Latent Semantic Indexing for Video Content Modeling and Analysis

Fabrice Souvannavong, Bernard Merialdo and Benoît Huet
Département Communications Multimédias
Institut Eurécom
2229, route des crêtes
06904 Sophia-Antipolis - France
(Fabrice.Souvannavong, Bernard.Merialdo, Benoit.Huet)@eurecom.fr

Abstract

In this paper we describe our method for feature extraction developed for the Video-TREC 2003 workshop. Latent Semantic Indexing (LSI) was originally introduced to efficiently index text documents by detecting synonyms and the polysemy of words. We successfully proposed an adaptation of LSI to model video content for object retrieval. Following this idea we now present an extension of our work to index and compare video shots in a large video database. The distributions of LSI features among semantic classes is then estimated to detect concepts present in video shots. K-Nearest Neighbors and Gaussian Mixture Model classifiers are implemented for this purpose. Finally, performances obtained on LSI features are compared to a direct approach based on raw features, namely color histograms and Gabor's energies.

Keywords: *Latent Semantic Indexing, Video Content Analysis, Gaussian Mixture Model, Kernel Regression*

1 Introduction

With the growth of numeric storage facilities, many documents are now archived in huge databases or extensively shared on the Internet. The advantage of such mass storage is undeniable, however the challenging tasks of content indexing and retrieval remain unsolved, especially for video sequences, without the expensive human intervention. Many researchers are currently investigating methods to automatically analyze, organize, index and retrieve video information [1, 7]. This effort is further underlined by the emerging Mpeg-7 standard that provides a rich and common description tool of multimedia contents. It is also encouraged by Video-TREC which aims at developing and evaluating techniques for video content analysis and retrieval.

One Video-TREC task focuses on the detection of high-level features in video shots; such features include *outdoors, news subject, people, building, ...* To solve this problem, we propose to model the video content with Latent Semantic Indexing. Then based on these new features, we train two classifiers to finally detect semantic concepts. Performances of the K-Nearest Neighbors and Gaussian Mixture Models classifiers are compared and provide a framework to evaluate the efficiency of Latent Semantic Indexing for video content modeling.

Latent Semantic Analysis was proven effective for text document analysis, indexing and retrieval [2] and some extensions to audio and image features were proposed [4, 9]. In [8], we have introduced LSA to model a single video sequence for enhanced navigation. This article extends our previous work to model and compare video shots in a large video database. Contrary to single video modeling, the diversity of the content requires specific adaptations to correctly model video shots.

The next section introduces the Latent Semantic Indexing conjointly with methods to improve performances,

i.e. combination of color and texture information and better robustness. Then, K-Nearest Neighbors and Gaussian Mixture Model classifiers are presented in this context. Next, their performance and the efficiency of LSI are discussed through experimental results. Finally, we conclude with a summary and future work.

2 Video Content Modeling

In order to efficiently describe the video content, we decided to borrow a well-known method used for text document analysis named Latent Semantic Indexing [2]. First we detail the adaptation of LSI to our situation and then propose methods to include multiple features and to improve the robustness of LSI in our particular case, i.e modeling of video shots in a large database.

Latent Semantic Indexing (LSI) is a theory and method for extracting and representing the contextual meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other [5]. In practice, we construct the occurrence matrix A of words into documents. The singular value decomposition of A gives transformation parameters to a singular space where projected documents can efficiently be compared.

For video content analysis, a corpus does not naturally exist, however one can be obtained thanks to vector quantification technics. In [8], we presented an approach on single video sequences that relies on k-means clustering to create a corpus of frame-regions. Basically, key-frames are segmented into regions [3] and each region is represented by a set of features like color histogram and Gabor's energies. They are then mapped into a codebook, obtained with the k-means algorithm, to construct the co-occurrence matrix A of codebook elements in video key-frames. Thus each frame is represented by the occurrence of codebook terms. LSI is then applied to the matrix A and provides projection parameters U into a singular space where frame vectors are projected to be indexed and compared. This can be extended to model a set of video sequences; the set can be seen as a unique video where key-frames are the representative frames of shots.

Mathematical operations are finally conducted in the following manner:

- First a codebook of frame-regions is created on a set of training videos,
- The co-occurrence matrix is constructed:
  Let A of size M by N be the co-occurrence matrix of M centroids (defining a codebook) into N key-frames (representing the video database). Its value at cell (i, j) corresponds to the number of times the region i appears in the frame j.
- Next, it is analyzed through LSA:
  The SVD decomposition gives $A = USV^t$ where

$$UU^t = VV^t = I,\ L = \min(M,N)$$

$$S \approx diag(\sigma_1,..,\sigma_L),\ \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_L$$

  Then A is approximated by truncating U and V matrices to keep k factors in S corresponding to the highest singular values.

$$\hat{A} = U_k S_k V_k^t \text{ with } S_k = diag(\sigma_1,..,\sigma_k)$$

- Finally, indexing of a context of A noted $c(j)$ and a new context q is realized as follows:

$$p_{c(j)} = \text{row j of } VS$$

$$p_q = q^t U_k$$

- And to retrieve the context q in a database containing indexed contexts p_j, the cosine measure m_c is used to compare elements.

$$m_c(p_j,q) = \frac{p_q . p_j}{||p_q||.||p_j||}$$

  The most similar elements to the query are those with the highest value of m_c.

