Hierarchical Late Fusion for Concept Detection in Videos

Tiberius Strat, Alexandre Benoit, Hervé Bredin, Georges Quénot, Patrick Lambert

Hierarchical late fusion for concept detection in videos

Sabin Tiberius Strat[1,4], Alexandre Benoit[1], Hervé Bredin[2], Georges Quénot[3], and Patrick Lambert[1]

[1] LISTIC - Université de Savoie, Annecy, France,
http://www.polytech.univ-savoie.fr/index.php?id=listic-accueil
[2] Université Paris-Sud / CNRS-LIMSI, Orsay, France,
http://www.limsi.fr/
[3] Laboratory of Informatics of Grenoble, France,
http://www.liglab.fr/?lang=en
[4] IPAL - University "POLITEHNICA" of Bucharest,
http://alpha.imag.pub.ro/en/home_page.html

Abstract. We deal with the issue of combining dozens of classifiers into a better one, for concept detection in videos. We compare three fusion approaches that share a common structure: they all start with a classifier clustering stage, continue with an intra-cluster fusion and end with an inter-cluster fusion. The main difference between them comes from the first stage. The first approach relies on a priori knowledge about the internals of each classifier (low-level descriptors and classification algorithm) to group the set of available classifiers by similarity. The second and third approaches obtain classifier similarity measures directly from their output and group them using agglomerative clustering for the second approach and community detection for the third one.

Key words: late fusion, hierarchical, semantic concepts, video, semantic indexing

1 Introduction

Semantic indexing, as defined in the TRECVid evaluation campaign, consists in automatically detecting the presence of visual concepts in pre-segmented video shots [1] and returning a ranked list of shots the most likely to contain a given concept. Judging from the performance obtained by the best system in 2010 (with a mean inferred average precision on 30 concepts of 0.090), there is still a long way to go to solve this problem [2]. Some concepts appear to be much easier to detect than others and no single classifier emerges as *the one* that systematically (for any concept) outperforms the others. Therefore, for the sake of universality, many systems rely on the combination of a large (up to 100+) set of classifiers. They usually differ in the type of descriptors (color, texture, or bag of visual words, etc.) and/or in the machine learning algorithm (support vector machine or k nearest neighbors, for instance) they rely upon.

This paper focuses on the last step of this common semantic indexing pipeline: the late fusion of available classifiers. Let $\mathtt{K}$ be the number of classifiers and $\mathtt{N}$ the number of video shots.

Each classifier $k \in \{1 \dots \mathtt{K}\}$ provides scores $\mathbf{x}_k = [x_{k1}, \dots, x_{k\mathtt{N}}]$ indicating the likelihood for each shot $n \in \{1 \dots \mathtt{N}\}$ to contain the requested concept. The objective is to find a combination function $\mathbf{f}$ so that the resulting classifier $\mathbf{x} = \mathbf{f}(\mathbf{x}_1, \dots, \mathbf{x}_\mathtt{K})$ is better than any of its components, and as good as possible.

When looking for an effective combination of classifiers, several questions arise. Should we use them all in the fusion process, or just the best ones? Does combining two classifiers always yield better results than the two of them taken separately? Should we weigh them differently in case one is much better than the other? Tackling a similar problem, *Ng and Kantor* [3] proposed a method to predict the effectiveness of their fusion approach and concluded that "[...] *schemes with dissimilar outputs but comparable performance are more likely to give rise to effective naive data fusion*". There are multiple ways of measuring this similarity between two classifiers i and j. One of them is the Spearman rank correlation coefficient ρ_{ij}:

$$\rho_{ij} = \frac{\sum_{n=1}^{n=\mathtt{N}} (r_{in} - \overline{r_i})(r_{jn} - \overline{r_j})}{\sqrt{\sum_{n=1}^{n=\mathtt{N}} (r_{in} - \overline{r_i})^2 \sum_{n=1}^{n=\mathtt{N}} (r_{jn} - \overline{r_j})^2}} \tag{1}$$

where r_{kn} is the rank of shot n according to classifier k: $r_{kn} = 1$ (resp. $\mathtt{N}$) for the shot whose value x_{kn} is the maximum (resp. minimum). ρ_{ij} ranges from -1 (one ranking is the exact opposite of the other one) to 1 (rankings are identical). $\rho_{ij} = 0$ can be understood as classifiers being independent from each other.

