MSRA-USTC-SJTU AT TRECVID 2007: HIGH-LEVEL FEATURE EXTRACTION AND SEARCH

Tao Mei [1], *Xian-Sheng Hua* [1], *Wei Lai* [1], *Linjun Yang* [1]
Zheng-Jun Zha [2*], *Yuan Liu* [2*], *Zhiwei Gu* [2*], *Guo-Jun Qi* [2*], *Meng Wang* [2], *Jinhui Tang* [2], *Xun Yuan* [2]
Zheng Lu [3*], *Jingjing Liu* [4*]

[1] Microsoft Research Asia
[2] University of Science and Technology of China
[3] Shanghai Jiao Tong University
[4] Massachusetts Institute of Technology
{xshua, tmei, weilai, linjuny}@microsoft.com

ABSTRACT

This paper describes the MSRA-USTC-SJTU experiments for TRECVID 2007. We performed the experiments in high-level feature extraction and automatic search tasks. For high-level feature extraction, we investigated the benefit of unlabeled data by semi-supervised learning, and the multi-layer (ML) multi-instance (MI) relation embedded in video by MLMI kernel, as well as the correlations between concepts by correlative multi-label learning. For automatic search, we fuse text, visual example, and concept-based models while using temporal consistency and face information for re-ranking and result refinement.

Index Terms— support vector machines, semi-supervised learning, manifold ranking, multi-layer multi-instance kernel, linear neighborhood propagation, temporally consistent Gaussian random field, optimal multi-graph learning, correlative multi-label annotation, video annotation, video search.

1. INTRODUCTION

This is the second time we participate in TRECVID. We took part in two tasks and submitted 6 runs for high-level feature extraction and 6 runs for automatic search task.

In high-level feature extraction, we focused on applying a variety of new learning-based methods for video annotation. The first category is semi-supervised learning by leveraging both labeled and unlabeled data, including manifold ranking (MR) [1], optimal multi-graph semi-supervised learning (OMGSSL) [2], temporally consistent Gaussian random field (TCGRF) [3], and linear neighborhood propagation (LNP) [4]. The second category is using multi-layer multi-instance kernel (MLMIK) [5] which considers multi-layer structure and multi-instance relation embedded in video in a single formulation. And the third category is correlative multi-label learning (CML) [6] which simultaneously models the concepts and correlations among concepts. Support Vector Machine (SVM) [7] was adopted as the baseline. As a result, there are 7 classifiers employed in high-level extraction. For each classifier, we trained three different models on three types of low-level visual modalities (i.e., shot, frame, and region level), as well as different data splitting manners. In total, we trained 73 different models. Then, based on different fusion strategies (i.e., early and late fusion; linear, max, and average fusion) of these models, we had 15 methods in total. Finally, we fused the models and methods according to different fusion strategies and submitted 6 runs. The pipeline of high-level feature extraction is shown in Figure 1, including data preparation, modalities (i.e., low-level features), classifiers, models, and methods. And the runs we submitted are

- **A_MSRA_USTC_SJTU_HLF_1:** linearly weighted fusion of all the 15 methods and the 23 models.
- **A_MSRA_USTC_SJTU_HLF_2:** linearly weighted fusion of all the 15 methods.
- **A_MSRA_USTC_SJTU_HLF_3:** linearly weighted fusion of SVM related runs where the weights are obtained based on 2007 fusion set (defined in Figure 1), including (1) max fusion of SVM-LF and SVM-TVS-LF, (2) max fusion of SVM-TV-EF-CV3, SVM -TVS-EF-CV3, and SVM-TVS-LF-CV3, (3) MLMIK-EF, MLMIK-EF-C+C-, and MLMIK-LF, and (4) CML-LF.
- **A_MSRA_USTC_SJTU_HLF_4:** linearly weighted fusion of the top 5 methods for each concept based on the evaluations on 2007 selection set (defined in Figure 1).
- **A_MSRA_USTC_SJTU_HLF_5:** linearly weighted fusion of MLMIK-EF, MLMIK-EF-C+C-, and MLMIK-

[*] This work was performed when this author was visiting at Microsoft Research Asia as a research intern.

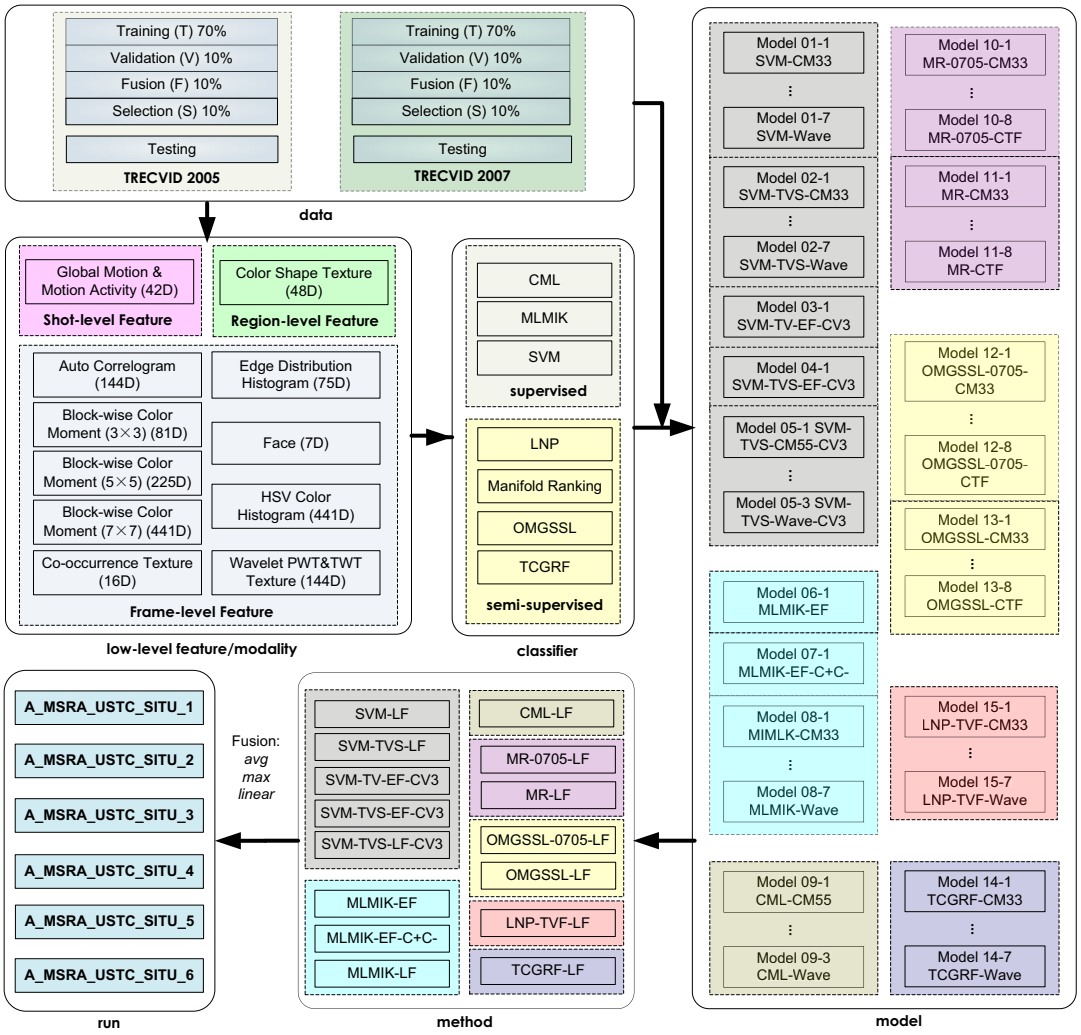

Fig. 1. The MSRA_USTC_SJTU TRECVID 2007 high-level feature extraction pipeline. CV3 - 3-fold cross validation, EF - early fusion, LF - late fusion, 2007 - 2007 data set, 2005 - 2005 data set, T - training, V - validation, F - fusion, S - selection, C - only tuning parameter C in SVM, C^+ and C^- – tuning both of two parameters for SVM.