The number of singular values kept for the projection drives the LSA performance. On one hand if too many factors are kept, the noise will remain and the detection of synonyms and the polysemy of visual terms will fail. On the other hand if too few factors are kept, important

information will be lost degrading performances. Unfortunately no solution has yet been found and only experiments allows to find the appropriate factor number.

In the particular situation of video content, many features can be extracted. Three methods were evaluated to consider multiple features in [8]. They are combined at the origin, before the creation of the codebook, or independent codebooks are merged to create a single occurrence matrix, or the LSI is applied to each feature and the similarity measure is modified to combine outputs from each singular space. In [8], we retained that equivalent performances were obtained when features were combined just before or after LSI. The latter solution being the most flexible is kept for our task. Indeed features can easily be weighted and new features added without the need to do all computation tasks again.

Contrary to the modeling of a single video content, LSI does not reveal as performant for many videos. The occurrence information in each frame is too weak compared to approximations inherent to the the use of a codebook and this effect is further emphasized when many videos are implied. To compensate for codebook instability, we match a region to its k-nearest elements in the codebook. This one-to-many relationship allows to inject more occurrence information for each key-frame and to deal with the sub-optimality of the codebook. We observe a real improvement when looking for similar frames in the database.

3 Feature Detection

We focus our attention on general models to detect Video-TREC features and given our selected visual features, i.e color and texture, we do not expect to succeed for all of them. In particular, color and texture information are not sufficient and well adapted to detect *female speech, monologue* and *zoom in*. We propose two classifiers namely Gaussian Mixture Models and K-Nearest Neighbors classifiers for which input features are the projected vectors of color and texture features in their respective singular space. 200 factors out of 500 and 1500 where kept for projections. Classifiers are trained to recognize the 133 items from the IBM tool [6]. Then detection scores are merged according to the target Video-TREC feature, this point is discussed after the presentation of classifiers.

Let assume that the distribution of Video-TREC features can be modeled by mixtures of Gaussians. The Mahalanobis distance remains valid to compare projected vectors when they are normalized. The classical Expectation-Maximization algorithm trains mixture of ten Gaussians assumed to have a diagonal covariance. The detection score of shots containing a feature, denoted F_x, is then based on the likelihood value computed on the corresponding mixture. Another solution consists in training two mixtures for each feature, one for positive samples (P_p), i.e. that contains F_x and one for negative samples (P_n), i.e. that does not contain F_x. We then compute a detection score as:

$$Ds(shot_i) = P_p(shot_i)/[P_p(shot_i) + P_n(shot_i)]$$

Since we have no information about the distribution shape of the data, we find natural to compare the performance of GMM with K-NN. Given a shot i, its 20 nearest neighbors in the training set are identified. Then it inherits a detection score as follows:

$$Ds(shot_i) = \sum_{k=0}^{k=20} sim(shot_i, shot_k) * Ds(trshot_k)$$

Where detection scores of training shots, $trshot_k$, are either 1 if F_x was annotated or 0 if not.

Detection scores of IBM's features are then weighted and summed according to the Video-TREC features to find. For simplicity only weights in $\{-1, 0, +1\}$ are possible. For example the score for the feature *vegetation* is the sum of scores obtained on *nature vegetation* and its children. Given Video-TREC and id's of IBM's items, tabular 1 provides a summary of the mapping between features.

4 Experiments

Our submission to Video-TREC included 6 runs to compare both classifiers and the effect of LSI over raw features. Figure (5) shows the evaluation result (only experiment results of KNN and GMM trained with positive and negative samples are shown). Results presented in this paper differ from the one submitted for two reasons. The training set used for the submission was composed of the

first half part of the complete development set. And only texture information was included when dealing with K-NN.

Figure (4) presents the performances of Gaussian Mixture Models for the detection of Video-TREC features using LSA features. Modeling positive and negative classes significantly improves detection capacities. It reveals the complexity and diversity of the content repartition in classes. We also encountered the problem of data starving during the training of many features and especially when 20 mixtures were trained. This explains the decrease of performances for 20 mixtures compared to 10 mixtures. We conclude that Gaussian Mixtures are not adapted in this context. The complexity of the content requires many mixtures whom the necessary amount of training data is not available. Figure (4) presents the same experiments when mixtures are trained on raw features.