Figure 1 uses a *spring layout* to represent this similarity measure for 90 classifiers trained for the concept *Computers*. Each classifier is represented by a node and similar classifiers (higher value of ρ_{ij}) are positioned closer to each other. It appears that some kind of community structure naturally emerges, with several groups of classifiers being more strongly connected internally than with the outside of their group. This is partly due to the low-level descriptors used internally by the classifiers (the type of descriptor is denoted by the shape of the nodes). For instance, classifiers based on color (circles) seem to agglutinate, as do classifiers based on audio features (diamonds). Finally, the size of items is directly proportional to the performance of the corresponding classifier. Therefore, best performing classifiers (i.e. larger items) also tend to agglutinate as they provide rankings that are closer to the reality – therefore closer to each other.

In this paper, we compare three fusion approaches that rely on these observations and share a common structure described in Figure 2. They all begin with a classifier clustering stage, continue with an intra-cluster fusion and end with an inter-cluster fusion.

Section 2 describes the first approach that relies on manually grouping classifiers of similar origin, in a hierarchic manner. The second and third approaches, described in Sections 3 and 4, group classifiers automatically according to their output scores, either iteratively in an agglomerative fashion, or based on a com-

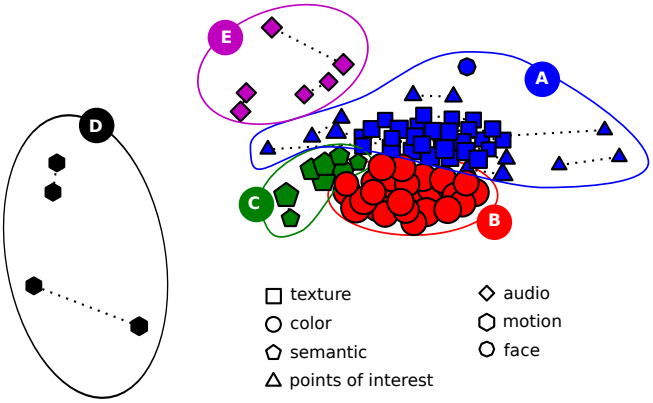

Fig. 1. Similarity of classifiers trained for detection of concept *Computers*. Each node represents a classifier, and edges represent the similarity between them (we only display some of the edges). The dotted edges represent classifiers which derive from the same descriptor, but have a different machine learning algorithm.

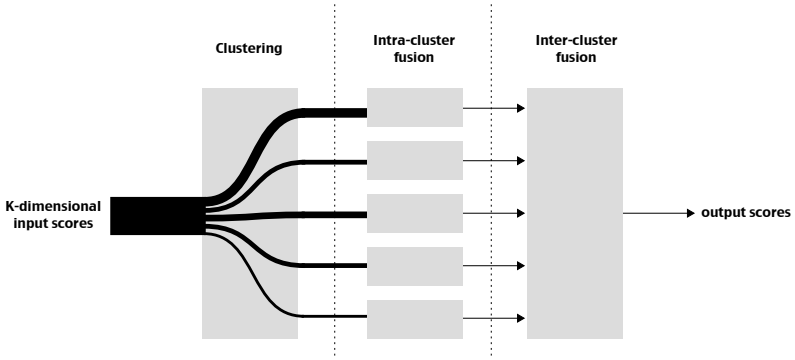

Fig. 2. Overview of the proposed late fusion approaches, sharing a common structure: a classifier clustering stage, an intra-cluster fusion and finally an inter-cluster fusion.

munity detection algorithm respectively. Experiments and results are described in Section 5.

