VITALAS at TRECVID-2008

Christos Diou [*†] Christos Papachristou[*†] Panagiotis Panagiotopoulos[*†]
George Stephanopoulos[†] Nikos Dimitriou[†] Anastasios Delopoulos[*†]
Henning Rode[‡] Robin Aly[§] Arjen P. de Vries[‡] Theodora Tsikrika[‡]

Abstract

High Level Feature Extraction runs.

1. A_VITALAS.CERTH.ITI_1: Combination of early fusion and concept score fusion with feature selection.
2. A_VITALAS.CERTH.ITI_2: Concept score fusion with feature selection.
3. A_VITALAS.CERTH.ITI_3: Clustering within feature space and concept score fusion with feature selection.
4. A_VITALAS.CERTH.ITI_4: Concept score fusion for selected low level features.
5. a_VITALAS.CERTH.ITI_5: Mandatory type 'a' run, concept score fusion for selected low level features.

This is the first participation of VITALAS in TRECVID. In the high level feature extraction task, our submitted runs are based mainly on visual features, while one run utilizes audio information as well; the text is not used. The experiments performed aim at evaluating the effectiveness of different approaches to input processing prior to the final classification (i.e., ranking) stage. These are (i) clustering of feature vectors within the feature space, (ii) fusion of classifier output scores for other concepts and (iii) feature selection. The results indicate that (i) fusion of the classifier output of other concepts can provide valuable information, even if the original features are not discriminative, (ii) feature selection generally improves the results (especially when the original number of dimensions is high) and (iii) clustering within the feature space with small number of clusters does not seem to provide any significant additional information.

Search runs.

1. ASR ranks shots solely based on the ASR collection,
2. TOP20 uses only the 20 highest scored concepts for each shot,
3. SIGMA2 defines a deviation-based threshold to determine which concepts will be considered,
4. DEVTOP20 combines the previous two methods TOP20 and SIGMA2,
5. DEVTOP50 works as DEVTOP20 but using the top 50 concepts,
6. ASR-DEVTOP20 combines ASR and concept-based ranking.

Our experiments for the search task are focused on concept retrieval. We generate an artificial text collection by merging context descriptions according to the probability of each concept to occur in a given shot. To make the approach feasible, we further need to investigate techniques for pruning the dense shot concept matrix. Despite the poor overall retrieval quality, our concept search runs show a similar performance to the pure ASR run. Only the combination of ASR and concept search yields considerable improvements. Among the tested concept pruning strategies, the simple top k selection works better than the deviation-based thresholding.

1 Introduction

VITALAS is a EU-funded Integrated Project that aims at the development of a system capable of

[*] Multimedia Understanding Group
Information Processing Laboratory
Electrical and Computer Engineering
Aristotle University of Thessaloniki, Greece

[†] Informatics and Telematics Institute
Centre for Research and Technology Hellas

[‡] Centrum voor Wiskunde en Informatica (CWI)
Amsterdam, The Netherlands

[§] Database Group, University of Twente

large-scale indexing and retrieval of video and images, specifically targeted towards multimedia professionals and archivists [1]. This first participation of VITALAS in the TRECVID benchmark aims at the evaluation of ideas related to one of the most challenging research problems VITALAS faces, namely the indexing and retrieval of multimedia data at the semantic level. We therefore submitted runs for the High Level Feature Extraction (HLFE) and Search tasks, with the Search task focusing on retrieval using concept information.

For the HLFE task, four type 'A' and one type 'a' runs were submitted. The runs utilize a number of low level feature vectors describing video keyframes in terms of color, texture, motion and keypoint information. These features are presented in Section 2. However, none of the runs depends solely on low level features. The focus is on approaches that process low level feature vectors in order to generate new descriptions with higher discriminative power, thus improving the ranking of keyframes with respect to concepts with the use of classifiers. The methods applied are (i) fusion of multiple concept scores (Section 3.3), (ii) clustering in the feature space (Section 3.2) and (iii) feature selection (Section 3.5). In addition, a donated audio concept detector, that provides scores for 12 audio concepts, is used in one of the runs. Results indicate that there is still much work to be done in order for these post-processing approaches to be efficient. They lead, however, to useful conclusions, outlined in Section 4.1.

For the search task, our main focus lies on concept pruning strategies. Observing that a dense shot concept matrix results in inefficient retrieval, regarding both space and time, we are investigating techniques for selecting for each shot an optimal subset of concepts that should be indexed. The concept retrieval itself is mapped to a text search task by using the textual concept description for building an artificial text collection.

2 Low level features

This section presents the low-level features that are used for the HLFE task. Depending on the feature extraction algorithm applied, these can be categorized into "global", "regional" and "local" features.

2.1 Global features

Global features are extracted directly from the entire video keyframe area. The ones employed are a variant of the MPEG-7 *Color Structure Descriptor*(GL-CSD), a variant of the *Dominant Color Descriptor* [2] (GL-DCOLOR) and a feature vector extracted using the Hough transform (GL-HOUGH). Instead of using the descriptors already available from the MPEG-7 experimentation model [3], GL-CSD and GL-DCOLOR have been implemented anew, to allow for certain modifications.

