Unsupervised Audio Analysis for Categorizing Heterogeneous Consumer Domain Videos

Pradeep Natarajan, Stavros Tsakalidis, Vasant Manohar, Rohit Prasad, Prem Natarajan

Speech, Language, and Multimedia Business Unit
Raytheon BBN Technologies
Cambridge, MA 02138
{pradeepn,stavros,vmanohar,rprasad,pnataraj}@bbn.com

Abstract

The ever increasing volume of consumer domain videos on the Internet has led to a surge in interest in automatically analyzing such content. The audio signal in these videos contains salient information, but applying current automatic speech recognition (ASR) techniques is not viable due to high variability, noise and multilingual content. We present two unsupervised techniques which do not rely on ASR to address these challenges. The first method involves learning an unsupervised codebook by clustering audio features, and the second involves directly matching low-level features using the pyramid match kernel (PMK). Experimental results on a ≈200 hour audio corpus downloaded from YouTube show that both our approaches significantly outperform the traditional approach of first segmenting the audio stream to a set of mid-level classes (e.g. speech, non-speech, music, silence) and using the duration statistics of these classes to train high-level classifiers.

Index Terms: Web Audio Retrieval, Pyramid Match Kernel, Audio Codebook Learning

1. Introduction

The ability to automatically search through large volumes of web videos and summarize their content has several applications including retrieval, copy detection and intelligence analysis. Many web videos contain a complementary stream of information in the accompanying audio track. Due to its lower dimensionality compared to the video track, and its relative invariance with respect to video quality and viewpoint, audio provides a powerful mechanism for video query and retrieval, as well as for the recognition of complex actions that may be difficult, or even impossible to perform using video alone. Extensive research has been conducted over many years for analyzing the audio stream of video sources. However, developing such capabilities is complicated due to the unstructured, noisy and sometimes irrelevant information of the audio content.

A common approach is to convert the audio stream into words or phonemes using a speech recognition system and then use text-based topic identification techniques [1, 2, 3]. Another approach [4] is to train dedicated ergodic Hidden Markov Models (HMM) for each type of content in the target audio collection (e.g. commercials, news reports, sports). However, under noisy and mismatched audio conditions, the recognition performance diminishes for such approaches. Furthermore, various audio events (e.g. music, applause, car-engine) cannot be detected by a traditional speech recognition system.

An alternative is to detect events based on cues in the audio stream. Numerous techniques have been proposed to address this problem. A suite of techniques [5, 6] based on Support Vector Machine (SVM) and HMM based classifiers have been developed to detect basic audio events (e.g. silence, music, background sound, pure speech, and non-pure speech). Techniques that detect more detailed audio cues (e.g. car-engine, sirens, applause) have also been proposed [7, 8, 9, 10]. While these approaches are encouraging, they require labeled examples for supervised training which are challenging to collect. Moreover, these techniques are only capable of detecting a predefined, closed set of events.

Due to the highly heterogeneous nature of the audio source in web videos a fully unsupervised approach is more suitable. Towards this end, fully unsupervised techniques have been developed for topic classification [11]. The topic classifier is built using an unsupervised Segmental Gaussian mixture model (SGMM) tokenizer, which performs tokenizations on multi-frame segments. In particular, a set of segmental mixtures was used to model acoustic units via clustering. Then, the SGMM model was employed to produce unsupervised segment labels that formed the initial transcription for HMM training. This technique was applied on the Switchboard corpus, which consists of telephone conversations about pre-assigned topics.

In our work, we first present such an unsupervised approach for learning an audio *codebook*, by clustering low-level features extracted from a sequence of audio frames and then to the codebook, to get a *histogram-of-codewords* representation of a video's audio content. This representation can then be used for supervised training of classifiers using labels of high-level concepts, or unsupervised nearest neighbor retrieval.

One limitation of the unsupervised codebook approach is that it quantizes the audio stream to a set of discrete codewords and could potentially lose important information. Simply increasing the codebook size would cause overfitting and a drop in performance. However, directly matching low-level audio features is difficult due to different number of feature vectors from audio clips of varying length. We address this by using the *pyramid match kernel* (PMK) [12], which has been previously proposed for matching sets of image features. PMK in effect provides a measure of similarity between two *sets* of feature vectors. Our experiments show that PMK significantly outperforms codebook projection based distance measure for audio classification and retrieval.

