Unsupervised Learning of Acoustic Unit Descriptors for Audio Content Representation and Classification

Sourish Chaudhuri, Mark Harvilla, Bhiksha Raj

Carnegie Mellon University, Pittsburgh, PA.
{sourishc, mharvill, bhiksha}@cs.cmu.edu

Abstract

In this paper, we attempt to represent audio as a sequence of acoustic units using unsupervised learning and use them for multi-class classification. We expect the acoustic units to represent sounds or sound sequences to automatically create a sound alphabet. We use audio from multi-class Youtube-quality multimedia data to converge on a set of sound units, such that each audio file is represented as a sequence of these units. We then try to learn category language models over sequences of the acoustic units, and use them to generate acoustic and language model scores for each category. Finally, we use a margin based classification algorithm to weight the category scores to predict the class that each test data point belongs to. We compare different settings and report encouraging results on this task.

Index Terms: audio representation, sound alphabet, unsupervised sound units

1. Introduction

With the rapid growth in the amount of multimedia data available on the web, the ability to efficiently, and effectively, retrieve information from audio or video files is crucial to the success of search engines. Current approaches for multimedia retrieval often rely heavily on the analysis of video, or on annotations or tags provided by users. We believe that the audio also contains significant information that can be leveraged to understand the content. In this paper, we work on analysing and characterizing *audio* in multimedia recordings. Specifically, we address the problem of automatically identifying which of a set of semantic categories a recording belongs to, based on audio.

Automatic content analysis for audio has been the subject of a significant amount of research in the past, motivated by the task of audio search and retrieval. The most common approach is to use a *vocabulary* of sounds, comprising clearly characterizable sounds such as gunshots, laughter, speech, animal sounds, music, crowd sounds etc. Audio is analyzed by detecting the presence of sounds from this vocabulary in it. Additional analysis builds on top of such detection. For instance, Chang *et. al.* [1] identify the presence of sounds from a vocabulary and combine this information with evidence from video. Slaney [2] describes a system that could be used to map between regular vocabulary and sounds of this kind by association. Friedland [4] navigates Seinfeld episodes taking advantage of traditional sitcom artifacts, such as music indicating scene changes and laughter following punchlines. Other analyses detect repeated sequences in a television broadcast stream [3], with the intent of identifying jingles, advertisements and so forth.

In the above, the basic mechanism involves *spotting* a set of known sound types in audio. Higher-level descriptions of audio must be obtained by further inference or by human supervision, after the sounds are detected. In this paper, we take a different approach. We model all audio as being composed of a sequence of a relatively small set of atomic sound units. The contents of an audio recording are represented by the specific sequence of units that compose it. All recordings of any *category* of sound are have similar compositions in terms of these units.

To characterize the audio, we must therefore automatically learn the sound units, which we christen *acoustic unit descriptors* or "*AUD*s". We must also learn the distribution of *AUD* sequences for different categories of recordings that we wish to identify. To do so, we model *AUD*s with Hidden Markov models. Learning the sound units is equivalent to learning the parameters of the HMMs for the *AUD*s. We model the distributions of *AUD*-sequences for different categories as category-specific n-gram language models over the vocabulary of *AUD*s.

We propose a maximum-likelihood solution that jointly estimates the parameters of the HMMs for the *AUD*s and the language models for the categories of audio from a training corpus. The solution is analogous to the unsupervised and semisupervised automatic learning of sub-word units in speech *e.g.* [9] [10]. The *AUD*-HMMs and category LMs can be employed to decide which category any new recording belongs to. We also propose a new classification algorithm for this purpose.

We evaluate the proposed analysis on the MED-10 data provided by NIST. These data comprise several Youtube quality recordings of several categories of events. The task we evaluate on is that of identifying the correct category for a given recording. Although, in principle, we could also employ evidence from the video in the recordings, we perform our classification based only on the audio, since our goal is analyze the amount of evidence derivable from the audio alone. Our experiments reveal that the proposed methodology is able to classify the recordings with remarkably high accuracy, given the complexity of the task.

The rest of the paper is arranged as follows. In Section 2 and 3 we describe the learning algorithm used to discover sound units from training data. In Section 4, we describe the algorithm to classify recordings into one of the categories. Section 5 describes the data we use for our experiments, and we describe our results in Section 6 and conclude with our findings.