For GL-CSD the image colorspace is transformed to the *Gaussian color model* [4] and a 3×3 box is used to compute 128 color structure histogram bins ($8 \times 4 \times 4$, for color intensity and two spectral components respectively).

For GL-DCOLOR, *octrees* [5] are used in order to achieve efficient and fast color reduction in the keyframe (see also [6]). More specifically, a color signature is extracted from each keyframe I,

$$CS_I = \{(c_0, p_0), \ldots, (c_N, p_N)\} \tag{1}$$

where each c_i is a dominant color and p_i is the corresponding percentage of c_i in the image, after reduction with octrees has been performed. Since the number of colors N depends on the image, the Earth Mover's Distance (EMD) [7] is employed in order to compare two color signatures. To produce the final feature vector, a method similar to the one used in [8] is applied. Since this method is also used for the extraction of other low level features, it is presented here for completeness.

Manually labeled image regions corresponding to 15 concepts are considered[1] that form 15 sets $\mathbf{P}_i, i = 1, \ldots, 15$ of reference images (these are denoted *proto-concepts* in [8]). For a keyframe I and a set $\mathbf{P}_i$ two values can be computed, namely the average and best distance of I to the images in $\mathbf{P}_i$,

$$H^i_{avg} = \frac{1}{|\mathbf{P}_i|} \sum_{j=0}^{|\mathbf{P}_i|} EMD(CS_I, CS_{P_i(j)}) \tag{2}$$

$$H^i_{best} = \min_j (EMD(CS_I, CS_{P_i(j)})) \tag{3}$$

This approach leads to a 30-dimensional feature vector for dominant color.

For the description of keyframes based on the direction of lines detected by the Hough transform, a simple histogram of angles is used that was extracted from the Hough accumulator matrix. An accumulator matrix with 12 angles is used and after computing the values for the matrix cells, only values that exceed a predefined threshold and are

[1] The concepts are: *building*, *car*, *charts*, *crowd*, *desert*, *fire*, *maps*, *mountain*, *road*, *sky*, *smoke*, *snow*, *US-flag*, *vegetation*, *water* and the corresponding image regions have been manually extracted from images in the TRECVID-2005 development set.

local maxima are kept (otherwise they are set to zero). Adding all rows of the accumulator matrix leads to a 12-bin angle histogram, that is normalized to a unit vector with its L^2 norm to produce the final feature vector.

2.2 Regional features

Regional features are extracted based on information computed at different image regions followed by an integration step that produces a single representation for the entire keyframe. The regional features that were used are based on color (R-DCOLOR), texture (R-WBL, R-FBANK and R-FBANKR) and motion information (R-MVECTOR).

Color and texture-based features are all extracted using the same methodology. Two region sizes are considered, namely $\frac{1}{2}$ and $\frac{1}{6}$ of the keyframe width and height. These two region sizes correspond to 9 and 121 overlapping image regions at points $(i\frac{1}{2}r_x, j\frac{1}{2}r_y)$, where r_x and r_y correspond to the region width and height and $0 \leq i < \frac{2\text{width}}{r_x} - 1$, $0 \leq j < \frac{2\text{height}}{r_y} - 1$. For each region, a signature is extracted, depending on the feature used. The feature vector is computed using the average and best distances from images of reference concepts, in a manner similar to GL-DCOLOR,

$$H^{i,s_R}_{avg} = \frac{1}{|\mathbf{P}_i||R|}\sum_{r\in R}\sum_{j=0}^{|\mathbf{P}_i|} D(S_r, S_{P_i(j)}) \tag{4}$$

$$H^{i,s_R}_{best} = \min_{r\in R}(\sum_{j=0}(S_r, S_{P_i(j)})) \tag{5}$$

where s_R is the region size corresponding to the set of regions R (of the same size), $S_r, S_{P_i(j)}$ are the signatures of the region r and image $P_i(j)$ of the i-th reference concept respectively and D is a dissimilarity metric between signatures (the operator min becomes max when a similarity metric is used instead). Hence there are two numbers for each region size s_R and reference concept i.

The regional dominant color descriptor (R-DCOLOR) is a variation of its global counterpart, that uses the above scheme with only one region size ($\frac{1}{2}$ of the image width and height) to take into account regional color information, by computing a color signature for each region instead of the entire image. It still has 30 elements, since two values are again computed for each of the 15 reference concepts.

For describing texture, two types of features are used, one based on the integrated Weibull distribution of edges in a region and and one based on a filterbank response of each region. For the Weibull-based features, the approach described in [8] is followed. The colorspace of each keyframe is first transformed to the Gaussian Color Model. Each color channel is then filtered using one Gaussian derivative filter for each image direction (horizontal and vertical). This process is repeated for two filter scales (i.e., two values of σ). The edges in each region of the 12 resulting images is assumed to follow an Integrated Weibull distribution,

$$\frac{\gamma}{2\gamma^{\frac{1}{\gamma}}\beta\Gamma(\frac{1}{\gamma})}\exp\left\{-\frac{1}{\gamma}\left|\frac{r-\mu}{\beta}\right|^{\frac{1}{\gamma}}\right\} \tag{6}$$

where β and γ are the distribution parameters and μ is assumed to be zero (this is ensured by preprocessing of the region values). Computation of the β and γ values is performed numerically, while comparison between two distributions is achieved using a metric derived in [8] from the Cramér von Mises statistic $C = \frac{\min(\beta_1,\beta_2)}{\max(\beta_1,\beta_2)}\frac{\min(\gamma_1,\gamma_2)}{\max(\gamma_1,\gamma_2)}$. There are two region sizes, two scales for the Gaussian derivative filters and two values for each region size and scale leading to 8 values for each reference concept and a 120-d feature vector based on the Integrated Weibull distribution.