Table 1. The performances of six runs for feature extraction

RUN ID	MAP
A_MSRA_USTC_SJTU_HLF_1	0.0960
A_MSRA_USTC_SJTU_HLF_2	0.0926
A_MSRA_USTC_SJTU_HLF_3	0.0903
A_MSRA_USTC_SJTU_HLF_4	0.0909
A_MSRA_USTC_SJTU_HLF_5	0.0646
A_MSRA_USTC_SJTU_HLF_6	0.0707

Table 2. The performances of six runs for automatic search

RUN ID	MAP
A_MSRA_USTC_SJTU_SEARCH_1	0.0873
A_MSRA_USTC_SJTU_SEARCH_2	0.0639
A_MSRA_USTC_SJTU_SEARCH_3	0.0610
A_MSRA_USTC_SJTU_SEARCH_4	0.0659
A_MSRA_USTC_SJTU_SEARCH_5	0.0375
A_MSRA_USTC_SJTU_SEARCH_6	0.0291

LF where the weights are obtained based on 2007 selection set.

- **A_MSRA_USTC_SJTU_HLF_6:** max fusion of SVM-LF and SVM-TVS-LF.

The corresponding performances of high-level feature extraction are listed in Table 1, in which we found that A_MSRA_USTC_SJTU_HLF_1 achieved the best MAP among the submitted 6 runs.

In automatic search, we focused on text and visual baseline, query by example (QBE), fusion, reranking, and result refinement methods. The pipeline of automatic search is shown

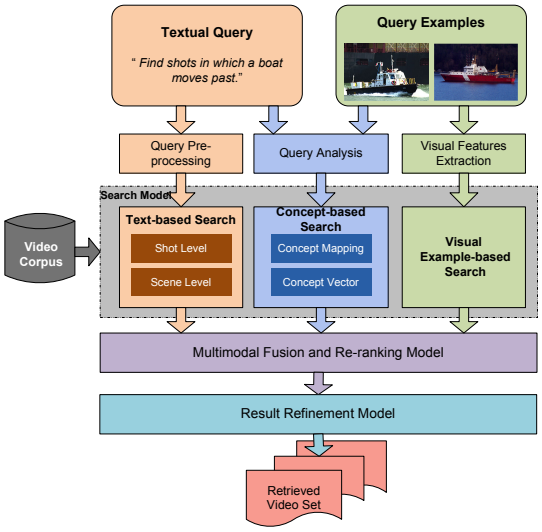

Fig. 2. MSRA_USTC_SJTU automatic search pipeline.

in Figure 2. Finally, we submitted the following 6 runs:

- **A_MSRA_USTC_SJTU_SEARCH_1:** linearly weighted fusion of SEARCH_4 and concept mapping search.
- **A_MSRA_USTC_SJTU_SEARCH_2:** average fusion of SEARCH_4 and concept mapping search.
- **A_MSRA_USTC_SJTU_SEARCH_3:** average fusion of SEARCH_1, SEARCH_4, SEARCH_5 and query by example (QBE), as described below.
- **A_MSRA_USTC_SJTU_SEARCH_4:** linear fusion of text-based search with reranking and result refinement at shot and scene level.
- **A_MSRA_USTC_SJTU_SEARCH_5:** visual baseline without using text information, i.e., average fusion of concept mapping and concept vector based search.
- **A_MSRA_USTC_SJTU_SEARCH_6:** text baseline.

The corresponding performances of automatic search are listed in Table 2, in which we found that A_MSRA_USTC_SJTU_SEARCH_1 achieved the best MAP among the submitted 6 runs.

2. HIGH-LEVEL FEATURE EXTRACTION

2.1. Modalities (Low-level features)

We extracted three types of low-level feature (referred to as "modality" in this notepaper) for each shot or key-frame, including shot, frame, and region levels. As a result, there are 42 dimensional shot-level feature, 1181 dimensional frame-level feature, and 48 dimensional region-level feature in total. Table 3 lists the detailed information of these modalities.

2.2. Classifiers

2.2.1. Support Vector Machine (SVM)

Implementation: SVM [7] was adopted as the baseline. Here we adopt two fusion strategies, i.e., late fusion and early fusion. In late fusion, SVM was trained in each of the seven frame-level features described in Table 3 except for "co-occurrence texture" and "face" modalities. Then we fused these seven models by linear weights. In early fusion, all the nine frame-level features are concatenated into an 1181-dimentional feature vector to train SVM models. As a result, we applied the following five methods: (1) SVM-LF, (2) SVM-TVS-LF, (3) SVM-TV-EF-CV3, (4) SVM-TVS-EF-CV3, and (5) SVM-TVS-LF-CV3.

We separated both TRECVID 2005 and 2007 development set into four partitions including "Training," "Validation," "Fusion," and "Selection" set, respectively. The detailed information of data splitting, models and methods is described in Figure 3. We used RBF kernels in SVM, which have two primary parameters: C (the cost parameter in soft-margin SVMs) and γ (the width of the RBF function). The effectiveness of SVM classifiers is highly subject to the selection of model parameters. To address the unbalance problem, we set different cost parameters for positive and negative samples, respectively. Therefore, we considered three model parameters: C^+ (the cost parameter for the positive examples), C^- (the cost parameter for the negative examples), and γ. In our method, we assign the ratio $\frac{C^+}{C^-} = \frac{N^+}{N^-}$, where N^+ and N^- are the numbers of positive and negative training examples, respectively. Based on the "Validation" set shown in Figure 1, we selected the best choice of these parameters. For more details we used SVM for feature extraction, please refer to [9].

2.2.2. Correlative Multi-Label Video Annotation (CML)

Formulation: We have proposed a kernel-based multi-label classification algorithm for video annotation [6]. This algorithm simultaneously classifies concepts and models correlation between them in a single step. Let $\boldsymbol{x} = (x_1, x_2, \cdots, x_D)^T \in \mathcal{X}$ denote the input pattern representing feature vectors extracted from video clips; Let $\boldsymbol{y} \in \mathcal{Y} = \{+1, -1\}^K$ denote the K dimensional concept label vector of an example, where each entry $y_i \in \{+1, -1\}$ of $\boldsymbol{y}$ indicates the membership of this example in the ith concept. $\mathcal{X}$ and $\mathcal{Y}$ represent the input feature space and label space of the data set, respectively. CML aims at learning a linear discriminative function $F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \boldsymbol{w}, \theta(\boldsymbol{x}, \boldsymbol{y}) \rangle$, where $\theta(\boldsymbol{x}, \boldsymbol{y})$ is a vector function mapping from $\mathcal{X} \times \mathcal{Y}$ to a new feature vector and $\boldsymbol{w}$ is the linear combination weight vector. With such a discriminative function, for an input pattern $\boldsymbol{x}$, the label vector $\boldsymbol{y}^*$ can be predicted by maximizing over the argument $\boldsymbol{y}$ as $\boldsymbol{y}^* = \max_{\boldsymbol{y} \in \mathcal{Y}} F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$. The constructed feature $\theta(\boldsymbol{x}, \boldsymbol{y})$ is a high-dimensional feature vector, whose elements can be par-