2 Manual hierarchy

The manual hierarchy was designed according to a high level knowledge about the descriptors and the classifiers. The main principle considered is to fuse first descriptors or classifiers that are expected to be closer considering their nature or principle of operation. The manual hierarchy incorporates more levels than the automatic ones, with branches with different depths. In practice, we fused first the output of all the available machine learning algorithms for each descriptor (e.g. kNN and SVM). We then fuse different variants of the same descriptor (e.g. BOW of the same local descriptor but with different dictionary sizes). Afterwards, we fuse the classifiers corresponding to different image spatial decompositions (pyramid) if available. Finally, the last level concerns descriptors of different types within the same modality (e.g. color, texture, interest points, percepts or faces) and descriptors from different modalities (audio and visual).

Various experiments with manually defined hierarchies suggested that going from the most similar to the most different was a good strategy. These experiments also showed that the best results are obtained when using as many combinations as possible of descriptors and machine learning algorithms. Even combinations with low performance can contribute to a global performance increase, especially if they are complementary to better ones.

Late fusion was performed at all levels using a weighted arithmetic mean of normalized scores. Several other and more complex methods were tried but produced no or very small improvements. Three weighting strategies were considered: uniform (simple arithmetic mean), MAP based (simple function of the MAP of the different inputs), and direct optimization by cross-validation. Cross-validation experiments showed that in the early stages, uniform weighting was preferable for robustness while in latter stages MAP-based or directly optimized weighting provided better results.

3 Agglomerative clustering

This fusion approach automatically filters out irrelevant classifiers, then it groups highly-correlated ones in an iterative manner. The target semantic concepts are treated individually, meaning that each semantic concept may generate different groupings. The method consists of the following steps:

1. determine the individual relevance of each classifier for the target concept. The relevance is taken as the average precision α of the classifier for the target concept on the training dataset, normalized by the proportion of true positives in the training dataset.
2. retain only classifiers with a relevance higher than 1 (better than random classification). Additionally, the classifiers must have at least 1/8th of the

relevance of the best one, so as not to "pollute" the good classifiers with bad ones.
3. Some of the retained classifiers are highly correlated, so we look for the pair with the maximum correlation and fuse it into a single classifier. We update the correlation between the resulting classifier and the remaining ones.
4. The previous step is repeated many times, until a sufficiently correlated pair can no longer be found. This has a dimensionality-reduction effect and also helps to reduce the classification "noise".

The correlation measure used is the correlation coefficient of the raw classification scores. We consider a pair of attributes as correlated if *(a)* the correlation coefficient for all video shots is at least 0.75, to ensure that the two classifiers give similar information on a global scale, and *(b)*, the correlation between the scores for just the positive shots must be at least 0.65, to ensure that the positives tend to be classified in the same way. We add the second constraint because in TRECVid, most of the target concepts have very few positives, and otherwise, the classification scores on the negatives would dominate the correlation.

Now, the resulting classifiers are again filtered based on their average precision on the training set, using the same criteria as before. Afterwards, this approach separates into two versions, which are detailed below.

First version: weighted average The individual relevances of each remaining classifier are used as weights for a weighted arithmetic mean, thus obtaining the final classification score. We have not used this version in the official TRECVid 2011 submissions, because previous tests have indicated that the second version should be better (however with a different performance metric).

In the end, because all the previous steps consisted of selections and averaging, without any normalisation operations as those in section 4, this approach is in fact a weighted arithmetic mean, with a more elaborate way of choosing the weights.

Second version: PCA and nearest-neighbors In this version, we continue with a Principal Component Analysis (PCA) to further eliminate correlation and reduce the dimensionality of the problem. We retain the first 1, 2, 3, 4 and 5 most important dimensions, and on each of these choices independently, we apply a neighborhood-based fusion strategy, as follows: for a test shot, we count the positives and negatives from the training base in a volume of radius d around it. This radius is taken as an average distance between training shots. If no neighbors are present within the volume, we consider the shot as a positive, based on the assumption that because positives are generally much rarer than negatives, they are unable to densely cover their entire corresponding space.