A different approach uses a filterbank consisting of two types of 'difference of Gaussian' (DoG1 and DoG2) filters and a set of 'difference of offset Gaussian' (DooG) filters. Details of each filter can be found in [9]. The mean and standard deviation of the filter responses are computed for each region (after the keyframe has been converted to grayscale) and are used as a signature, i.e., $S_{\text{FBANK}} = [\mu_1\,\sigma_1\,\mu_2\,\sigma_2\,\ldots\,\mu_N\sigma_N]$ for a filterbank with N filters, and these are compared against the reference images using the euclidean distance. For R-FBANK, the filterbank consists of two DoG1 filters (with different scales), two DoG2 filters and twelve DooG filters. For the DooG filters, since they are directional and symmetric along their two principal axes, 6 pairs of filters with two different scales are used to cover the $[0, \pi]$ angle range. A single region size is considered ($\frac{1}{2}$ width and height) and (4), (5) are computed for each filter separately, leading to a $16 \times 2 \times 15 = 480$-d feature vector. A reduced version of this feature is also computed, using the same procedure, but with one DoG1, one DoG2 and two DooG filters, leading to a 120-d feature vector (R-FBANKR).

The R-MVECTOR feature is computed based on the motion vectors of the encoded video stream. For each macroblock (16 × 20 in total for the

QVGA resolution videos of TRECVID-2008) the motion vector is extracted in each frame of the shot. By adding all vectors, a single motion vector $\mathbf{m}_{ij} = [m_{ijx}\ m_{ijy}]^T$ is computed for each macroblock for the entire shot. These motion vectors are then used at three processing resolutions. For each resolution $r = 0, 1, 2$, there exist 1, 5 and 25 equal-sized overlapping regions, i.e., $2^r \times 2^r$ regions at points $(i\frac{20}{2^r}, j\frac{16}{2^r})$ for $i, j = 0, \ldots, 2^r - 1$ and $(2^r - 1) \times (2^r - 1)$ regions at points $((i + \frac{1}{2})\frac{20}{2^r}, (j + \frac{1}{2})\frac{16}{2^r})$, for $i, j = 0, \ldots, 2^r - 2$ (where point unit is a macroblock).

For each resolution, the mean and standard deviation of the motion vectors of macroblocks in each region are stored (4 elements per region). In addition, for each resolution the standard deviation of motion for all regions is stored separately (2 additional elements for each resolution, except resolution 0 that only has a single region). This process leads to R-MVECTOR having 128 elements for the description of motion in a shot.

2.3 Local features

A feature vector based on local features (L-KBSURF) is also computed. All images are resized to half width and height and initially, a set of keypoints are extracted from each keyframe using the Kadir-Brady detector [10, 11]. For each keypoint, the SURF [12] descriptor is computed, resulting to a 64-d feature vector. Then, following an approach similar to [13], all feature vectors extracted from the keyframes in the development set are used to create 50 clusters.

A grid splits each keyframe into 2×3 (i.e., $\frac{1}{2}$ height by $\frac{1}{3}$ width) non-overlapping regions of equal size. For each keypoint in each region the nearest cluster center is found and this process leads to the construction of a 50-bin histogram for each region. Concatenation of the 6 histograms leads to the computation of a 300-dimensional feature vector.

3 High level features

For the high level feature extraction task, the general model of *low level feature extraction, preprocessing and classification* has been used. Since the pre-processing step itself often involves the use of classifiers, the classification approach is presented first.

3.1 The classifier

The base classifier is the SVM classifier with RBF kernel [14, 15]. Libsvm [16, 17] is the selected implementation for the experiments performed. Still, directly applying an SVM classifier with cross-validation for the computation of the optimal classification parameters (C, γ, or the class weights $w+$ and $w-$) suffers from two major drawbacks: (i) It is computationally demanding, to the degree that extensive experimentation is impossible at reasonable cost (in terms of CPU time) (ii) its performance may suffer when the positive samples for a concept are sparse, as is the case for some concepts in this year's benchmark that have a very low frequency of occurrence in the training data. The latter becomes evident when the dimensionality of the feature space is high.

In order to tackle (at least, partially) these two problems, a classifier that employs SVM with rankboost [18] and sampling in each iteration has been used, such as the one described in [19, 13]. This process is illustrated in Algorithm 3.1. For the initial step of this algorithm a logarithmic search for the optimal C parameter of the SVM and γ parameter of the RBF kernel is performed using cross-validation. These parameters are then used for the rest of the iterations.