In the rest of the paper, we present the unsupervised codebook learning approach in section 2, the pyramid match kernel (PMK) in section 3, experimental results in section 4 and conclude in section 5.

28 – 31 August 2011, Florence, Italy

2. Unsupervised codebook learning and representation

We learn codewords to represent the audio content by clustering a set of feature vectors extracted from different frames of an audio dataset, using k-means or similar unsupervised clustering algorithm. Given a new audio clip, we assign each feature vector x_i to the nearest codeword:

$$\boldsymbol{\alpha_i} \in \{0,1\}^K, \alpha_{i,j} = 1 \Leftrightarrow j = arg\min_{k \leq K} ||x_i - c_k||^2 \quad (1)$$

where c_k is the k^{th} codeword. Then, we take the average of the $\boldsymbol{\alpha_i}$ assigned to different codewords for different feature vectors, and represent each video by the vector **h**:

$$\mathbf{h} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\alpha_i} \quad (2)$$

where N is the total number of feature vectors. This approach is similar to the commonly used *vector quantization* [13]. Given a set of training videos and their class labels, we first obtain their codebook histogram representation as in equation (2), and then train suitable classifiers. In our experiments we train SVM with radial basis function (RBF) kernels. These classifiers were then used to classify test videos.

This representation is simple, but effective in automatically learning key patterns in the data and does not require extensive manual annotation of audio segments. However, such vector quantization compresses the features extracted from the video, and could potentially lose important discriminative information, while increasing the codebook size beyond a point will result in overfitting and drop in performance. We next present an approach to address this, by directly matching low-level features.

3. Low-level feature matching using Pyramid Match Kernel

A key challenge in directly matching sets of low-level features from two different audio clips is that they have different number of feature vectors, and hence cannot be directly input to a classifier. The simplest approach to address this is to classify each frame individually and then use voting to decide the clip's class. However, this approach is sub-optimal and it is also difficult to train the classifier on large training sets with hundreds of hours of data and millions of frames. We address this by directly matching the features extracted from the frames of audio clips using the *pyramid match kernel* (PMK) [12], that has been used in the computer vision community for matching unstructured image features.

The basic idea behind PMK is that, instead of matching each individual feature between two feature sets, we compute intersections over multi-resolution histograms of the features. The similarity between two sets is then measured based on the weighted sum of the histogram intersections at each level. The compute time of the multi-resolution pyramids as well as the weighted intersection are linear in the number of features. We will next describe the PMK approach following the notations in [12].

Let each input feature set **X** be such that:

$$\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\} \quad \mathbf{x}_i \in \Re^n \quad (3)$$

Also, each element $\mathbf{x}_i$ in **X** have a maximal range D, and the minimum inter-vector distance between unique points is 1. This can be enforced by scaling the data to a required precision and then truncating to integers. In our experiments, we normalized each dimension using mean and standard deviations estimated from a sample set of vectors and multiplied each dimension by 10.

We define a multi-resolution feature extraction function Ψ:

$$\Psi(\mathbf{X}) = [H_0(\mathbf{X}), \cdots, H_{L-1}(\mathbf{X})] \quad (4)$$

where, L=$\lceil \log_2 D \rceil + 1$. $H_i(\mathbf{X})$ is a histogram vector formed over points in **X** using d-dimensional bins with sides of length 2^i, i.e. $\Psi(\mathbf{X})$ is a histogram pyramid, with each subsequent component histogram having bins whose sides are double the previous one. The lowest level histogram H_0 has a bin size small enough to contain just 1 unique feature point per bin, which is then increased until all points in F are in a single bin at level $L-1$.

The pyramid match score P_Δ is computed based on the correspondences found in the multi-resolution histogram space Ψ. Given two input sets **Y** and **Z**, we define the similarity P_Δ as:

$$\hat{P}_\Delta(\mathbf{X}, \mathbf{Y}) = \sum_{l=0}^{L-1} w_l N_l, \quad w_l = \frac{1}{2^l} \quad (5)$$

where N_l is the number of newly matched pairs at level l, and w_l is the weight for matches formed at level l. Following the implementation in [12], we set w_l=1/2^l. Further, N_l can be efficiently computed using the difference between the intersections at successive histogram levels:

$$N_l = I(H_l(\mathbf{Y}), H_l(\mathbf{Z})) - I(H_{l-1}(\mathbf{Y}), H_{l-1}(\mathbf{Z})) \quad (6)$$

