts) and (b) TREVID 2013 (38 concepts) datasets

fused with OVA for each group of trees. Since the tree generation using the algorithm in figure 1 generates balanced tree, the random tree generation is also done a balanced way for a fair comparison. The depth of every tree is similar and so is the number of SVMs. The proposed method always outperforms random partitioning for single classification and also for fusion with OVA. The black box in both the figures represents MAP score for OVA. For TV2010 every forest performs better than OVA and when fused the performance increases by around 30% for each forest. LF approaches do not perform as good as OVA for TV2013 due to reasons listed earlier but again the fusion results in around 20% increase for every forest. The improvements over the baseline are verified in their significance by randomization testing [16] for the proposed approach, for both the datasets.

5. CONCLUSIONS

The proposed label partitioning method uses effective measures of similarities generating meaningful partitions to increase learnability of the tree. The technique is complementary to single label classification as it improves performance significantly with as little as only two trees in the forest when the two are fused. The error correcting capabilities of the tree increase as more labels are available.

During iterative tree generation to create a forest a part of validation data can be used to weight similarities between labels. Furthermore like [4] more nodes can be added to the already generated tree to further classify certain confusing labels. Since a label tree essentially results in unique codes for each category eventually other methods could be used to generate such unique codes.

6. REFERENCES

[1] Y. Wang and D. Forsyth, "Large multi-class image categorization with ensembles of label trees.," in *ICME*, 2013.

[2] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *MIR*, 2006.

[3] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. Smeaton, and G. Quenot, "Trecvid 2013 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *Proceedings of TRECVID 2013*. NIST, USA.

[4] O. Pujol, S. Escalera, and P. Radeva, "An incremental node embedding technique for error correcting output codes," *Pattern Recognition*, vol. 41, 2008.

[5] C. Ferng and H. Lin, "Multi-label classification with error-correcting codes.," in *ACML*, 2011, vol. 20 of *JMLR*.

[6] T. Kajdanowicz and P. Kazienko, "Multi-label classification using error correcting output codes.," *Applied Mathematics and Computer Science*, vol. 22, 2012.

[7] S. Escalera, O. Pujol, and P. Radeva, "Separability of ternary codes for sparse designs of error-correcting output codes," *Pattern Recognition Letters*, vol. 30, no. 3, 2009.

[8] G. Armano, C. Chira, and N. Hatami, "Error-correcting output codes for multi-label text categorization.," in *IIR*, 2012, CEUR.

[9] U. Niaz and B. Merialdo, "Leveraging from group classification for video concept detection," in *CBMI*, 2013.

[10] M. Rastegari, A. Farhadi, and D. Forsyth, "Attribute discovery via predictable discriminative binary codes," in *ECCV*, 2012.

[11] J. Heo, Y. Lee, J. He, SF. Chang, and S. Yoon, "Spherical hashing," in *CVPR*, 2012.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, November 2004.

[13] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "Labelme: A database and web-based tool for image annotation," *Int. J. Comput. Vision*, vol. 77, May 2008.

[14] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," 2008.

[15] S. Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," 2007, ICML.

[16] M. Smucker, J. Allan, and B. Carterette, "A comparison of statistical significance tests for information retrieval evaluation," in *ACM-IKM*, 2007.