As illustrated in [13], the rankboost approach combined with sampling has significantly lower computational complexity, compared to SVM with cross-validation. Regarding the problem of sample sparsity: When very few positive samples for the high-level feature are available and when these are not easily discriminated within the feature space, the problem of overfitting becomes apparent; almost all positive samples become support vectors. Although there is no rigorous evidence that the employed approach performs better in this aspect, balancing the positive and negative samples of the training set and using rankboost in order to emphasize on misclassified samples can bypass the overfitting problem in many cases.

3.2 Clustering in the feature space

In order to examine and exploit possible structures of samples within each feature space, an approach based on feature space clustering is also applied for each feature separately. More specifically, all visual features have been extracted from the TRECVID-2005 development set and the k-means algorithm applied to create 20 clusters for each feature. Thus, each feature is replaced by a 20-d feature vector of distances between the feature vector and the 20

Algorithm 1 Classification with rankboost, sampling and SVM

Input: A training set $\mathbf{T}_C$ for a high level feature C and two disjoint sets $T_1, T_{-1} \subseteq \mathbf{T}_C$ of positive and negative samples of C. A threshold on the maximum number of positive samples selected M_p and the number of iterations N.

Output: A set of weights α_i and trained classification models h_i, $i = 1, \ldots, N$.

```
/* Initialize */
$v_1(x) = \begin{cases} \frac{1}{|T_1|} & x \in T_1 \\ \frac{1}{|T_{-1}|} & x \in T_{-1} \end{cases}$
$w = \min(M_p, |T_1|)$
for $i = 1$ to $N$ do
    $T_s = \text{sample}(\mathbf{T}_C, v_i, w)$
    /* Train the $i$'th model, $h_i : X \to \mathbb{R}$ */
    $h_i = \text{train\_svm}(T_s)$
    $r = \sum_{x\in\mathbf{T}_C} v_i(x) y_j h_i(x)$, $\alpha_i = \frac{1}{2}\log\left(\frac{1+r}{1-r}\right)$
    $v_{i+1} = \begin{cases} \frac{v_i(x)\exp(-\alpha_i h_i(x))}{\sum_{x\in T_1} v_i(x)\exp(-\alpha_i h_i(x))} & x \in T_1 \\ \frac{v_i(x)\exp(\alpha_i h_i(x)}{\sum_{x\in T_{-1}} v_i(x)\exp(\alpha_i h_i(x))} & x \in T_{-1} \end{cases}$
end for
```

where the sampling routine is

```
sample($\mathbf{T}, v, w$)
for all $x \in \mathbf{T}$ do
    /* Pick a random number in $[0, 1]$ */
    $r = \text{random}([0, 1])$
    $p = v(x)w$
    if $p > r$ then
        select $x$ ($x \in T_s$)
    end if
end for
return $T_s$
```

Prediction can be performed using $H(x) = \sum_{i=1}^{N} \alpha_i h_i(x)$

cluster centers computed.

The general goals of this approach are (i) To reduce dimensionality, since only 20 clusters are used within feature spaces with hundreds of dimensions. (ii) For feature spaces where samples form groups or have structure, to capture and describe that structure. For selecting the number of clusters, it has been assumed that all feature spaces will have the same number of clusters for all concepts. The number of clusters has been decided after experimenting in the TRECVID-2005 development set using the LSCOM annotations [20, 21] for various high level features.

3.3 Fusion of LSCOM concept scores

Describing a concept in terms of low level features is not always efficient, due to the poor discriminative ability of the features. It is possible, however, to use the output of concept detectors (i.e., classifier scores) to describe other concepts. The most apparent and intuitive justification for this is the strong correlation that may exist between concepts. This correlation may be due to relations such as hyponymy, meronymy etc, or simply by a high co-occurrence frequency. Hence it is very reasonable that one would look for the concept 'Sea', for example, when searching for the concept 'Boat'. The questions that automatically arise with respect to the practical application of such an approach have to do with: (i) The selection of appropriate concepts and (ii) the detection of these concepts.

Given a high level feature C, the LSCOM ontology [20] and corresponding annotations [21], it is possible to select a set of L LSCOM concepts $\mathbf{S}_C$ that are relevant to C and train a set of L classifiers H_{C_i}, $C_i \in \mathbf{S}_C$ using the TRECVID-2005 development set and some low level feature. For a sample $x \in \mathbf{T}_C$ (i.e., the training set of C), the score outputs of H_{C_i} can be concatenated in a feature vector

$$\mathbf{F}_x = [H_{C_1}(x) \ \ldots \ H_{C_L}(x)] .$$

The feature vectors $\mathbf{F}_x$ can be used to train a final classifier that will compute the final ranking for C.

For the concept set $\mathbf{S}_C$, only LSCOM concepts with more than 50 annotated positive samples are considered. Initially, experiments were performed with manual selection of the concepts C_i in $\mathbf{S}_C$. The selection was performed on the basis of visual or conceptual similarity, as well as the expected co-occurrence of C_i with C. For evaluation, 5-fold cross-validation was performed on the development set and the average precision achieved was not satisfactory.