Table 3. Low-level feature (modalities)

Level	Feature	Dim	Description
Shot	Global motion	42	Extract frames per one second for each shot, for each frame, extract motion features [5]
Frame	Auto-correlogram	144	36-bincolor histogram based on 4 different distance k, i.e., k = 1,3,5,7.
	ColorMoment3-by-3	81	Based on 3 by 3 division of images in Lab space
	ColorMoment5-by-5	225	Based on 5 by 5 division of images in Lab space
	ColorMoment7-by-7	441	Based on 7 by 7 division of images in Lab space
	Co-occurrence Texture	16	The same feature as in [1]
	Edge Distribution Histogram	75	The same feature as in [1]
	Face	7	Face number, face area ratio, the position of the largest face
	HSV Color Histogram	64	The same feature as in [1]
	Wavelet PWTTWT Texture	128	The same feature as in [1]
Region	Color correlogram, color moment, shape descriptor	48	Based on image segmentation by JSEG [8]

titioned into two types as follows.

Type I The elements for *individual* concept modeling:

$$\theta^{l}_{d,p}(\boldsymbol{x},\boldsymbol{y}) = x_d \cdot \delta\left[\!\left[y_p = l\right]\!\right], \quad l \in \{+1,-1\}, 1 \le d \le D, 1 \le p \le K \tag{1}$$

where $\delta\left[\!\left[y_p = l\right]\!\right]$ is an indicator; D and K are the dimensions of low level feature vector space $\mathcal{X}$ and the number of the concepts respectively. These entries of $\theta(\boldsymbol{x},\boldsymbol{y})$ serve to model the connection between the low level feature $\boldsymbol{x}$ and the labels $y_k (1 \le k \le K)$ of the concepts.

Type II The elements for concept correlations:

$$\theta^{m,n}_{p,q}(\boldsymbol{x},\boldsymbol{y}) = \delta\left[\!\left[y_p = m\right]\!\right] \cdot \delta\left[\!\left[y_q = n\right]\!\right], \quad m,n \in \{+1,-1\}, 1 \le p < q \le K \tag{2}$$

where the superscripts m and n are the binary labels (positive and negative label), and subscripts p and q are the concept indices. These elements serve to capture all the possible pairs of concepts and labels.

We concatenate the above two types of elements to form the feature vector $\theta(\boldsymbol{x},\boldsymbol{y})$. The kernel function (i.e. the dot product) between the two vectors, $\theta(\boldsymbol{x},\boldsymbol{y})$ and $\theta(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$, can be represented in a very compact form as

$$\langle\theta(\boldsymbol{x},\boldsymbol{y}),\theta(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})\rangle = \langle\boldsymbol{x},\tilde{\boldsymbol{x}}\rangle \sum_{1\le k\le K} \delta\left[\!\left[y_k = \tilde{y}_k\right]\!\right] + \sum_{1\le p<q\le K} \delta\left[\!\left[y_p = \tilde{y}_p\right]\!\right] \delta\left[\!\left[y_q = \tilde{y}_q\right]\!\right] \tag{3}$$

Using the feature vector we constructed above and its kernel representation in Eq. (3), the learning procedure trains a classification model by solving an optimization problem. Please refer to [6] for more details.

Implementation: We used all the key-frames in "Training" subset as training samples and validated the RBF kernel radium σ and trading-off parameter λ based on the "Validation" subset. For more information we used CML, please refer to Figure 3.

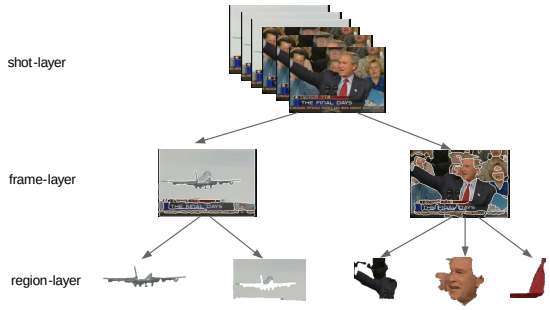

Fig. 4. MLMI setting for video annotation. A video shot is represented as three layers (i.e., shot, frame, and region layers), where each layer have different granularities of visual descriptors. The "bag-instance" correspondence is embedded in the three layers.

2.2.3. Multi-Layer Multi-Instance Kernel (MLMIK)

Formulation: In MLMIK, we consider video is essentially a structured media with multi-layer representation. For example, a video can be represented by a hierarchical structure including, from large to small, shot, key-frame, and region. Moreover, it fits the typical Multi-Instance setting in which the "bag-instance" correspondence is embedded among contiguous layers. We call such multi-layer structure and the "bag-instance" relation embedded in the structure as Multi-Layer Multi-Instance (MLMI) setting [5]. We formulate concept detection as an MLMI learning problem, in which a rooted tree with MLMI nature embedded is devised to represent a video shot. Furthermore, by fusing the information from different layers, we construct a novel MLMI Kernel (MLMIK) to measure the similarities between the instances in the same and different layers. In MLMI setting, each sample can be viewed as an L-layer rooted tree. Each node in the tree denotes a pattern of certain granularity. As shown in Figure 4, it is a three layer tree consisting of, from root to leaf, shot, key-frame, and region, where the granularity of descriptors of each layer in-

creases with more detailed information of the samples. Given an L-layer rooted tree $\mathbf{T}$, denote $N = \{n_i\}_{i=1}^{|N|}$ be the node set in $\mathbf{T}$, $|N|$ is the number of nodes. Let $\mathbf{S}$ be the sub-tree set, we denote $\boldsymbol{s}_i$ be the sub-trees whose parent is n_i, $\boldsymbol{s}_i = \{s|s \in \mathbf{S} \wedge parent(s) = n_i\} \in pow(\mathbf{S})$, $pow(\mathbf{S})$ refers to the power set of $\mathbf{S}$; besides, we have the bijection mapping $n_i \rightarrow \boldsymbol{s}_i$. For each node $n_i \in N$, define "node pattern" of n_i to be all the information associated with n_i, which is composed of three elements: layer info ℓ_i, descriptor f_i and sub-trees rooted at n_i (or $\boldsymbol{s}_i$), denoted in the triplet form $\hat{n_i} =< \ell_i, f_i, \boldsymbol{s}_i >$. Hence, $\mathbf{T}$ is expanded to the node pattern set $\hat{N} = \{\hat{n_i}\}_{i=1}^{|N|}$. We construct the kernel of trees by the expanded node pattern set according to convolution kernel as follows,

$$k_{MLMI}(T, T') = \sum_{\hat{n} \in \hat{N}, \hat{n'} \in \hat{N'}} k_{\hat{N}}(\hat{n}, \hat{n'}) \tag{4}$$

in which $k_{\hat{N}}$ is a kernel on the triplet space, since $\hat{n}$ is composed of three elements, we construct the kernel by tensor product operation ($K_1 \otimes K_2((x,u),(y,v)) = K_1(x,y) \times K_2(u,v)$).

$$k_{\hat{N}}(\hat{n}, \hat{n}') = k_\delta(\ell_n, \ell'_n) \times k_f(f_n, f'_n) \times k_{st}(\boldsymbol{s}_n, \boldsymbol{s}'_n) \tag{5}$$

where $k_\delta(x,y) = \delta_{x,y}$ is the matching kernel, k_f is a kernel on the feature space. k_{st} is kernel of sub-tree sets,

$$k_{st}(\boldsymbol{s}_n, \boldsymbol{s}'_n) = \sum_{c \in \boldsymbol{s}_n, c' \in \boldsymbol{s}'_n} k_{\hat{N}}(\hat{c}, \hat{c'}) \tag{6}$$

For leaf nodes, k_{st} set to be 1.