Note that, we define level -1 to be the level at which there are no intersections, i.e. $I(H_{-1}(\mathbf{Y}), H_{-1}(\mathbf{Z}))$=0. We normalize $\hat{P}_\Delta(\mathbf{X}, \mathbf{Y})$ using each input's self similarity:

$$P_\Delta(\mathbf{X}, \mathbf{Y}) = \frac{\hat{P}_\Delta(\mathbf{X}, \mathbf{Y})}{\sqrt{(\hat{P}_\Delta(\mathbf{X}, \mathbf{X})\hat{P}_\Delta(\mathbf{Y}, \mathbf{Y}))}} \quad (7)$$

This prevents favoring larger input sets, and higher values for $P_\Delta(\mathbf{X}, \mathbf{Y})$ correspond to a better match.

3.1. PMK-based empirical kernel map

The pyramid match kernel in equation (5) is a Mercer kernel, and hence can be used for kernel based classification algorithms, such as SVM. Following the approach in [12], we can compute the PMK distances of each feature set **X** to a set of training examples $\mathbf{X}_1 \ldots \mathbf{X}_m$:

$$\Phi_m(\mathbf{X}) = (P_\Delta(\mathbf{X}, \mathbf{X}_1), \ldots, P_\Delta(\mathbf{X}, \mathbf{X}_m))^T \quad (8)$$

Here, $\Phi_m(\mathbf{X})$ is referred to as the *empirical kernel map*. For the training data, this would give the Gram matrix of distance values between all pairs of training examples. It can be seen that the resulting matrix will have dominant diagonal terms. To address this problem we can apply a sub-polynomial kernel to the original Gram matrix values according to the transformation suggested in [14]. This transformed matrix is then used as the kernel to train SVMs. For the test data, we apply the same transformation to the distance vector in equation (8).

4. Experimental results

4.1. Experimental setup

For our experiments, we used 2794 videos of varying duration (ranging from 2 min to more than an hour) downloaded from YouTube using different queries. We aggregated these videos based on the queries into 9 classes: *Baking*, *Baseball*, *Military Parade*, *Protest*, *Robbery*, *Building Shelter*, *Sports*, *Traffic*, *War Footage*. The videos returned by YouTube for different queries had significant error in each of the classes including wrong retrievals, cartoon videos simulating robbery/fighting, news reports etc. We had human annotators manually analyze the videos we downloaded and remove the erroneous retrievals. After pruning by human annotators, we had 55, 53, 229, 289, 345, 48, 589, 246, 940 videos respectively, for the different classes. The total duration of the audio corpus was 200 hours.

We first learned our codebooks from a held out set of 61 videos, which was randomly selected with a uniform distribution across the different classes. The remaining video set (of 2733 videos) was used to train and test SVM classifiers based on the codeword histogram projections. We split the data into 10 different partitions at 8:2 train:test separation. We ensured that videos from each class were also split with the same ratio. Throughout all experiments we train SVM with RBF kernels and set the parameters γ to 0.5 and c to 10^4.

The low-level audio feature extraction was also common across all our experiments. The input raw audio was first transformed to a 45-dimensional feature stream. Features were extracted from overlapping frames of audio data, each 29 ms long, at a rate of 100 frames per second. Each frame was windowed with a Hamming window, and a 36-pole LPC smoothed power spectrum was computed for the frequency band 80-6000 Hz. From this, 14 Mel-warped cepstral coefficients were computed. The mean cepstrum and peak energy of each segment of speech was removed non-causally from the appropriate sub-vector, removing any long term bias due to the channel. In addition, the feature vectors were scaled and translated such that for each video the data has zero mean and unit variance. These base cepstral features with their first and second derivatives, together with the energy and its first and second derivatives, compose the 45-dimensional feature vector.

4.2. Audio classification experiments

In this section we evaluate the proposed unsupervised codebook learning and the PMK methods in terms of audio classification accuracy. We also compare their performance to phoneme recognition [1, 2, 3] and Segmental Gaussian mixture [11] approaches. Table 1 summarizes the classification accuracy of the various approaches.

Approach	Accuracy
Phoneme Recognition	40.5±1.1
SGMM	56.7±2.3
Unsupervised Codebook (k=64)	56.9±1.6
Unsupervised Codebook (k=1024)	66.1±2.6
PMK	**71.8±1.7**

Table 1: Comparison of the various approaches for audio label generation for classification with 8:2 train:test separation (2187 videos in training, 586 in test) and 10-fold validation.