Using a different approach, selection of a concept C_i is performed based on the confidence on its detector H_{C_i}. The judgment is made on the basis of both the number of positive samples used during training and its performance on a held-out test dataset. A concept is selected if the number of positive samples and the average precision exceed predefined thresholds. No manual selection is performed, apart from the removal of named entities from the set. This process leads to a set $\mathbf{S}_C$ of 89 concepts that is chosen to be the same for all concepts C, regardless of the low level feature that is used originally for building H_{C_i}. Repeating the experiment using this set, lead to a significant improvement in the average precision achieved.

These observations lead to an interesting conclusion: A good choice of feature $\mathbf{F}_x$ is not necessarily obtained when the concepts in the set $\mathbf{S}_C$ are conceptually related or have a high co-occurrence with C. What is more important is that the models H_{C_i} are robust, in the sense that (i) positive samples in the training dataset used to produce them are not too sparse and (ii) H_{C_i} results into average precision values significantly above random selection.

3.4 Audio concepts

Although no low level audio features have been used in the experiments, audio information is included in one of the runs by using the score output of an audio concept detector donated by the Netmedia team of Fraunhofer IAIS. More specifically, the audio concept detector (based on [22]) performs segmentation of each video and for each audio segment produces the log-likelihood of a set of 12 audio concepts[2]. Audio segments are mapped to shots and for each shot and each audio concept the maximum log-likelihood from all audio segments is selected, thus forming a 12-d vector.

3.5 Feature selection

The combined use (via simple vector concatenation/early fusion) of all low level feature vectors, the feature vectors produced from score fusion and clustering as well as the audio concept scores yields a feature vector of 2071 dimensions. Apart from computational complexity issues, for such high dimensional spaces the classifier performance deteriorates significantly. This becomes apparent if the relatively small number of training samples is considered. In order to remove redundant features and at the same time select the most informative ones, the feature selection procedure described in [23] is employed.

4 HLFE runs

The submitted high-level feature extraction runs aim at examining the performance of various system configurations that use the low level features and pre-processing components described in the previous sections. The annotation unit is the video shot. The TRECVID-2005 and TRECVID-2008 development sets have been used for system development and training. The annotations used are the LSCOM annotations and the result of the TRECVID-2008 collaborative annotation effort kindly organized by LIG and LIRIS groups [24]. No special keyframe extraction algorithm is applied in the test set, other than a simple periodic selection policy.

Figure 1 illustrates the system structure. Dashed lines correspond to optional system operations.

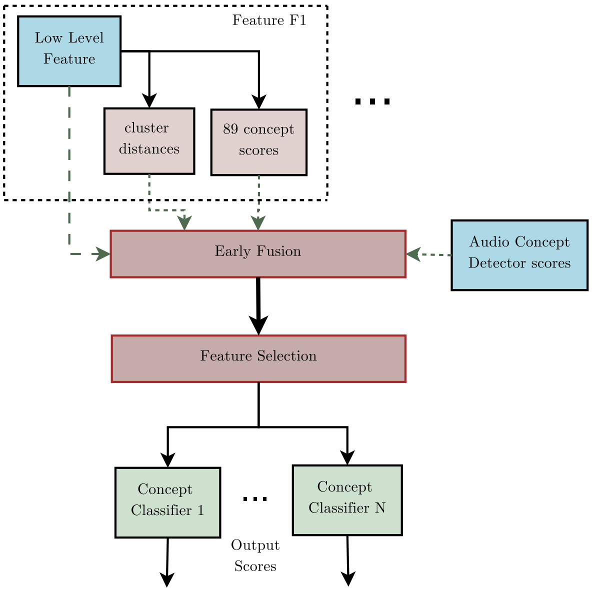

Figure 1: The various components used in the HLFE runs and the ways they interconnect. Different configurations occur by activating different dashed lines.

Concept score fusion for selected low level features (Run 4) Initially, the idea of using the 89 LSCOM concept scores is evaluated. The features used are R-FBANK, R-WBL, GL-CSD and L-KBSURF. The 89 predictor scores are evaluated for each feature and the resulting feature vectors are concatenated, leading to a feature vector of 356 values that is directly used for training the final high level feature models.

Concept score fusion with feature selection (Run 2) This run is similar to run 4. However, instead of manually selecting the original low level features to use, all low level features are selected initially (i.e., 9 × 89 = 801 dimensions). By employing feature selection with the same number of

[2] the audio concepts are *applause*, *cheering*, *crowd noise*, *explosion*, *laughter*, *silence*, *singing*, *speech*, *street noise*, *telephone noise*, *traffic*, *whistling*

features as the previous run (i.e., 356), these two runs can be compared.

Concept score fusion and distances from clusters with feature selection (Run 3) Just like in run 2, but instead of using only fusion of scores, cluster distances are also added. Again, feature selection leads from the original to 356 features, so that the results can be compared against the previous runs.

Concept score fusion, low level features and audio concept scores with feature selection (Run 1) In this run, all low level feature vectors and audio concept scores are concatenated and feature selection is applied to produce a 120-d feature vector. This feature vector is then concatenated with the scores used in run 2, forming a 476-d vector that is used in the final classification stage.