Finally, we note that proposed approach has applications well beyond the application evaluated here. The ability to discover sound units in data is potentially useful in a variety of applications, such as segmentation, classification, indexing, and text- and example-based retrieval of audio data.

2. Modeling Sound Units

Consider sounds from a baseball video of a hitter batting in a run. Fig. 1 shows three video frames from an example of such a recording. The bat makes contact with the ball, producing a sharp metallic sound. This is followed by sounds of footsteps running, and muted exclamations from spectators. Finally, there is cheering and teammates congratulating the player once a run is scored. A listener familiar with baseball would be able to infer from the sequence of sounds that a baseball game is likely to be in progress, and that a hit or a run may have occurred. Although the precise sounds produced (bat hitting the ball, footsteps, cheering) may vary in nature, and the precise manner in which these sounds follow one another may vary, the overall pattern of sounds is still characteristic of the event.

This example illustrates the primary intuition behind our formulation. The acoustic events in the example – the sound of the ball being hit, footsteps, cheering, etc. are all *atomic* sound events that characterize the larger event of the run being batted in. Moreover, in addition to these key events there are other individual nondescript atomic events such as pauses, rustling, silence, etc. which occur in the recording. In fact, every instant of the audio may be considered to be a part of one such atomic event. The overall pattern of occurrence of these atomic events characterizes the larger event.

Following this intuition, we model all audio as a sequence of atomic sound units, or *AUD*s. Note that *AUD*s do not merely spot specific events in the audio stream – the entire audio stream can be *transcribed* in terms of these units, *i.e.* every segment of audio is part of some *AUD*. In principle, if every *AUD* were to have distinct semantic identity, the number of *AUD*s required to represent all audio would be very large. Instead, we hypothesize that if we represented the audio using just a small number of *AUD*s that represent *generalized* units, the patterns in the transcriptions of audio recordings in terms of these *AUD*s will still be characteristic of the larger events in the audio.

In the following discussion, we will not use the term *atomic event*, instead referring to them as *AUD*s. All reference to *events* refer only to larger-level events such as the above example of batting in a run. Patterns of *AUD* sequences characterize events.

The problems we must address now are twofold: A) **Learning:** i. Given a set of training audio recordings, we must learn the set of *AUD*s, and ii) In order to categorize or classify recordings into events, we must learn statistical characterizations of the patterns of *AUD* sequences for audio from different categories. B) **Classification:** Given the set of *AUD*s and statistical characterizations of categories, we need an algorithm to classify test recordings into categories. We address these issues below.

3. Learning Model Parameters

We model *AUD*s by hidden Markov models. Since we are primarily interested in characterizing the *AUD*s, rather than interpreting their semantics, learning the *AUD*s is equivalent to learning the parameters of the HMMs for the *AUD*s. We model the distribution of the *AUD* sequences for audio categories as N-gram language models over the vocabulary of *AUD*s.

We cast the learning problem as one of *maximum likelihood* estimation. We are given a collection of audio recordings $\mathcal{D}$. Assigned to each recording D_i in $\mathcal{D}$ is a *class label* $C_i \in \mathcal{C}$, where $\mathcal{C}$ represents the set of all classes. Although not necessary, we will assume that each recording is entirely assigned to only one class. Each audio recording D_i has an unknown *transcription* T_i as a sequence of *AUD*s. The *AUD*s are modelled by HMMs, whose parameters we collectively represent as Λ. The transcriptions of all recordings belonging to a class C are assumed drawn from an N-gram language model $H(C)$. The HMM parameters Λ and the set of language models for all classes $\mathcal{H} = \{H(C)\ \forall C\}$ are unknown and must be estimated

Figure 1: Example of a sequence from a baseball video.

from the data. We assume that the total number of *AUD*s K is known. In reality K is a hyperparameter that may be optimized.

We assume the dependencies shown by the graphical model in Fig 2: the acoustic realization of any recording depends on its transcription and not directly on the language model for the class. So also, the transcriptions only govern the acoustic realization and do not directly relate to HMM parameters.

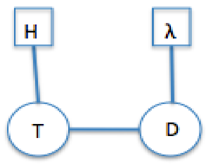

Figure 2: Graphical model for each data point D. Circles represent random variables and rectangles represent parameters.