4.2.1. Phoneme recognition approach

Initial experiments were performed to measure classification accuracy based on mid-level features extracted from a phoneme recognizer. The input speech was automatically segmented using a dual-gender phoneme-class recognizer that located pauses, non-speech and gender changes. The acoustic model used in decoding is part of BBN's audio segmentation system [15]. It is a gender-dependent, context-independent, phoneme-class model trained over 150 hours of broadcast news data. Phonemes are grouped into 8 broad classes: *voiced continuents, fricatives and sibilants, obstruents, laughter, breath, lipsmack, music* and *silence*. The first 5 classes are further tagged with male and female labels, resulting in a total of 13 distinct symbols. While the gender-specific labels are necessary for gender and speaker-change detection they add unwanted variability. Thus, we employed a mapping mechanism that reduced the label dimensionality and gives a compact but sufficient detail of audio events in a video. This map grouped the initial 13 labels into 4 distinct labels: speech, nonspeech (including noise), music and silence.

Recognition was performed over the 2733 audio set via a very fast one-pass Viterbi decoder that outputs a sequence of phoneme classes and their time offsets, given an audio waveform. Using the hypothesized output sequences of mid-level features we calculated for each video source the distribution of the mid-level duration (similar to equation 2). Then, as described in Section 4.1, we trained and tested SVM classifiers for different train:test partitions. The classification accuracy based on the phoneme recognition approach was 40.5% (also reported in Table 1). Not surprisingly, the performance of the phoneme recognizer approach is the worst due to the mismatch of the acoustic model to the recognition task. Also, the number of feature classes (4) is relatively low and therefore can only capture coarse variants of the numerous heterogeneous audio events present in our audio collection.

4.2.2. SGMM tokenizer approach

We investigated the use of an SGMM tokenizer following the approach described in [11]. The first component of the segmental tokenizer partitions the audio at boundaries defined by spectral discontinuities. The segmentation process is followed by modeling each segment with a polynomial (quadratic) trajectory model. The polynomial trajectory model parameters are used to compute the distance between segments for segment clustering. This distance is applied to the binary centroid splitting algorithm. The resulting clusters are used to train a SGMM via the Expectation-Maximization algorithm. During decoding, for each segment the index of the mixture term with the maximum likelihood is generated.

We trained the SGMM model on the held out set by using 64 clusters and used it to segment and tokenize the 2733 audio set. We then calculated, for each video source, the distribution of the duration of the tokens generated by the SGMM model. Finally, we trained and tested SVM classifiers on the 10, 8:2 train:test partitions. The classification accuracy based on the SGMM approach was 56.7% (also reported in Table 1).

4.2.3. Unsupervised codebook learning approach

We conducted a series of experiments to assess the effectiveness of unsupervised codebook learning described in Section 2. We trained codebooks of different sizes by unsupervised clustering of features from the frames of the 61-video held out set. We

varied codebook sizes from 8 to 2048 in factors of 2. Then, we projected the audio features from our video dataset to the codebooks, and obtained a codebook histogram representation as in equations (1) and (2). Next, we split the remaining video set (of 2733 videos) to training and test sets based on the 8:2 *train:test* ratio, trained and evaluated SVM-RBF classifiers following the procedure described in Section 4.1.

The unsupervised codebook learning approach improves classification performance as the codebook size increases with a codebook size of 1024 producing the best result, before saturating. The SGMM approach yields comparable performance to the codebook learning approach with a 64 codebook size. An analysis of the per-class classification errors shows that the error are complimentary.

4.2.4. Pyramid match kernel approach

In the final set of experiments, we tested our PMK based similarity measurement described in Section 3. For each train:test partition, we first computed the Gram matrix containing the PMK scores between all pairs of audio clips, and then trained our SVM for classification. The original approach in [12] was used to match sets of image features, with each set containing 200-300 feature points, which took ≈0.05s to compute each match. In our case, since each audio clip had 10k-50k frames, computing PMK on the entire set would be computationally expensive. To address this, we uniformly sampled 1 in 10 frames in each audio stream. Also, since we normalize the feature vectors based on the mean and standard deviation, we needed at most L=10 levels to match any given pair of clips. As we discussed in Section 3.1, to decrease the dynamic range of the Gram matrix, we used a sub-polynomial transformation and obtained the best performance 71.8% with p=0.2 (see [14] for details). This shows that using the entire low-level feature set for directly matching audio clips is most optimal.