Concept score fusion for selected low level features, with train and test sets from different datasets (Run 5) In this mandatory type 'a' run the idea of run 2 is applied, but this time the training and test sets come from different datasets. More specifically, the TRECVID-2005 development set was split in two disjoint sets. The first set was used to train the models that compute the 89 concept scores in all experiments. These models were applied in the second set, that was used to train the final classifiers for the high level features. This allowed the creation of models only for 19 out of 20 high level features, since no annotations for "Two people" were found for the TRECVID-2005 development set.

4.1 HLFE results

The results of the HLFE task are summarized in Table 1. The scores are in terms of inferred average precision [25, 26], as provided by the TRECVID-2008 organizers [27]. The most important observations to be made are the following:

1. Compared to early fusion of low-level features and feature space clustering, concept score fusion appears to be the most efficient representation. InfAP attained in run 4, with manually selected type of score achieves results comparable to the better runs (runs 1 and 2).

2. Fusion with additional concept scores (i.e., originating from additional low level features) combined with feature selection improves the results. In fact, this approach (run 2) achieved the best results overall.

3. The use of additional information of low-level features and audio concept scores in run 1 does not improve the results. Actually, the results are worse than those of run 2. This can be attributed to the increase in the number of dimensions.

4. Feature space clusters do not provide good results. Further investigation is in order, so as to determine if this is due to the use of a non-representative set for the cluster formation, or due to the small number of clusters, or simply because this approach is not efficient.

5. For the type 'a' system (run 5), the poor results can be attributed to the fact that the system is trained with the same dataset that is used to create the 89 concept models. This leads to a bias towards the characteristics of that specific dataset and finally to poor generalization.

5 Concept Search

This section describes our experiments and submission to the search task. Since this year's test queries are created with the obvious goal to hamper a search solely based on automatic recognized speech (ASR), concept search becomes an important issue. With concept search we refer to the search on a collection of resources (here shots) that had been annotated by concept detectors with the probabilities for each concept to occur in a resource. Such a concept annotated collection would be the output of the TRECVID high-level feature extraction task.

Concept search typically comes with the problem that queries are formulated in search terms rather than in concepts. Hence, a mapping from the large term space to the typically quite restricted concept space, defined by the number of recognized concepts, is required. The basic idea of our concept search is to exploit the text descriptions of concepts to create an artificial text collection, where the concept annotations of a given shot are translated to text. Having such a text annotated collection, we can employ common text retrieval models to return a ranked list of shots. When building such an artificial text collection, concept pruning – reducing the dense shot concept matrix – becomes an important issue for this approach, which is investigated in our experiments.

No.	HL Feature	Run 1	Run 2	Run 3	Run 4	Run 5
1.	Classroom	0.0010	0.0040	0.0230	0.0110	0
2.	Bridge	0.0030	0.0030	0.0020	0.0040	0
3.	Emergency_Vehicle	0.0020	0.0210	0.0030	0.0020	0.0010
4.	Dog	0.0180	0.0240	0.0060	0.0040	0.0010
5.	Kitchen	0.0450	0.0620	0.0290	0	0.0060
6.	Airplane_flying	0	0.0780	0.0570	0.0690	0.0040
7.	Two people	0.0070	0.0290	0.0290	0.0250	0.0010
8.	Bus	0.0020	0.0010	0.0010	0.0010	0
9.	Driver	0.0280	0.0310	0.0330	0.0380	0.0040
10.	Cityscape	0.0600	0.0380	0.0350	0.0350	0.0100
11.	Harbor	0.0050	0.0030	0.0030	0.0030	0.0030
12.	Telephone	0.0170	0.0020	0.0070	0.0200	0.0010
13.	Street	0.0770	0.0650	0.0240	0.0220	0.0080
14.	Demonstration_Or_Protest	0.0020	0.0030	0.0020	0.0010	0.0010
15.	Hand	0.0460	0.0290	0.0330	0.0270	0.0160
16.	Mountain	0.0420	0.0540	0.0540	0.0520	0.0510
17.	Nighttime	0.0910	0.0950	0.0780	0.1260	0.0240
18.	Boat_Ship	0.0990	0.0810	0.0590	0.0740	0.0520
19.	Flower	0.0180	0.0390	0.0160	0.0150	0.0020
20.	Singing	0.0170	0.0140	0.0110	0.0100	0.0050
Mean InfAP		0.0290	0.0338	0.0253	0.0270	0.0095

Table 1: Overview of results for the High level feature extraction task.

Although we submitted runs to the high-level feature extraction task, our search task submission is not based on our own concept detector output. Instead we used the concept detector output from Columbia University (average fusion score [28]), which were kindly made publicly available.

5.1 Manually Assisted Query Formulation

The test queries are provided only as natural language queries. Since we are interested in investigating concept search rather than linguistic query analysis, we manually reduced the search topics to simple keyword queries to avoid further error sources that are not related to the tested retrieval method. As an example, the topic 226 “Find shots of one or more people with mostly trees and plants in the background no road or building visible” is translated to “people with trees and plants in the background”. The manual reduction followed a few guiding rules: (1) commands to the search engine as “find shots” are removed, (2) quantity measures such as “one or more” are removed, (3) negative statements as “no road or building” are removed, (4) and meta-descriptions of the shot such as “filling more than half of the frame area” are removed.