Figure (4) shows better performances. Representing the shot content with count vectors instead of raw features performs better. And more improvement is obtained when using LSA on count vectors. We have surprisingly good results with feature 8, i.e. *female speech*. It is explained by the fact that shots of the test set annotated with *female speech* are mostly *female news person* shots and most *news person* shots in the development set are *female news person* shots. In that case, visual features are sufficient to achieve good performance.

5 Conclusion and Future Work

We presented Latent Semantic Indexing to efficiently model video contents. It gives an efficient representation of key-frame (thus shot) content. However the proposed adaptation relies on the creation of a codebook, operation that is often sub-optimal. To overcome this problem, we introduced a method that improves approximations robustness by matching a frame-region to k codebook elements. We then used LSI features to train two classifiers: Gaussian Mixture Model and K-Nearest Neighbors, the first models semantic classes with mixture of Gaussians whereas the second makes any assumption about feature distribution in classes. Finally classifiers were compared and used to evaluate the gain obtained with LSI.

Future work will take several directions. One disadvantage of Latent Semantic Indexing, as presented, is the loss of spatial information. Thus, efforts will be conducted to include spatial relationship between regions. On the other hand, we do not take advantage of the whole video content. New features will be included such as object and camera motion, text and audio. Moreover a shot could be represented by all its frames instead of only its key-frames.

References

[1] Shih-Fu Chang, W. Chen, H.J. Meng, H. Sundaram, and Di Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 8, pages 602– 615, 1998.

[2] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[3] P. Felzenszwalb and D. Huttenlocher. Efficiently computing a good segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 98–104, 1998.

[4] Mikko Kurimo. Indexing audio documents by using latent semantic analysis and som. In Erkki Oja and Samuel Kaski, editors, *Kohonen Maps*, pages 363–374. Elsevier, 1999.

[5] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[6] Ching-Yung Lin, Belle L. Tseng, and John R. Smith. Video collaborative annotation forum: Establishing ground-truth labels on large multimedia datasets. In *Proceedings of the TRECVID 2003 Workshop*, 2003.

[7] M.R. Naphade, T. Kristjansson, B. Frey, and T.S. Huang. Probabilistic multimedia objects (multijects): a novel approach to video indexing and retrieval. In *IEEE International Conference on Image Processing*, volume 3, pages 536–540, 1998.

TREC	outdoors	news subject	people	building	road	vegetation
IBM	49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73	100 104	106	70	72	50 51 52 53 54
TREC	animal	female speech	car-truck-bus	aircraft	monologue	non studio setting
IBM	84 85 86 87 88 89 90	101 102 103 104	125 128 129	122		39 40 41 42 43 44 45 46 47 48
TREC	sporting event	weather	zoom in	physical violence	Madeleine Albright	
IBM	13 14 15 16 17 18 19 20 21	30		31	102	

Table 1: Mapping between Video-TREC features and IBM's items.

[8] Fabrice Souvannavong, Bernard Merialdo, and Benoˆ Huet. Video content modeling with latent semantic analysis. In *Third International Workshop on Content-Based Multimedia Indexing*, 2003.

[9] Rong Zhao and William I Grosky. From features to semantics: Some preliminary results. In *International Conference on Multimedia and Expo*, 2000.

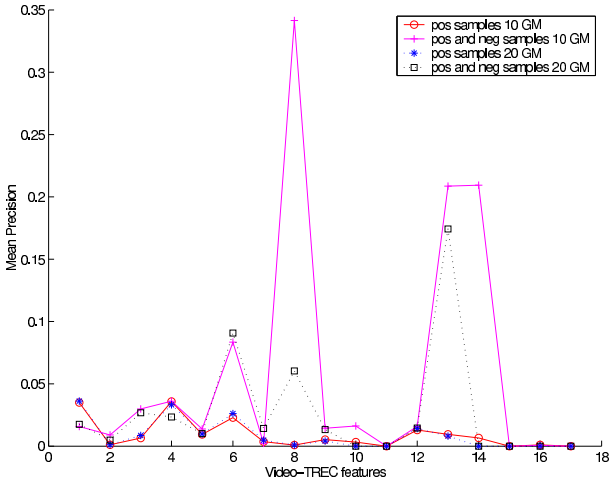


Figure 1: Gaussian Mixture Model on LSA features.