Because experiments on cross-validation have shown that we cannot predict the optimal number of dimensions to take after PCA, the last step consists in averaging the classification scores obtained with 1, 2, 3, 4 and 5 dimensions.

This version has been used to generate two official submissions for TRECVid 2011, the details of which will be given in Section 5.

4 Community detection

This last fusion approach is very similar to the previous one. The main difference relies in the way classifier clusters are discovered.

We define the agreement $A_{ij} = \max(0, \rho_{ij})$ between two classifiers i and j, where ρ_{ij} is the Spearman rank correlation coefficient given in equation 1. A complete undirected graph $\mathcal{G}$ is constructed with one node per classifier. Each pair of classifiers (i, j) is connected by an undirected edge, whose weight is directly proportional to A_{ij}. A simplified representation of such a graph is given in Fig. 1.

We rely on the so-called *Louvain* approach for automatic community detection proposed by *Blondel et al.*, and apply it on graph $\mathcal{G}$. It is a heuristic method that is based on the maximization of modularity $\mathcal{Q}$:

$$\mathcal{Q} = \frac{1}{\sum_{i,j} A_{ij}} \sum_{i,j} \left[A_{ij} - \frac{\sum_k A_{ik} \sum_k A_{kj}}{\sum_{i,j} A_{ij}} \right] \delta_{ij} \tag{2}$$

where $\delta_{ij} = 1$ if classifiers i and j are members of the same community, 0 otherwise. $\mathcal{Q}$ can be seen as a measure of the quality of the detected communities. It increases when communities have stronger intra-community and weaker inter-community edges [4]. For a detailed description and analysis of the algorithm, we refer the interested reader to [5].

With no objective groundtruth to compare with, it is difficult to evaluate the detected communities. However, looking at Fig. 1 and the five detected communities (A to E), it seems that the *Louvain* algorithm did a good job at finding communities related to the nature of the low-level descriptors on which classifiers are based. In particular, a dotted edge between a pair of classifiers indicates that they are based on the very same descriptors and they only differ in the machine learning algorithm they rely on. None of these pairs is split into two different communities.

After the clustering stage, classifiers from each community are combined by simple sum of normalized scores, in order to obtain one new classifier per community. The normalization strategy is presented further below. These new classifiers are expected to be at least as good as the best of their components and can sometimes lead to much better performance.

Since they come from different communities, these new *community classifiers* are very likely to output very dissimilar scores and rankings. They are combined using a weighted sum fusion of normalized scores:

$$\mathbf{x} = \sum_{c=1}^{c=\mathtt{C}} \alpha_c \widehat{\mathbf{x}_c} \tag{3}$$

where the weights α_c are in fact the average precisions of each of these new *community classifiers*. They are estimated using a development set.

Both fusion steps rely on normalized scores. We investigated multiple normalization techniques (min/max, σ/μ, TanH) but only report on the one that proved to be the best, TanH normalization [6]:

$$\widehat{x_{kn}} = \frac{1}{2}\left\{\tanh\left[0.01\left(\frac{x_{kn}-\mu_k}{\sigma_k}\right)\right]+1\right\} \quad (4)$$

where μ_k and σ_k) are respectively the mean and standard deviation of scores provided by classifier k on test set.

5 Experiments

The fusion approaches are tested on the TRECVid 2011 dataset. We train our algorithms on the official development collection, and we evaluate the performances on the official test collection.

5.1 Input classifiers

The input classification scores are obtained by applying supervised classification algorithms on multidimensional descriptors extracted from the video shots. We use a battery of multidimensional descriptors such as various color histograms, Gabor transforms, spatio-temporal interest points, SIFT or SURF Bag-of-Words, face tracks, presence of mid-level semantic concepts, audio spectral profiles, histograms of Local Binary Patterns etc. Most of the descriptors have several versions, obtained by varying the extraction parameters (such as the number of bins when performing clustering, the spatial pyramid decomposition etc.).

Additionally, we apply power transformations on the multidimensional descriptors in order to optimize the data distribution before doing the supervised classification. We selected an optimal power coefficient for each descriptor by cross-validation on the training set.