The provided machine translated queries in Dutch language were manually reduced to keyword queries following the same policy. In this case, we also corrected translation failures found in the machine translation, again with the aim to avoid failure sources outside the tested retrieval method.

5.2 From Concepts To Text

Related research on concept search addresses the mapping from term to concept space by translating the term query into a concept query. Aly et al. [29] are using textual concept descriptions to find the most related concepts to given a term query. They also improve the query mapping to concept space by enriching the textual concept description with the abstract of the concept’s Wikipedia article.

In contrast to these approaches, we do not translate the term query into a concept query, but the concept collection into an artificial text collection. The text collection is built by merging the textual context descriptions of all concepts belonging to a certain shot according to their detected probability of occurrence.

Textual concept descriptions are provided with the set of used concepts. We extend these descriptions as shown by Aly et al. [29] with the abstracts of those Wikipedia articles that match the concept name best.

When merging concept descriptions to an artificial textual description of a shot, we sum up term counts $Cnt_{t,s}$ according to the concepts' probabilities of occurrence $x_{c,s}$ in that shot:

$$Cnt_{t,s} = \sum_{c \in C} x_{c,s}\, Cnt_{t,c}. \tag{7}$$

To clarify the notation, $x_{c,s}$ will ideally be the probability of concept c given the low-level feature vector $\vec{F}$ of the shot. In practice, this detector score is a value in the range $[0, 1]$ but does not always represent a real probability. Whereas $Cnt_{t,s}$ denotes the calculated count of term t in the artificial shot text, $Cnt_{t,c}$ represents the term count of t in the concept description of c.

5.3 Concept Pruning

A concept annotated collection is typically represented by a dense shot concept matrix. For any given shot, we have detector scores for all concepts in the concept space C. This is in fact a major difference to text collections where each document is described only by the terms that occur in the document, but not by terms that do not occur. As a result of the dense matrix, our artificial text collection would become extraordinary large and the text search on an inverted index rather inefficient. In order to avoid these problems, we show here different techniques to prune the dense matrix with the aim to select per shot only those concepts having a high likelihood of occurrence.

Top k Selection This method simply selects for each shot the top k concepts having the highest detector score. All other concept scores for the shot are deleted from the score concept matrix.

Deviation Thresholding While the previously suggested top k selection is the obvious method when the detector scores come as real probabilities and hence represent values that are comparable across different concepts, in practice, this is seldom the case. Instead we find certain concept detectors delivering higher values than others even for concepts that are unlikely to occur often. Apart from noisy detector scores, a concept that occurs with a high likelihood in a large fraction of shots is less valuable for retrieval than concepts showing a more distinct distribution among shots.

In order to distinguish between the two cases, we analyze the distribution of the scores x_c of a certain concept over all shots, and calculate for each concept its expected value $\bar{x}_c$ and standard deviation σ_c. Instead of a top k selection, we determine now the set of most promising concepts for a given shot by a deviation-based threshold:

$$C_s = \{c \in C \mid (x_{c,s} - \bar{x}_c) > 2\sigma_c\}. \tag{8}$$

Combined Methods The deviation-based threshold can solve the above described problems with concept scores, but neglects the actual detector outcome to a large extent. We will find a number of concepts in the result set for a given shot that come with rather low score values. Assuming that the absolute detector values are not completely meaningless, we tried to combine both above suggested concept selection methods by selecting the top k concepts coming with the highest value $z_{c,s}$:

$$z_{c,s} = x_{c,s} \frac{(x_{c,s} - \bar{x}_c)}{\sigma_c}. \tag{9}$$

The new defined z value combines the actual score value x with its relative deviation.

5.4 Results and Conclusion

For the final text ranking on the generated text corpus we employed the PF/Tijah XML [30] retrieval system using the NLLR retrieval model. We will report here about the main evaluation measure being mean inferred average precision *infAP* [26]. An overview of the results is shown in Table 2.

The run named ASR is solely based on the automatic speech recognition output. For creating a shot-segmented text corpus, we concatenated the one-best output of the speaker segments provided by Huijbregts et al. [31] which overlapped with a given shot. On average 2.6 speaker segments overlapped with one shot.

Run TOP20 represents the top k selection approach using the 20 best scored concepts per shot, whose textual descriptions are merged as described above. The deviation-based thresholding is tested with run SIGMA2. Combined concept selection methods are represented by the runs DEVTOP20 and DEVTOP50, using the top 20, respectively top 50 highest rated concepts. The final run ASR-DEVTOP20 combines the relevance evidence of the two correspondingly named runs by adding their log-based scores.

The measured mean inferred average precision values remain rather low in all cases. We had expected low values for the ASR run, due to the queries asking for explicitly visual features. The

Run	*infAP*	*P@10*
ASR	0.0124	0.0596
TOP20	0.0167	0.0500
SIGMA2	0.0085	0.0208
DEVTOP20	0.0155	0.0458
DEVTOP50	0.0145	0.0354
ASR-DEVTOP20	**0.0238**	**0.0625**

Table 2: Result Overview

concept-based ranking, however, does not solve the shortcomings of pure ASR. Compared to other approaches, the retrieval quality of the proposed techniques remains around the median of all TRECVID runs.