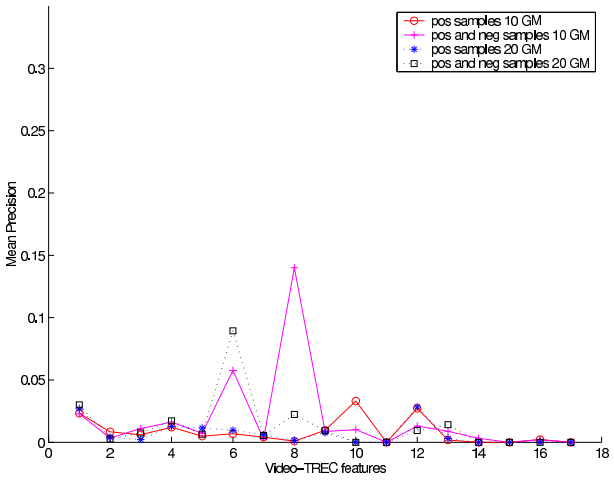


Figure 2: Gaussian Mixture Model on raw features.

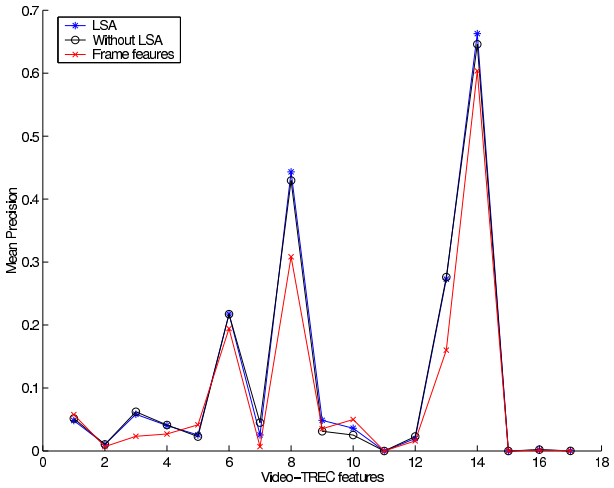
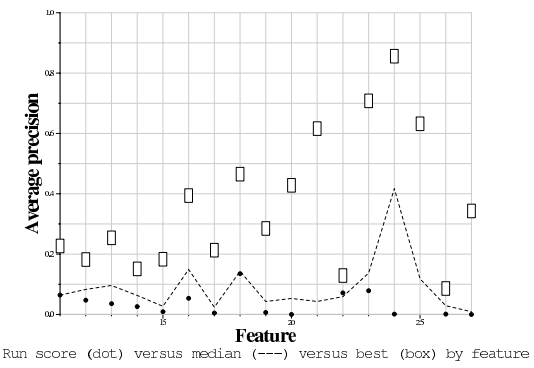
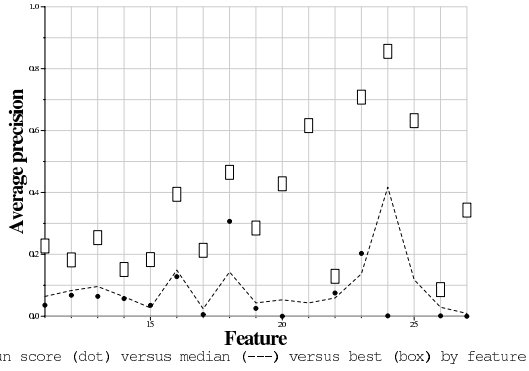


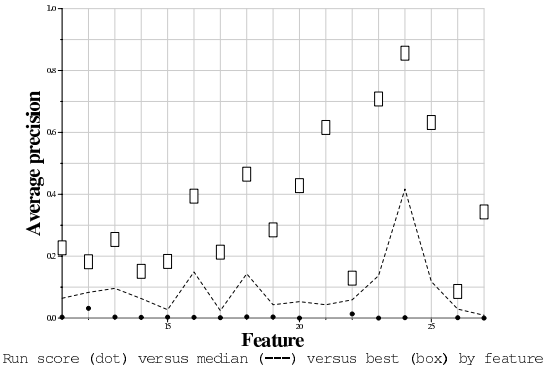
Figure 3: K-Nearest Neighbors.



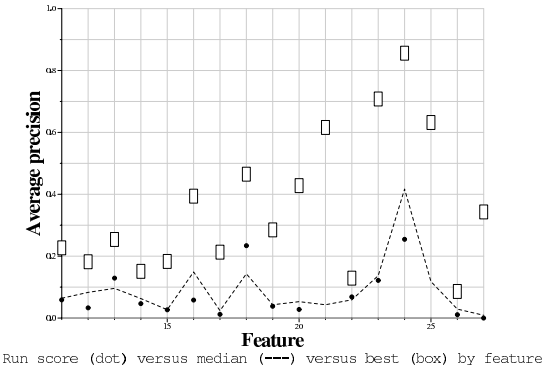
(a) GMM trained on raw features



(b) GMM trained on LSA features



(c) KNN "trained" on raw features



(d) KNN "trained" on LSA features

Figure 4: Results of runs submitted to Video-TREC 2003.