Given l training samples and the labels $(x_1, y_1), \ldots, (x_l, y_l) \in X \times Y, Y = \{-1, 1\}$, once the kernel function is determined, learning from structured data is then transformed to the standard SVM problem where the kernel function is replaced by MLMIK.

Implementation: We used three layer features, including 42-dimentional shot-level feature, nine frame-level features, and 48-dimensional region-level feature. For more details, please refer to [5]. We applied the following methods in MLMIK: (1) MLMIK-EF, (2) MLMIK-EF-C+C-, and (3) MLMIK-LF. We chose RBF kernel for k_f. The optimal parameters σ and C are determined by grid search based on the "Validation" subset. In MLMIK-EF-C+C-, we set $\frac{C^+}{C^-} = \frac{N^+}{N^-}$, which is similar to the settings in SVM.

2.2.4. Manifold Ranking (MR)

Formulation: Manifold Ranking is a graph-based semi-supervised learning method. The original description of this algorithm can be found in [10]. We consider a binary classification problem. Denote by $\mathbf{W}$ an affinity matrix with W_{ij} indicates the similarity between the *i*-th and *j*-th sample. Given two samples x_i and x_j, their similarity is often estimated based on a distance measure $d(x_i, x_j)$ and a positive radius parameter σ

$$W_{ij} = \begin{cases} \exp\left(-\frac{d(x_i,x_j)}{\sigma}\right) & \text{if } i \neq j \\ 0 & \text{else} \end{cases} \tag{7}$$

Here we choose $L1$ distance, i.e., $d(x_i, x_j) = ||x_i - x_j||$. Then the regularization framework is formulated as follows

$$f^* = \arg\min_f \left\{ \sum_{i,j} W_{ij} \left(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right)^2 + \mu \sum_i |f_i - Y_i|^2 \right\} \tag{8}$$

where $D_{ii} = \sum_j W_{ij}$, and f_i can be regarded as relevance score. We can classify x_i according to the sign of f_i (positive if $f_i > 0$ and negative otherwise). A noteworthy issue here it how to set Y_i. For general classification task, Y_i is set to 1 if x_i is labeled as positive, -1 if x_i is labeled as negative, and 0 if x_i is unlabeled. Here we decide Y_i for positive samples by validation. Usually, positive samples are expected to contribute more in video concept learning. In fact this setting is equivalent to duplicating $(1/frequency - 1)$ copies for each positive training sample, so that they are balanced with negative ones, where $frequency$ is the percentage of positive samples in training set. But here we further tune the value around $(1/frequency - 1)$. It modulates the effect of positive samples and can yield better results.

Implementation: Let $\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}$, which is usually named as *normalized graph Laplacian*. Thus, f^* in Eq. (8) can be solved in an iterative way as follows

---

1: Initialize $f^{(t)}$ where $t = 0$.
2: Update f by

$$f^{(t+1)} = \frac{1}{1+\mu}(\mathbf{I} - \mathbf{L})f^{(t)} + \frac{\mu}{1+\mu}Y$$

3: Let $t = t + 1$, and then jump to step 2 until convergence.

We make matrices $\mathbf{W}$ sparse by only keeping N (in our experiment, N=80) largest values in each row. This is a frequently used strategy in graph-based learning methods, which significantly reduces the computational costs while retaining comparable performance. We constructed the following experiments for MR: (1) MR-LF, and (2) MR-0705-LF.

2.2.5. Optimizing Multi-Graph Learning (OMGSSL)

Formulation: OMGSSL is a semi-supervised method to learn from multiple graphs [2]. Suppose we have G graphs $\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_G$, the regularization framework is formulated as

$$Q(f, \alpha) = \sum_{g=1}^{G} \sum_{i,j} \alpha_g^r \left(W_{g,ij} \left| \frac{f_i}{\sqrt{D_{g,ii}}} - \frac{f_j}{\sqrt{D_{g,jj}}} \right|^2 + \mu \sum_i |f_i - Y_i|^2 \right)$$

$$[f, \alpha] = \arg\min_{f,\alpha} Q(f, \alpha), s.t. \sum_{g=1}^{G} \alpha_g = 1 \quad (9)$$

Note that we have proposed to adopt multiple distance metrics in [2]. However, in TRECVID 2007 experiments, we have only used $L1$ for simplicity. We generate M graphs from M modalities and generate a graph to indicate temporal consistency, i.e., $G = M + 1$.

Implementation: Eq. (9) can be solved in an EM-style iterative way. But in [2], we have mentioned that when l is not extremely small, we can derive an approximate solution which can reduce computational costs. Specifically, we first compute α_g [2]. Then we compute f

$$f = \left(\mathbf{I} + \frac{1}{\sum_{g=1}^{G} \alpha_g^r \mu_g} \frac{\sum_{g=1}^{G} \alpha_g^r \mathbf{L}_g}{\sum_{g=1}^{G} \alpha_g^r} \right)^{-1} Y \quad (10)$$

Analogous to that described in manifold-ranking, Eq. (10) can be solved in an iterative way as follows

---

1: Initialize $f^{(t)}$ where $t = 0$.
2: Update f by

$$f^{(t+1)} = \frac{1}{1 + \sum_{g=1}^{G} \alpha_g^r \mu_g} (\mathbf{I} - \frac{\sum_{g=1}^{G} \alpha_g^r \mathbf{L}_g}{\sum_{g=1}^{G} \alpha_g^r}) f^{(t)} + \frac{\mu}{1+\mu} Y$$

3: Let $t = t + 1$, and then jump to step 2 until convergence.

Similar to the experiments described in section 2.2.4, we also constructed two experiments including OMGSSL-LF and OMGSSL-0705-LF. In OMGSSL-0705-LF, we utilized both the samples in TRECVID 2005 devel set and those in TRECVID 2007 "Training" subset, while we only used the TRECVID 2007 "Training" subset in OMGSSL-LF. There are four parameters should be tuned in this algorithm, including σ_g, μ_g, γ, and A. The optimal parameter configuration was selected based on "Validation" subset.

2.2.6. Temporally Consistent Gaussian Random Field (TCGRF)

Formulation: We adapted the temporal consistency assumption into graph-based semi-supervised learning and proposed a novel method called temporally consistent Gaussian random field (TCGRF) for video annotation [3]. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n samples. The first l points are labeled as $\mathbf{y} = [y_1, y_2, \ldots, y_l]^T$ with $y_i \in \{0, 1\}$ $(1 \leq i \leq l)$ and the remaining points x_u $(l + 1 \leq u \leq n)$ are unlabeled. Consider a connected undirected graph $G = (V, E)$ with node set $V = L \bigcup U$ corresponding to the n data points, where the node set $L = 1, \ldots, l$ contains labeled points and node set $U = l + 1, \ldots, l + u$ are unlabeled ones. The edges E are weighted by the $n \times n$ affinity matrix W with entry $w_{ij} = exp\{-\frac{\|x_i - x_j\|^2}{2\sigma^2}\}$ when $j \neq i$ and $w_{ii} = 0$. Denote the predicted labels of X with vector $\mathbf{f} = [f_1, f_2, \ldots, f_l, f_{l+1}, \ldots, f_n]^T = [\mathbf{f}_L^T, \mathbf{f}_U^T]^T$. We believe that temporal consistency provides valuable contextual clues to video semantic annotation. From the intuition that the label of one shot may be similar to the adjacent ones, we define a measurement of the probability that two samples have the same label in temporal order index i and j $h_{ij} = exp\{-\frac{(i-j)^2}{2\sigma_t^2}\}$ where σ_t is a scale parameter over the temporal order. Then the following energy function can be defined