Despite the low overall precision, the results indicate that the simple top k selection and the combined concept selection methods are more effective than the deviation-based thresholding. Furthermore, k should be set to a value smaller than 50. The combination of ASR and concept retrieval finally shows the best performance and hence supports the argument that video retrieval has to be cross media retrieval. Our suggested method of using an artificially generated text collection for concept retrieval comes here with the advantage to allow an easy integration of concept and ASR based ranking.

Acknowledgments

The authors are grateful to the Netmedia team of Fraunhofer IAIS and especially Jochen Schwenninger and Daniel Schneider for kindly providing the audio concept detector used in the experiments. Christos Diou is supported by the Greek State Scholarships Foundation.

References

[1] VITALAS, Integrated Project funded by the IST 6th Framework Programme of the European Commission, FP6-045389, visit `http://vitalas.ercim.org` for more information.

[2] B.S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Interface.* Wiley, 2002.

[3] T. Sikora. The mpeg-7 visual standard for content description-an overview. *IEEE Trans. on Circuits and Systems for Video Technology*, 11 (6):696–702, June 2001.

[4] J.-M. Geusebroek, R. van den Boomgaard, A.W.M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, December 2001.

[5] M Gervautz and W. Purgathofer. A simple method for color quantization: Octree quantization. In *New Trends in Computer Graphics.* Springer Verlag, Berlin, 1988.

[6] C. Diou, N. Batalas, and A. Delopoulos. Indexing and browsing of color images: Design considerations. In *Advances in Semantic Media Adaptation and Personalization*, volume 93 of *Studies in Computational Intelligence*, pages 329–346. Springer, 2008.

[7] Y. Rubner, C. Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, November 2000.

[8] J. C. van Gemert, J.-M. Geusebroek, C.J. Veenman, C.G.M. Snoek, and A.W.M. Smeulders. Robust scene categorization by learning image statistics in context. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop.* IEEE Computer Society, 2006.

[9] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America*, 7 (5):923–932, May 1990.

[10] T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, November 2001.

[11] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *In Proceedings of the 8th European Conference on Computer Vision*, pages 228–241, 2004.

[12] H. Baya, A. Essa, T. Tuytelaarsb, and L. Van Goola. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[13] D. Wang, X. Liu, L. Luo, J. Li, and B. Zhang. Video diver: generic video indexing with diverse features. In *MIR '07: Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 61–70. ACM, 2007.

[14] V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1989.

[15] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[16] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. Available: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] h-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2003.

[18] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[19] D. Wang, J. Li, and B. Zhang. Relay boost fusion for learning rare concepts in multimedia. In *Image and Video Retrieval*, volume 4071 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2006.

[20] M. Naphade, J.R. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE Multimedia*, 13(3):86–91, July-Sept. 2006.

[21] Lscom lexicon definitions and annotations version 1.0. DTO Challenge Workshop on Large Scale Concept Ontology for Multimedia, Columbia University ADVENT Technical Report, 217-2006-3, March 2006.

[22] K. Biatov and J. Köhler. Improvement speaker clustering using global similarity features. In *In Proceedings of the 9th International Conference on Spoken Language Processing, Interspeech 2006 - ICSLP*, Pittsburgh, PA, USA, September 17-21 2006. International Speech Communication Association.

[23] H. Peng, F. Long, and Ding C. Feature selection based on mutual information: Criteria of max-dependency, max-relevance and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1226–1238, August 2005.

[24] G Quénot and S. Ayache. Trecvid-2008 collaborative annotation. Available online at `http://mrim.imag.fr/tvca/`, 2008.

[25] E. Yilmaz and J.A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111. ACM, 2006.

[26] Javed A. Aslam and Emine Yilmaz. Inferring document relevance from incomplete information. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 633–642. ACM, 2007. ISBN 978-1-59593-803-9.

[27] A.F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, 2006.

[28] Yu-Gang Jiang, Akira Yanagawa, Shih-Fu Chang, and Chong-Wah Ngo. Cu-vireo374: Fusing columbia374 and vireo374 for large scale semantic concept detection. Technical Report 223-2008-1, Columbia University, 2008.

[29] Robin Aly, Djoerd Hiemstra, Arjen de Vries, and Franciska de Jong. A probabilistic ranking framework using unobservable binary events for video search. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 349–358. ACM, 2008. ISBN 978-1-60558-070-8.

[30] Djoerd Hiemstra, Henning Rode, R. van Os, and J. Flokstra. Pftijah: text search in an xml database system. In *Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR), Seattle, WA, USA*, pages 12–17. Ecole Nationale Supérieure des Mines de Saint-Etienne, 2006.

[31] Marijn Huijbregts, Roeland Ordelman, and Franciska de Jong. Annotation of heterogeneous multimedia content using automatic speech recognition. In Bianca Falcidieno, Michela Spagnuolo, Yannis S. Avrithis, Ioannis Kompatsiaris, and Paul Buitelaar, editors, *SAMT*, volume 4816 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2007. ISBN 978-3-540-77033-6.