The supervised classification algorithms are either a k-Nearest Neighbor (KNN) or a multiple Support Vector Machine (MSVM). By combining descriptors with different power transformations with the supervised classifiers, we obtain sets of classification scores which we call *KNNG, KNNC, KNNB, MSVM* and *ALLC*.

The KNN scores are relatively fast to compute, as the nearest-neighbors need to be determined only once for all target semantic concepts. This is important, because finding neighbors in a large collection and with many dimensions is a computationally-expensive operation. Only the counting of positives and negatives among neighbors needs to be done for each concept individually, but this is trivial. The optimization of the KNN hyper parameters can be done either at the individual target concept level (KNNC) or globally (KNNG). By partinioning the development set, the latter was found to be more robust and the late fusion of both (KNNB) was found to be better than both in almost all cases.

The MSVM classifiers are generated only using the optimal power transformation for the 49 available descriptors. ALLC, being the average of KNNB

and MSVM, also has 49 classifiers, being slightly better than both KNNB and MSVM. However, because the MSVM supervised classification is much more computationally-demanding (and because it needs to be done for each concept separately), only a subset of MSVM and ALLC scores was available for the official submissions of TRECVID.

5.2 Submissions

The official submissions consist of a *Manual hierarchy fusion* with all the available classifiers, two *Agglomerative clustering fusions (second version)*, one applied on the KNNB and one on the KNNC classifiers, and a *Community detection fusion* applied on the KNNC/KNNG sets. Additionally, we apply a video shot re-ranking strategy based on temporal coherence, which further increases performances. The MSVM and ALLC classifiers, even though they are better than KNN, were only used for the *Manual hierarchy fusion*, because they were not all ready before the submission deadline.

Because the number of permitted official submissions per team was limited to 5, we extend our study with unofficial experiments. For these, all the MSVM and ALLC scores are now available for input.

As a reference for comparison, we take, for each semantic concept individually, its best classifier in cross-validation on the training set. We complement this reference with the arithmetic mean, either with equal weights for all classifiers, or weighted for each semantic concept individually by the average precision obtained by a classifier for that concept. We also experimented in earlier stages with a geometric mean of the input classifiers and with a rank fusion, but these last two approaches give very similar results to the arithmetic mean.

5.3 Results and comparison

The results obtained by the various fusions are summarised in Table 1.

Among our official submissions, the *Manual hierarchy fusion* is the best. This is partly due to the fact that manually grouping classifiers obtained in a similar manner ensures more homogeneous properties within a group. The rest of the performance increase of the manual hierarchy is due to the inclusion of some of the MSVM and ALLC classifiers (which are better than KNN), even if they were not all available at that time. This submission ranked 8th among all submissions from all participants in the task.

The *Community detection* applied on the KNNC/KNNG sets ranked 15th and 25th (with and without temporal re-ranking respectively) in the official hierarchy, while the *Agglomerative clustering fusions (second version)* ranked 25th for KNNB and 30th for KNNC, which situates it close to the middle of the official hierarchy.

Looking also at the unofficial experiments, a first thing to notice is that all of the methods outperform the *Best classifier* baseline, if they are applied on the same dataset and all of them either use, or do not use, the temporal re-ranking.

Table 1. Performance of fusion approaches. We display the Mean Average Precisions (mAvgPrec), and also the rankings of the official submissions among all Semantic Indexing task participants. The first four rows are official TRECVid submissions, which did not benefit from the complete MSVM and ALLC sets of classifiers. The other rows are unofficial experiments, when the MSVM and ALLC sets were complete.