The maximum likelihood estimate for Λ and $\mathcal{H}$ is given by:

$$\Lambda^*, \mathcal{H}^* = \mathrm{argmax}_{\Lambda,\mathcal{H}} P(\mathcal{D}|C(\mathcal{D}); \Lambda, \mathcal{H}) \quad (1)$$

Here $C(\mathcal{D})$ represents the classes assigned to each D in $\mathcal{D}$. In the notation above terms to the right of the semicolon are parameters, while remaining terms are random variables.

In principle, the above estimator must consider all possible transcriptions for any D. Instead we will approximate it by only considering the most likely transcription for any D. Also assuming that individual recordings D are independent, and that the class is represented primarily through the language model for the class, the estimator changes to:

$$\mathrm{argmax}_{\Lambda,\mathcal{H}} \prod_{C} \prod_{D_i : C_i = C} \max_{T} P(D_i, T; \Lambda, H(C)) \quad (2)$$

We obtain the above estimate using the iterative algorithm in Algorithm 1. In the figure superscripts appearing against the parameters indicate the iteration in which the estimate for the parameter was obtained.

It is simple to show that Algorithm 1 is a hill-climbing procedure that results in ever increasing likelihood for the data: Equation 3 ensures that

$$P(D_i, T^{r+1}; \Lambda^r, H(C)^r) \geq P(D_i, T^r; \Lambda^r, H(C)^r)$$

and Equations 4 and 5 ensure that

$$P(D_i, T^{r+1}; \Lambda^{r+1}, H(C)^{r+1}) \geq P(D_i, T^{r+1}; \Lambda^r, H(C)^r)$$

Equation 3 simply represents the automatic recognition of D_i using HMMs with parameters Λ^r and can be performed

Algorithm 1 Algorithm for learning *AUD*s and LMs

$$T_i^{r+1} = \text{argmax}_T P(T|D_i; H(C_i)^r; \Lambda^r) \quad (3)$$

$$\lambda^{r+1} = \text{argmax}_\lambda \prod_{D_i} P(D_i|T_i^{r+1}; \Lambda) \quad (4)$$

$$H(C)^{r+1} = \text{argmax}_H \prod_{D_i:C_i=C} P(T_i^{r+1}; H) \quad (5)$$

with the Viterbi decoder of a speech recognizer. Equation 4 is the learning procedure for HMM parameters Λ, given the recordings D_i and their transcriptions T_i^{r+1}, and can be performed using the Baum-Welch training module of any recognizer. Equation 5 represents the procedure for learning an N-gram language model $H(C)$ from the set of all transcriptions T_i^{r+1} of all recordings belonging to class C.

In order to model the data as proposed, we first represent all recordings as a sequence of mel-frequency cepstral vectors. Algorithm 1 requires an initial transcription for all recordings. We obtain this by segmenting all recordings by merging adjacent analysis frames, and finally clustering the obtained segments into K clusters. The sequence of cluster identities corresponding to the segments composing any recording form the initial transcription for that recording.

4. Classification

Given the HMM parameters Λ for all *AUD*s, and the set of language models $H(C)$ for all classes $C \in \mathcal{C}$, we can now classify a new audio recording D into one of the classes by Bayesian classification:

$$C^* = \text{argmax}_C P(D|C; H(C), \Lambda) P(C)$$

To compute $P(D|C)$ we must sum over all possible transcriptions of D, which is generally computationally intractable. Instead, we can employ the common approximation of only considering the most likely transcription:

$$\text{argmax}_C \log P(C) \max_T \log P(D, |T; \Lambda) + \log P(T; H(C)) \quad (6)$$

Here $\max_T \log P(D, |T; \Lambda) + \log P(T; H(C))$ is the log likelihood of the most likely transcription of D for class C for and can be computed by the decoder of a speech recognizer. $\log P(D, |T; \Lambda)$ is the *acoustic score* $A(D, C)$ for class C and $\log P(T; H(C))$ is the *language score* $L(D, C)$ for the class.

However, we do not use the above procedure directly for classifying the recordings. Instead we employ a second-level classifier that uses the acoustic and language scores for the class as features. Let $F(D, C) = [A(D, C)\ L(D, C)]^\top$ be a feature vector representing the acoustic and language scores for the data D computed for class C. For each class C we define a two-dimensional weights vector W_C. Classification is performed using the following classification rule:

$$C^* = C \ : \ W_C.F(D, C) > W_{C'}.F(D, C')\ \forall\ C \neq C' \quad (7)$$

To train the classifier we learn weights W_C for each class as follows: For each training instance D belonging to class C_D we decode D using $H(C)$ to obtain $F(D, C)$ for every class C. A training instance is correctly classified if:

$$W_{C_D}.F(D, C_D) > W_C.F(D, C)\ \forall\ C \neq C_D \quad (8)$$

The weights W_C can be learned to maximize classification accuracy on the training data.