$$R(\mathbf{f}) = \sum_{1 \leq i,j \leq n} h_{ij} (f_i - f_j)^2.$$

So, the low energy corresponds to a slowly varying function over the temporal order. Minimizing $R(\mathbf{f})$ subject to $\mathbf{f}_L = \mathbf{y}$ results in $f_i = \frac{1}{d_i'} \sum_{j=1}^{n} h_{ij} f_j, i \in U$ and $d_i' = \sum_{j=1}^{n} h_{ij}$. Combine temporal order adjacency and feature space similarity, we have

$$\mathbf{f} = ((1 - \alpha) D^{-1} W + \alpha D'^{-1} H)\mathbf{f} = P\mathbf{f} \quad (11)$$

subject to $\mathbf{f}_L = \mathbf{y}$, where $P = (1-\alpha)D^{-1}W + \alpha D'^{-1}H$, $D = diag(d_i)$, $D' = diag(d_i')$. Split the matrix P after the l-th row and l-th column, we will obtain the solution in matrix form as follows

$$\mathbf{f}_U^* = (I - P_{UU})^{-1} P_{UL} \mathbf{f}_L, \quad (12)$$

From Eq. (12), each sample will be assigned a real-value score indicating the degree belonging to a specific concept.

Implementation: We give the implementation of TCGRF as follows

---

1: Form affinity matrix W and H over feature space and temporal order respectively.
2: Construct matrix $P = (1 - \alpha) D^{-1} W + \alpha D'^{-1} H$ in which D is a diagonal matrix with its (i, i)-element equals to the sum of the i-th row of W, and D' is a diagonal matrix with its (i, i)-element equals to the sum of the i-th row of H.
3: Split the matrix P.
4: Predict the real-value labels for unlabeled samples by $\mathbf{f}_U^* = (I - P_{UU})^{-1} P_{UL} \mathbf{f}_L$.

We conducted experiments on TRECVID 2007 set. We used "Training" subset as the training data and learned seven models, including "TCGRF-CM33," "TCGRF-CM55," "TCGRF-CM77," "TCGRF-Auto," "TCGRF-HSV," "TCGRF-EDH," and

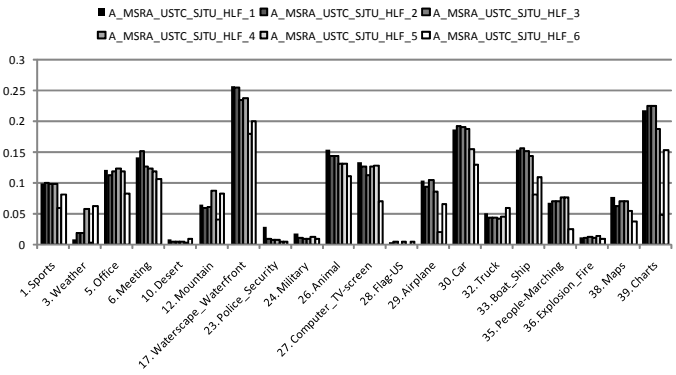

Fig. 5. The AP performances of MSRA_USTC_SJTU_HLF six runs.

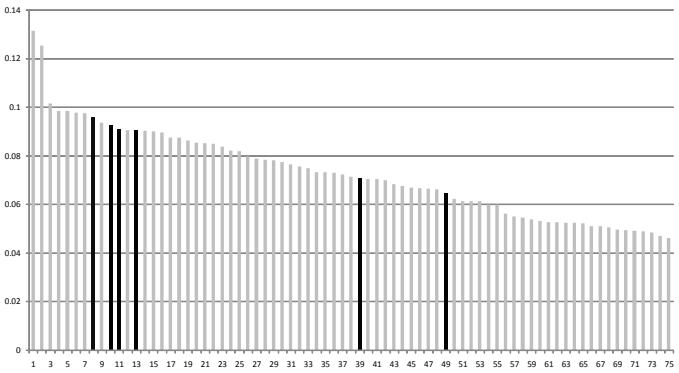

Fig. 6. Overview of all high-level feature extraction runs submitted to TRECVID 2007, ranked according to MAP. The black bars correspond to the performances of MSRA_USTC_SJTU_HLF.

"TCGRF-Wave." Based on the "validation" partition, we selected the parameters σ_t and α. The model fusion was performed over the "Fusion" subset.

2.2.7. Label Propagation through Linear Neighborhoods (LNP)

Formulation: We applied Linear Neighborhood Propagation (LNP) which propagates the labels from the labeled points to the whole dataset using the linear neighborhoods with sufficient smoothness. The original description of the algorithm can be found in [4]. We use $X = \{x_1, x_2, \ldots, x_l, x_{l+1}, \ldots, x_n\}$ to represent a set of n data objects in R^d, and $L = \{+1, -1\}$ represent the label set. The first $l(1 \leq i \leq l)$ points are labeled and the remaining points $x_u(l+1 \leq u \leq n)$ are unlabeled. The goal of LNP is to predict the labels of x_u, which can be achieved by two steps.

Step1: Construct the graph $G = (V, E)$ with node set V corresponding to the n data points. E is the edge set associated with each edge e_{ij} representing the relationship between data x_i and x_j. The edges E are weighted by the $n \times n$ affinity matrix W. Here we obtain the reconstruction weight of each data object through the following n standard quadratic programming problems.

$$\begin{array}{l} \min_{w_{ij}} \sum_{j,K:x_j,x_k \in N(x_i)} w_{ij} G^i_{jk} w_{jk} \\ s.t. \sum_j w_{ij} = 1, w_{ij} \geq 0 \end{array} \quad (13)$$

After all the reconstruction weights are computed, we will construct a sparse matrix W by $(W)_{ij} = w_{ij}$.

Step 2: Propagate the labels of the labeled data to the remaining unlabeled data $x_u(l+1 \leq u \leq n)$ using the graph constructed by the first step. We will use an iterative procedure to achieve this goal.

Let F denote the set of classifying functions defined on X, $\forall f \in F$ can assign a real value f_i to every point x_i. The label of the unlabeled data point x_u is determined by the sign of $f_u = f(x_u)$.

Implementation: We conducted experiments on TRECVID 2007 set. We used "Training," "Validation," and "Fusion" subset as the training data and learned seven models, including "LNP-TVF-CM33," "LNP-TVF-CM55," "LNP-TVF-CM77," "LNP-TVF-Auto," "LNP-TVF-HSV," "LNP-TVF-EDH," and "LNP-TVF-Wave." Based on the "Selection" partition, we performed the model fusion. It is worth notice that there is no parameter need to be tuned in this algorithm. Please refer to Figure 3 for more details.

2.3. Experimental Results

Figure 5 shows the AP performances of MSRA_USTC_SJTU_HLF six runs submitted to TRECVID 2007 for each concept. Figure 6 shows the overview of all high-level feature extraction runs submitted to TRECVID 2007, ranked according to MAP. The black bars correspond to the performances of MSRA_USTC_SJTU_HLF.