4.3. Audio retrieval experiments

In this section, we compared the retrieval performance of vector quantized (VQ) representation of audio clips using unsupervised codebooks, with direct matching of low-level features using PMK. Here, for each video in the dataset we computed its similarity to every other video in the dataset, and retrieved the top-N nearest neighbors to the query clip.

For the VQ representation, we computed similarity based on the Euclidean distances between the codeword histograms and retrieved videos with the smallest distance. For PMK, we retrieved videos with the highest PMK score according to equation (7). If the retrieved video belonged to the same class as the query video, it was counted as correct. We repeated the experiment for top-N retrieval with N=1,2,5,10.

N	Codeword histogram (k=1024)	PMK
1	57.9	73.4
2	55.0	71.0
5	50.4	66.3
10	47.0	59.6

Table 2: Precision of unsupervised nearest neighbor retrieval(%).

Table 2 compares the precision of PMK based low-level feature matching with the codeword histogram representation. PMK significantly outperforms in nearest neighbor retrieval, demonstrating the efficacy of the approach for retrieving unstructured, noisy web audio.

5. Conclusion

We presented two techniques for audio classification and retrieval. The first method involved learning an unsupervised codebook by clustering audio features, and the second involved directly matching low-level features using the pyramid match kernel (PMK). Experimental results on a noisy and highly heterogeneous 200 hour audio corpus downloaded from YouTube show that both our approaches improve classification accuracy compared to traditional approaches. We also compared the two techniques in terms of nearest neighbor retrieval performance. The PMK approach significantly outperforms the unsupervised codebook learning approach.

6. References

[1] J. McDonough, K. Ng, P. Jeanrenaud, H. Gish, and J. Rohlicek, "Approaches to topic identification on the switchboard corpus," *ICASSP*, vol. 1, pp. 385–388, 1994.

[2] T. Hazen, F. Richardson, and A. Margolis, "Topic identification from audio recordings using word and phone recognition lattices," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2007, pp. 659 –664.

[3] R. Schwartz, T. Imai, L. Nguyen, and J. Makhoul, "A maximum likelihood model for topic classification of broadcast news," *Eurospeech*, pp. 1455–1458, 1997.

[4] Z. Liu, J. Huang, and Y. Wang, "Classification of TV programs based on audio information using hidden Markov model," in *IEEE Workshop on Multimedia Signal Processing*, Dec. 1998.

[5] P. J. Moreno and R. Rifkin, "Using the Fisher kernel method for web audio classification," in *ICASSP*, 2000, pp. 2417–2420.

[6] L. Lu, H.-J. Zhang, and S. Z. Li, "Content-based audio classification and segmentation by using support vector machines," *Multimedia Systems*, vol. 8, pp. 482–492, 2003.

[7] J. Portelo, M. Bugalho, I. Trancoso, J. Neto, A. Abad, and A. Serralheiro, "Non-speech audio event detection," *ICASSP*, 2009.

[8] M. Bugalho, J. Portelo, I. Trancoso, T. Pellegrini, and A. Abad, "Detecting audio events for semantic video search," in *Interspeech*, 2009.

[9] I. Trancoso *et al.*, "Audio contributions to semantic video search," in *ICME*, 2009, pp. 630–633.

[10] C. G. M. Snoek *et al.*, "The MediaMill TRECVID 2009 semantic video search engine," in *Proceedings of the 7th TRECVID Workshop*, 2009.

[11] M.-H. Siu, H. Gish, A. Chan, and W. Belfield, "Improved topic classification and keyword discovery using an HMM-based speech recognizer trained without supervision," in *Interspeech*, 2010.

[12] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *ICCV*, 2005, pp. 1458–1465.

[13] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proceedings of the IEEE*, vol. 73, no. 11, pp. 1551 – 1588, Nov. 1985.

[14] J. Weston, B. Scholkopf, E. Eskin, C. Leslie, and W. Noble, "Dealing with large diagonals in kernel matrices," *Principles of Data Mining and Knowledge Discovery*, vol. 243, 2002.

[15] L. Nguyen, S. Matsoukas, J. Davenport, F. Kubala, R. Schwartz, and J. Makhoul, "Progress in transcription of broadcast news using byblos," *Speech Communication*, vol. 38, no. 1-2, pp. 213 – 230, 2002.