We optimize an objective function with an iterative algorithm that follows the MIRA update rules, as described in [8]. Let us assume that for each training instance (D), the features scores (features) have been computed, and the label (y_j) is known, and there are τ such training instances.

Algorithm 2 Learning weights for each class

```
M = maxiter; i = 0; v = 0
$w_c = (0, 0), \forall c \in C$
$\mathbf{w}^{(0)} = \{w_1, w_2, ..., w_{|C|}\}$
for $m = 1$ to M do
    for $j = 1$ to $\tau$ do
        $\mathbf{w}^{(i+1)} = \min_w ||w||$
        s.t. $S(x_j, y_j) \geq S(x_j, y_c), \forall y_c$
        v = v + $\mathbf{w}^{(i+1)}$
        i = i + 1
    end for
end for
w = v$/(N \times \tau)$
```

In the algorithm, the score $S(x_j, y_c) = W_{y_c}.F(D, y_c)$ denotes the weighted score when using the weights for label y_c. Thus, the constraint requires that for all possible labels, the score should be less than with the true label y_j. It is possible to modify this formulation to include a margin by which the score of the true label should be greater than the score of other labels.

5. Data description

For our experiments in this paper, we use the TRECVid 2010 Multimedia Event Detection dataset (MED, henceforth) [5], from the NIST MED10 evaluation task. The MED data comprises 1746 total clips of training data, totaling about 56 hours in length, and the 1724 clips of test data about 59 hours long. The recordings are publicly available, user-generated multimedia content uploaded to internet hosts. Each video is annotated with one of 4 labels – *making a cake*, *batting in run*, *assembling shelter* and *other*, identifying the kind of activity being performed in it. The class *other* appears to be a catch-all class consisting of all videos that do not belong to the first 3 classes. Participants in the NIST MED evaluation were required to retrieve recordings from the testset that included a queried activity or event. Participants largely focused on the video features available. The use of the audio features was usually limited to speech transcriptions [6], and detection of pre-specified sound types in the audio [7].

In this paper we only use the audio in the recordings, since our objective is to evaluate our ability to characterize the audio. While these characterizations could potentially be combined with the video, we have not attempted that in this paper. We do not use any annotations besides the basic four activity labels provided, and use no other external data of any kind.

6. Experimental Results

For our experiments, we used the CMU Sphinx toolkit for HMM training and decoding, and the SRILM toolkit for estimating n-gram language models. We used 39 dimensional MFCC features to represent the audio, including the Δ and $\Delta\Delta$. *AUD*s were modeled with 5-state HMMs with a Bakis topology.

Table 1: *Classification based on Viterbi decoding scores*

System	3-class	4-class
64 symbols 2-gram	76.51%	64.79%
64 symbols 3-gram	75.00%	53.10%
200 symbols 2-gram	41.88%	67.42%
Class-specific HMM	36.88%	43.56%
Random	33.33%	25%

Table 2: *Classification using MIRA classifier*

System	3-class	4-class
64 symbols 2-gram	81.61%	73.61%
64 symbols 3-gram	80.30%	59.72%
200 symbols 2-gram	55.63%	77.08%

We evaluated n-gram language models for the classes with different n values.

We learned models for the classes from the MED10 training data and attempted to identify the class for each of the test recordings. The 4-class classification task contains an extremely skewed majority class: the *other* class has far more recordings than the other classes. In order to experiment with a more balanced dataset, we also experimented with 3 class classification, leaving out the data from the *other* class.

Table 1 reports accuracy (recall) for the 3 best settings obtained by classifying data directly using their Viterbi decode scores. We also compare with a simple baseline model, where we simply model each of the four classes with an ergodic HMM and perform Bayesian classification. The HMMs and the *a priori* class probabilities employed were tuned for best performance in the last case. Table 2 reports results using weighted MIRA classifier.