3. AUTOMATIC SEARCH

Our video search system consists of several main components, including query pre-processing and query analysis, unimodal search, multimodal fusion, re-ranking, and result refinement. The framework of the search system is shown as Figure 2 . By analyzing and pre-processing the query, the multimodal query (i.e., text, key-frames and shot) are input to individual search models, such as text-based, visual example-based and concept-based model. Then a fusion and re-ranking model is applied to aggregate the search results. Finally, the search results are refined based on face information.

Our text-based search included search at shot scene level. For the required text baseline, we only used the common ASR/MT at shot level without any re-ranking method. We got the MAP of 0.0291 (SEARCH_6). After using re-ranking and result refinement to be discussed in the latter sections, we achieved the MAP of 0.0409. When we fuse the text-based search at both the scene and shot levels with re-ranking and result

refinement, the MAP reached 0.0659 (SEARCH_4), nearly +126.5% improvement over text baseline.

The use of pre-trained concept detectors has been shown to be a powerful tool in video search systems. We implemented two methods to exploit our 36 concept detectors in high-level feature extraction, including concept mapping and concept vector. The main issue with concept mapping is discovering relevant concepts and determining the weights of each concept detector for fusion. The fusion of the two concept-based methods got the MAP of 0.0375, i.e., without using any textual information (SEARCH_5).

Another important component of our search system is multimodal fusion, re-ranking and result refinement. We proposed a re-ranking method by analyzing the features of the whole video and selecting the seeds from the initial rank list. When we only applied re-ranking and result refinement to text baseline (SEARCH_6), we got the improvement of 40.5% (i.e., MAP=0.0409). We exploited linear fusion by analyzing the query text, and then got the best runs with MAP of 0.0873 (SEARCH_1).

3.1. Text-based search

3.1.1. Query pre-processing

The query pre-processing includes query expansion, stemming, stop words removal, N-gram query segmentation, and part-of-speech tagging.

Query expansion: We implemented query expansion proposed in [11] which can suggest related terms to a given keyword including etymons, synonyms, acronyms, related semantic, etc. Specifically, we first extracted all the keywords in each query using the method developed in [12], and then chose all the synonyms and content-based suggestion of each keyword to the original query for text-based search modal.

Stemming and stop words removal: After query expansion, all the queries and ASR are stemmed using Porter's algorithm [13] and stop words are removed.

N-gram query segmentation: The query strings are segmented into term sequences based on N-gram method [14] before being input to the search engine. Please refer to [12] for more details. Given a query like "Topic 0202: Find shots of a person talking on a telephone," the keywords after stemming are "person," "talk," "telephone," this particular example has three levels of N-gram, i.e., N is from 1 to 3. Therefore, seven query segments will be generalized as: (1) unigram: person, talk, telephone; (2) bigram: person talk, person telephone, talk telephone; (3) trigram: person talk telephone. These segments were input to the search engine as different forms of the query, and the relevance scores of video shots retrieved by different query segments will be aggregated with different weights which can be set empirically. The higher gram a query segment has, the higher weight should be assigned.

Part-of-speech tagging (POS): We perform POS on the query with Tree-tagger [15]. POS represents the syntactic property of a term, e.g., noun, verb, adjective, and so on. By labeling the query string with POS tags, we can extract the terms with verb tags which are closely related to event for query class.

3.1.2. Scene boundary information

We applied scene boundary detection algorithm over the search set [16]. The scene boundary information was used in the ASR expansion, i.e., the ASR of shots are expanded to the shots in the same scene. Moreover, the ASR of the last shot in the previous scene and the first shot in the current scene should be overlapped.

3.2. Visual example-based search

Example video shot and key-frames are used in a query-by-example (QBE) method. We extracted low-level visual features used in high-level feature extraction task to form the features vector, and ranked the test key-frames according to the Euclidean distance from the query images. This process was first performed at sub-shot level and then aggregated at shot level by min operation across all sub-shots.

3.3. Concept-based search

The concept-based search consists of concept mapping and concept vector methods.

Concept mapping: Concept mapping used the results from 36 concept detectors and textual query analysis. There are two basic methods. One used WordNet to compute the lexical similarity between the query text and the textual description for each concept detector. The other possible solution is based on a manually-defined mapping directly, including identifying which concepts are relevant to the query and the extent. For example, given "Topic 0211: Find shots with sheep or goats," we triggered the "animal" detector with high weight, trigger the "outdoor" and "vegetation" detectors with low weights, as well as not triggered the "office", "prisoner", "computer TV-screen" detectors. Once the most relevant concept detectors and weights are determined, we linearly fused the detection scores of the mapped concepts and ranked the testing shots.

Concept vector: We first applied concept detection to the query examples and each key-frame in the test set, and generated feature vectors consisting of the score of each concept detectors, that is, the feature vector is a 36-dimensional vector. The succeeding processes are similar to QBE.

3.4. Fusion, re-ranking, and result refinement

3.4.1. Fusion

The fusion process is the most useful component in our search system, consisting of average and linear fusion. In 2007 search

task, we set the weight either 0 or 1. The two basic linear fusion methods are described as follows

Original ASR (OriASR) and ASR using scene boundary information (SbASR): Generally, the scenes related to event (usually the query contains verbs) have long duration. Therefore, if verbs appeared in the query, we used the score by scene boundary information as follows

$$Score^{'} = Score_{OriASR} \times 0 + Score_{SbASR} \times 1 \quad (14)$$

Otherwise,

$$Score^{'} = Score_{OriASR} \times 1 + Score_{SbASR} \times 0 \quad (15)$$

For example, "Topic 0198: Find shots of a door being opened" and "Topic 0200: Find shots of hands at a keyboard typing or using a mouse" have verbs.

Text-based search (Text) and concept-based search (Concept): If 33 concept (36 concepts except for "people," "face," and "outdoor") appeared in the query sentences, the final score will be decided as follows.

$$Score^{'} = Score_{Text} \times 0 + Score_{Concept} \times 1 \quad (16)$$

Otherwise,

$$Score^{'} = Score_{Text} \times 1 + Score_{Concept} \times 0 \quad (17)$$

For example, "Topic 0207: Find shots of waterfront (Waterscape, Waterfront) with water (Waterscape, Waterfront) and buildings (Building)" and "Topic 0211: Find shots with sheep (Animal) or goats (Animal)."

3.4.2. Re-ranking

We give the steps of re-ranking as follows.

Step1: Given an initial rank list of shots, which can be obtained by the text-based, QBE or other search methods, find the corresponding videos (which contain these shots) with high confidence. The confidence is computed as follows based on several kinds of features:

- Term frequency (TF): The frequency of the query keywords appear in the ASR of a video. The higher the value is, the higher the confidence of this video is.
- Term number (TN): The number of words in the query which appear in the ASR of a video, which is an important parameter for determining the confidence of a video.
- Query length (QL): The number of words in the query. If the query have more keywords, it is more likely to have ambiguous words.

Based on the analysis above, we compute the confidence (*Conf*) based on the three features as following

$$Conf = TF \times \frac{TN^2}{QL} \quad (18)$$

Step2: Select the shots belonging to the videos with high confidences in the initial rank list as seed shots.

Step3: Take the seed shots as query examples, and find the visually similar shots in the videos containing these seed shots. Each seed shot brings a new rank list of similar shots. We add the top shots by thresholding the confidence ($Conf > 30.0$) in this new list into the initial rank list. The ranking scores for these shots are computed as follows.

$$Score^{'} = Score + \frac{Score_{seed}}{Dist_{seed}} \times w \quad (19)$$

where $Score^{'}$ and $Score$ are the new score and old score, respectively. If the new shots do not exist in the initial rank list, $Score = 0$. $Score_{seed}$ is the score of the selected seeds, $Dist_{seed}$ is the distance between the shot and seed, and w can be determined based on different queries and different seeds. In our experiment, we fixed w as 1.0. Intuitively, we can observe that if the new shots already exist in the initial rank list, this procedure will increase their ranking scores.