Fusion	Applied on	mAvgPrec	+ re-rank
Manual hierarchy	ALLC	0.1454	0.1529 (#8)
Agglomerative clustering (v2)	KNNB		0.1194 (#25)
Agglomerative clustering (v2)	KNNC		0.1142 (#30)
Community detection	KNNC/KNNG	0.1341 (#17)	0.1387 (#15)
Best classifier	KNNB		0.1146
Best classifier	ALLC	0.1178	0.1332
Arithmetic Mean	KNNB		0.1381
Arithmetic Mean	ALLC	0.1415	0.1481
Weighted Mean	ALLC	0.1419	0.1491
Agglomerative clustering (v1)	KNNB		0.1423
Agglomerative clustering (v1)	ALLC	0.1457	0.1520
Agglomerative clustering (v2)	ALLC	0.1332	0.1444
Community detection	ALLC	0.1438	

The *Community detection* using KNNC/KNNG outperforms the *Best classifier* on KNNB (KNNB is the fusion of KNNC and KNNG) by a good margin of 21% if using the temporal re-ranking, while the *Agglomerative clustering (v.2)* of KNNB achieves a small margin of 4%.

Regarding the addition of temporal re-ranking, we do not display the average precision for all methods, but we confirm that it improves the results for all fusions. The performance boost is especially obvious for the *Best classifier* approach, where, using the ALLC input scores, we increase the average precision by 13%.

For the automatic clustering approaches applied on ALLC, the *Agglomerative clustering (first version)* is the best, outperforming the *Best classifier* by 23%, followed by the *Community detection* (22%), and finally by the *Agglomerative clustering (second version)* (13%). However, the performance of the arithmetic mean, either simple or weighted, is not to be ignored. The margins by which the *Agglomerative clustering (v.1)* and the *Community detection* outperform the simple arithmetic mean are much lower: 3% and 1.6% respectively, while the *Agglomerative clustering (second version)* actually performs worst. The weighted arithmetic mean, with weights given by the individual relevance of attributes, gives very similar results.

Considering the simplicity of the arithmetic mean, the processing time constraints need to be taken into account when deciding whether or not the performance boost of a more complex fusion method is worth the effort. Already, the simple arithmetic mean manages to improve on the *Best classifier* by a margin of 20%.

6 Conclusion and future work

In this paper, we proposed several ways of combining dozens of input classifiers into better ones, and applied them in the context of the *TRECVid 2011 Semantic Indexing* task. We have shown that all of the methods outperform taking the best classifier for each concept, and they are all better than an arithmetic mean, except the second version of agglomerative clustering, which was optimized based on different criteria. The performance boost of the more complex methods however, needs to be balanced with the computational complexity, as the arithmetic mean is very close in performance.

In the future, we plan to experiment with combining the three fusion approaches presented, and also using various score normalisation strategies at different levels of the algorithm.

Acknowledgments This work was supported by the Quaero Program and the QCompere project, respectively funded by OSEO (French State agency for innovation) and ANR (French national research agency). The authors would also like to thank the members of the IRIM consortium for the classifier scores used throughout the experiments described in this paper.

References

1. Smeaton, A.F., Over, P., Kraaij, W.: High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements. In Divakaran, A., ed.: Multimedia Content Analysis, Theory and Applications. Springer Verlag, Berlin (2009) 151–174
2. Snoek, C.G.M., van de Sande, K.E.A., de Rooij, O., Huurnink, B., Gavves, E., Odijk, D., de Rijke, M., Gevers, T., Worring, M., Koelma, D.C., Smeulders, A.W.M.: The MediaMill TRECVID 2010 Semantic Video Search Engine. In: Proceedings of the 8th TRECVID Workshop, Gaithersburg, USA (2010)
3. Ng, K.B., Kantor, P.B.: Predicting the Effectiveness of Naive Data Fusion on the Basis of System Characteristics. Journal of the American Society for Information Science **51** (2000) 1177–1189
4. Newman, M.E.J.: Modularity and Community Structure in Networks. Proceedings of the National Academy of Sciences of the United States of America **103** (2006) 8577–8582
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast Unfolding of Communities in Large Networks. Journal of Statistical Mechanics: Theory and Experiment **2008** (2008) P10008
6. Ross, A.A., Nandakumar, K., Jain, A.K.: Handbook of Multibiometrics (International Series on Biometrics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)