Overall, our experiments indicate that bigram language models outperform both unigram and trigram models on this task. Further, using 64 sound units appears to outperform systems that use more sound units on the 3 class classification task, but it doesn't do as well on the 4-class task. This supports the intuition that more units better capture a larger set of sounds. The MIRA classifier is generally significantly superior to classification based on Viterbi scores alone.

Employing our approach on the MED dataset involves significant challenges. For instance, the *other* class is not consistent in content, and contains a wide array of different audio and video. Besides the *other* class, the remaining 3 classes are not all well-structured. Events in the *batting_in_run* class have audio structure to them, as discussed earlier, but the audio in the *assembling_shelter* and *making_cake* classes are widely varied. Table 3 compares the accuracy for each class for the 200 symbol bigram models with the simple baseline class-specific HMM models on 4 class data.

It is not clear to us why the making cake class is better predicted with class-specific HMMs, but we believe it may have something to with the fact that audio corresponding to making a cake class appears to contain speech only in most cases and non-speech sounds do not have any class-specific interpretation, or do not occur sufficiently close for the sequence information to be used by our model. Our system does a very good job of identifying the other class, which is useful in reducing the number of false positives for the other classes, and makes this approach higher precision than the baseline.

Qualitatively, on analyzing the transcriptions generated on

Table 3: *Category specific accuracy for the various classes*

Class	Class-specific HMM	200 symbol 2-gram
assembling_shelter	31.11%	44.00%
batting_in_run	34.62%	59.62%
making_cake	43.86 %	24.14%
other	64.67%	94.70%

the training data by the iterative learning procedure, we find that it does a consistent job in identifying some sounds, such as the sound of a baseball bat hitting the ball or clapping, but runs into trouble when encountering other sounds, such as speech. Speech information appears to be distributed among various units. A couple of things could be done to improve this in future work- first, identifying speech segments as a step before sound unit learning, possibly in a supervised manner to help the system focus on non-speech acoustic events, or use a speech detector to ensure that speech events are constrained within a few sound units; second, we could start with a small amount of supervised data, that specify class-specific characteristic sounds to help the system converge to a better solution instead of building a sound dictionary from scratch.

In conclusion, we would like to note that the method to automatically learn sound units is a potent one. In the context of audio data, it is perhaps necessary to add in a layer of supervision in order to help add semantic information. Detection and recognition of speech and using the transcripts to help distinguish between categories should produce significant improvements in performance.

7. References

[1] S.-F. Chang, D. Ellis, W. Jiang, K. Lee, A. Yanagawa, A. Loui, and J. Luo, "Large-scale multimodal semantic concept detection for consumer video", MIRworkshop, ACM-Multimedia, 2007.

[2] M. Slaney, "Mixture of Probability experts for audio retrieval and indexing", In ICME, 2002.

[3] S. Berrani, G. Manson and P. Lechat, "A non-supervised approach for repeated sequence detection in TV broadcast streams", Signal Processing: Image Communication, vol 23: 525-537, 2008.

[4] G. Friedland, L. Gottlieb and A. Janin, "Using Artistic Markers and Speaker Identification for Narrative-Theme Navigation of Seinfeld Episodes", In Workshop on Content-Based Audio/Video Analysis for Novel TV Services, Proceedings of the 11th IEEE International Symposium on Multimedia, 2009.

[5] "TRECVid 2010 Multimedia Event Detection Evaluation", Online:http://www.nist.gov/itl/iad/mig/med10.cfm, 2010.

[6] H. Li, L. Bao, Z. Gao, A. Overwijk, W. Liu, L. Zhang, S. Yu, M. Chen, F. Metze, A. Hauptmann, "Informedia @ TRECVID2010", In TRECVID, 2010.

[7] M. Hill, G. Hua, A. Natsev, J. R. Smith, L. Xie, B. Huang, M. Merler, H. Ouyang and M. Zhou, "IBM Research TRECVID-2010 Video Copy Detection and Multimedia Event Detection System", In TRECVID, 2010.

[8] K. Crammer and Y. Singer, "Ultraconservative Online Algorithms for Multiclass Problems", In JMLR, 2003.

[9] M. Bacchiani and M. Ostendorf, "Joint lexicon, acoustic unit inventory and model design," Speech Communication, 1999.

[10] R. Singh, B. Raj and R. Stern, "Automatic Generation of subword units for Speech Recognition Systems", IEEE Transactions on Speech and Audio Processing, 2002.