Step4: When all the seed shots are processed, sort the updated scores and get the final rank list.

3.4.3. Result refinement

The detection of face provides information of person-related shots. The queries are classified into the following categories:

- Term "Normal face": queries containing "people", "person" or "hands." For example, "Topic 0197: Find shots of one or more people walking up stairs," and "Topic 0202: Find shots of a person talking on a telephone." Usually, these queries belong to the person and event related shots. Intuitively, there are faces but no big faces appearing in the shots. The error results in the initial rank list may be introduced by ASR. We eliminate the shots which have big face or no face, and refine search results with the face detector in high-level feature extraction.
- Term "Big face": queries containing "interview." For example, "Topic 0213: Find shots of a woman talking toward the camera in an interview - no other people visible." For this category, we only remain the shots containing big faces.
- "Others": the complement to the above two.

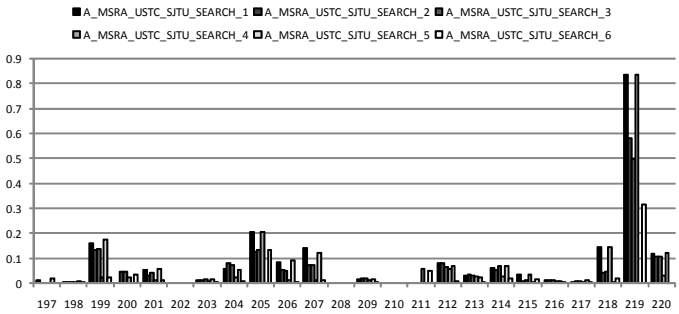

Fig. 7. The performances of six runs for type A automatic search of each query.

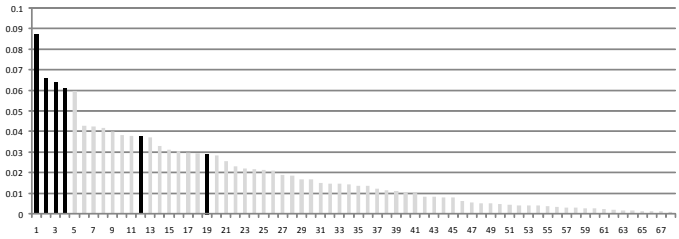

Fig. 8. The performances of six submitted runs for type A automatic search. The black bars are from MSRA_USTC_SJTU_SEARCH.

3.5. Experiments and Results

We submitted 6 automatic type A runs for search task. Figure 7 shows the performances of six submitted runs for automatic search. Our two baseline run had our lowest MAP of 0.0291 and 0.0375, ranked in the top 19 and 12 among all the type A automatic runs as shown in Figure 6, respectively. When we introduced the scene boundary information in the text-based search, added re-ranking and result refinement, the MAP score was significantly improved to 0.0659 and ranked in the top 2. The best run is the fusion of text-based and concept mapping methods with the MAP of 0.0873, ranked as the No. 1 among all the runs.

4. CONCLUSIONS

We participated in high-level feature extraction and automatic search tasks in TRECVID 2007. In this paper, we have presented preliminary results and methods for these two tasks. We observed that by fusing MLMIK and CML with SVM, the performances of high-level feature extraction are significantly improved, and using scene information, reranking and result refinement can also improve the performance of automatic search.

5. REFERENCES

[1] X.-S. Hua, T. Mei, W. Lai, M. Wang, J. Tang, G.-J. Qi, L. Li, and Z. Gu, "Microsoft Research Asia TRECVID 2006: High-level Feature Extraction and Rushes Exploitation," in *NIST TRECVID Workshop*, November 2006.

[2] M. Wang, X.-S. Hua, X. Yuan, Y. Song, and L.-R. Dai, "Optimizing multi-graph learning: towards a unified video annotation scheme," in *Proceedings of ACM Multimedia*, 2007.

[3] J. Tang, X.-S. Hua, T. Mei, G.-J. Qi, and X. Wu, "Video Annotation Based on Temporally Consistent Gaussian Random Field," *Electronics Letters*, vol. 43, no. 8, 2007.

[4] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," in *ICML*, 2006.

[5] Z. Gu, T. Mei, X.-S. Hua, J. Tang, and X. Wu, "Multi-layer multi-instance kernel for video concept detection," in *Proceedings of ACM Multimedia*, 2007.

[6] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," in *Proceedings of ACM Multimedia*, 2007.

[7] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001.

[8] G.-J. Qi, X.-S. Hua, Y. Rui, T. Mei, J. Tang, and H.-J. Zhang, "Concurrent multiple instance learning for image categorization," in *CVPR*, 2007.

[9] Z.-J. Zha, Y. Liu, T. Mei, , and X.-S. Hua, "Video concept detection using Support Vector Machines – TRECVID 2007 evaluations," Tech. Rep., Microsoft Research Asia, Oct. 2007.

[10] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003.

[11] W. Gao, C. Niu, J.-Y. Nie, M. Zhou, J. Hu, K.-F. Wong, and H.-W. Hon, "Cross-lingual query suggestion using query logs of different languages," in *SIGIR*, 2007.

[12] J. Liu, W. Lai, X.-S. Hua, Y. Huang, and S. Li, "Video search re-ranking via multi-graph propagation," in *Proceedings of ACM Multimedia*, 2007.

[13] M. F. Porter, "An algorithm for suffix stripping," 1997.

[14] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, 1992.

[15] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *International Conference on New Methods in Language Processing*, Manchester, UK, 1994.

[16] Z. Gu, T. Mei, X.-S. Hua, X. Wu, and S. Li, "Ems: Energy minimization based video scene segmentation," in *ICME*, 2007.

Method ID	Method Name	Classifier	Training Data	Fusion (Early or Late)	Model ID	Model Name	Modality (Low-level feature)
Method 01	SVM-LF	SVM	2007 devel set (T); and 7 concepts also use the additional 2005 positive samples.	Late Fusion (Linear weighted fusion of 7 models)	Model 01-1	SVM-CM33	Color Moment 3-by-3
					Model 01-2	SVM-CM55	Color Moment 5-by-5
					Model 01-3	SVM-CM77	Color Moment 7-by-7
					Model 01-4	SVM-Auto	Auto-correlogram
					Model 01-5	SVM-HSV	HSV Color Histogram
					Model 01-6	SVM-EDH	Edge Distribution Histogram
					Model 01-7	SVM-Wave	Wavelet PWT-TWT Texture
Method 02	SVM-TVS-LF	SVM	90% of 2007 devel set (T+V+S); and 7 concepts used the additional 2005 positive samples	Late Fusion (Linear weighted fusion of 7 models)	Model 02-1	SVM-TVS-CM33	Color Moment 3-by-3
					Model 02-2	SVM-TVS -CM55	Color Moment 5-by-5
					Model 02-3	SVM-TVS -CM77	Color Moment 7-by-7
					Model 02-4	SVM-TVS-Auto	Auto-correlogram
					Model 02-5	SVM-TVS-HSV	HSV Color Histogram
					Model 02-6	SVM-TVS-EDH	Edge Distribution Histogram
					Model 02-7	SVM-TVS-Wave	Wavelet PWT-TWT Texture
Method 03	SVM-TV-EF-CV3	SVM	80% of 2007 devel set (T+V); 3 fold Cross Validation	Early Fusion; 9 frame-level features, 1181-D	Model 03-1	SVM-TV-EF-CV3	Concatenated feature, including 9 frame-level features in Table 2.
Method 04	SVM-TVS-EF-CV3	SVM	90% of 2007 devel set (T+V+S); 3 fold Cross Validation	Early Fusion; 9 frame-level features	Model 04-1	SVM-TVS-EF-CV3	ditto
Method 05	SVM-TVS -LF-CV3	SVM	90% of 2007 devel set (T+V+S); 3 fold Cross Validation	Late Fusion (Linear weighted fusion of three models)	Model 05-1	SVM-TVS-CM55-CV3	Color Moment 5-by-5
					Model 05-2	SVM-TVS-Auto-CV3	Auto-correlogram
					Model 05-3	SVM-TV S-Wave-CV3	Wavelet PWT-TWT Texture
Method 06	MLMIK-EF	Multi-layer Multi-instance Kernel (MLMIK)	70% of 2007 devel set (T)	Early Fusion; shot-level feature, frame-level feature, and region-level feature	Model 06-1	MLMIK-EF	Shot-level feature, Frame-level feature, and Region-level feature
Method 07	MLMIK-EF-C+C-	Multi-layer Multi-instance Kernel (MLMIK)	70% of 2007 devel set (T); tune cost parameter for positive sample (C+) and negative sample (C-)	Early Fusion; shot-level feature, frame-level feature, and region-level feature	Model 07-1	MLMIK-EF-C+C-	Shot-level feature, frame-level feature, and region-level feature
Method 08	MLMIK-LF	Multi-layer Multi-instance Kernel (MLMIK)	70% of 2007 devel set (T)	Late Fusion (Linear weighted fusion of three models); shot-level feature, frame-level feature, and region-level feature	Model 08-1	MLMIK-CM55	Shot-level feature, Frame-level feature: Color Moment 5-by-5, and Region-level feature
					Model 08-2	MLMIK-Auto	Shot-level feature; Frame-level feature: Auto-correlogram; Region-level feature
					Model 08-3	MLMIK-Wave	Shot-level feature; Frame-level feature: Wavelet PWT&TWT Texture; Region-level feature
Method 09	CML-LF	Correlative Multi-Label Annotation (CML)	70% of 2007 devel set (T)	Late Fusion (Linear weighted fusion of three models)	Model 09-1	CML-CM55	Color Moment 5-by-5
					Model 09-2	CML-Auto	Auto-correlogram
					Model 09-3	CML -Wave	Wavelet PWT-TWT Texture
Method 10	MR-0705-LF	Manifold Ranking (MR)	70% of 2007 devel set (T); and most of the 36 concepts also use 2005 development set	Late Fusion (Linear weighted fusion of eight models)	Model 10-1	MR-0705-CM33	Color Moment 3-by-3
					Model 10-2	MR-0705-CM55	Color Moment 5-by-5
					Model 10-3	MR-0705-CM77	Color Moment 7-by-7
					Model 10-4	MR-0705-Auto	Auto-correlogram
					Model 10-5	MR-0705-HSV	HSV Color Histogram
					Model 10-6	MR-0705-EDH	Edge Distribution Histogram
					Model 10-7	MR-0705-Wave	Wavelet PWT-TWT Texture
					Model 10-8	MR-0705-CTF	Concatenated feature including Co-occurrence Texture and Face.

Method 11	MR-LF	Manifold Ranking (MR)	70% of 07 devel set (T)	Late Fusion (Linear weighted fusion of eight models)	Model 11-1	MR-CM33	Color Moment 3-by-3
					Model 11-2	MR-CM55	Color Moment 5-by-5
					Model 11-3	MR-CM77	Color Moment 7-by-7
					Model 11-4	MR-Auto	Auto-correlogram
					Model 11-5	MR-HSV	HSV Color Histogram
					Model 11-6	MR-EDH	Edge Distribution Histogram
					Model 11-7	MR-Wave	Wavelet PWT-TWT Texture
					Model 11-8	MR-CTF	Concatenated feature, including Co-occurrence Texture and Face.
Method 12	OMGSSL-0705-LF	Optimal Multi-Graph Semi-Surprised Learning (OMGSSL)	70% of 2007 devel set (T); and most of the 36 concepts also use 2005 devel set	Late Fusion (Linear weighted fusion of eight models)	Model 12-1	OMGSSL -0705-CM33	Color Moment 3-by-3
					Model 12-2	OMGSSL -0705-CM55	Color Moment 5-by-5
					Model 12-3	OMGSSL -0705-CM77	Color Moment 7-by-7
					Model 12-4	OMGSSL -0705-Auto	Auto-correlogram
					Model 12-5	OMGSSL -0705-HSV	HSV Color Histogram
					Model 12-6	OMGSSL -0705-EDH	Edge Distribution Histogram
					Model 12-7	OMGSSL -0705-Wave	Wavelet PWT-TWT Texture
					Model 12-8	OMGSSL -0705-CTF	Concatenated feature, including Co-occurrence Texture and Face.
Method 13	OMGSSL-LF	Optimal Multi-Graph Semi-Surprised Learning (OMGSSL)	70% of 2007 devel set (T)	Late Fusion (Linear weighted fusion of eight models)	Model 13-1	OMGSSL-CM33	Color Moment 3-by-3
					Model 13-2	OMGSSL-CM55	Color Moment 5-by-5
					Model 13-3	OMGSSL-CM77	Color Moment 7-by-7
					Model 13-4	OMGSSL-Auto	Auto-correlogram
					Model 13-5	OMGSSL-HSV	HSV Color Histogram
					Model 13-6	OMGSSL-EDH	Edge Distribution Histogram
					Model 13-7	OMGSSL-Wave	Wavelet PWT-TWT Texture
					Model 13-8	OMGSSL-CTF	Concatenated feature, including Co-occurrence Texture and Face.
Method 14	TCGRF-LF	Temporally Consistent Gaussian Random Field (TCGRF)	70% of 2007 devel set (T)	Late Fusion (Linear weighted fusion of seven models)	Model 14-1	TCGRF -CM33	Color Moment 3-by-3
					Model 14-2	TCGRF -CM55	Color Moment 5-by-5
					Model 14-3	TCGRF -CM77	Color Moment 7-by-7
					Model 14-4	TCGRF -Auto	Auto-correlogram
					Model 14-5	TCGRF -HSV	HSV Color Histogram
					Model 14-6	TCGRF -EDH	Edge Distribution Histogram
					Model 14-7	TCGRF -Wave	Wavelet PWT-TWT Texture
Method 15	LNP-TVF-LF	Linear Neighborhood Propagation (LNP)	90% of 2007 development (T+V+F)	Late Fusion (Linear weighted fusion of seven models)	Model 15-1	LNP-TVF -CM33	Color Moment 3-by-3
					Model 15-2	LNP-TVF -CM55	Color Moment 5-by-5
					Model 15-3	LNP-TVF -CM77	Color Moment 7-by-7
					Model 15-4	LNP-TVF -Auto	Auto-correlogram
					Model 15-5	LNP-TVF -HSV	HSV Color Histogram
					Model 15-6	LNP-TVF -EDH	Edge Distribution Histogram
					Model 15-7	LNP-TVF -Wave	Wavelet PWT-TWT Texture

Fig. 3. The description of all the methods and models for high-level feature extraction.