Acoustic Features for Multimedia Event Classification

Stephanie Pancoast

CS 229 Project: Final Report

1. Introduction

Because of the popularity of online multimedia videos, there has been much interest in recent years in in multimedia event detection (MED) research. MED requires a system that can search user-submitted quality videos, like those found on YouTube, for specific events. Video features play a significant role is determining the content for MED tasks. However, the audio component for a given video can also be critical. Consider the case of detecting a home run in a baseball game video. From the still frames, video researchers may determine that the setting is a baseball game, but without the capability to detect cheering in the audio, it would be difficult to discriminate between an uneventful game and one wiht a home run.

There are various possbile approaches for using the audio component to better understand the video content. For my project, I explored the method of representing a video file's sound track as a *bag-of-audio-words*. Common to document classification (*bag-of-words*) and image classification (*bag-of-visual-words*) [2], the bag-of-audio-words method has also recently been used for both audio document retrieval [3, 4] and MED tasks [1]. In this paper I first discuss the acoustic features extracted from the audio component of videos as well as the bag-of-audio-words method in more detail. My experiments consisted of 15 binary classifications. This setup is presented in Section 3 and followed with the results and conclusion.

2. Methodology

A high-level illustration of the classification system is presented in Figure 1.

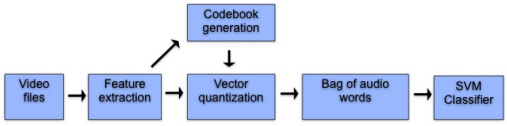

Figure 1: Block diagram of the classifications system

For each video file, the Mel-frequency cepstral coefficients (MFCCs) are extracted from the audio file. A codebook, also commonly referred to a dictionary, is generated using these MFCC vectors. The original features are then mapped to the nearest vector in the codebook. The file is then represented by what I refer to as a *word-vector*. The word-vectors for each file are then used as the feature vector for the classifier. All of these system components are discussed in more detail in this section.

2.1. Features

The mel frequency cepstrum represents the short-term power spectrum of a sound. It is based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency and can be calculated using Equation 1. The cepstrum can be thought of as a "spectrum of the spectrum" and provides information on how the spectrum energy changes over time. The MFCCs are the coefficients that make up the mel cepstrum [5]. The mel-scale on which the cesptrum is computed captures the non-linearity of human hearing. Equation 2 can be used to covert the typical linear frequency scale into the mel-scale.

$$Cepstrum = |\mathcal{F}\{log(|\mathcal{F}\{f(t)\}|^2)\}|^2 \quad (1)$$

$$freq_{mel} = 2595 log_{10}(1 + \frac{f}{700}) \quad (2)$$

MFCCs are standard features used in many speech recognition tasks such as automatic speech recognition and speaker identification [6]. The features used in my experiments are are computed for every 10ms audio segment and are extracted using a hamming window with 50% overlap. The features consist of 12 MFCCs as well as the log energy of the segment resulting in a 13-dimensional vector.

Often MFCCs (not including the log energy) are normalized within a given file so that the mean of each coefficient is 0 across all vectors. This is used to compensate for channel variability. For example, if one microphone tends to amplify a certain frequency range, MFCC mean normalization will account for this. In the remainder of this paper I use the term *zero-mean* to indicate normalized MFCCs and *original-mean* to indicate that non-normalized MFCC features were used.

2.2. Bag of Audio Words

As mentioned, written document retrieval commonly uses an approach referred to as "bag-of-words". Since documents can vary in length and classifiers often require fixed-length feature vectors, bag-of-words resolves this issue by representing a variable-length document with a fixed length word-vector. The dimension of this word-vector depends on the size of the codebook and each element i in the vector represents the number of times the word w_i appears in the given document. The word-vector can be thought of as a histogram or bag of words (as indicated by the method's name) as it does not capture the ordering of the words in the document. In image research word-vectors represent the distribution of visual words rather than words from a written language. Each visual word is the centroid of a cluster. The clusters are commonly created with the k-means clustering algorithm on the original visual feature space [2].

I chose to use a bag-of-audio-words approach for the project because my problem was similar to that of image and document retrieval. I had an initial set of acoustic features (MFCC vectors) and sought to classify a given document based on these features. My algorithm for generating the bag-of-audio-words involved the following steps:

1. Generate codebook
2. Quantize feature vectors

3. Replace features with codebook indices
4. Aggregate codeword distribution and use the resulting word-vector as the new feature for classifiers

The idea behind the codebook is to create a smaller set of features that are representative of the data. Here documents are represented by a bag of audio words, where each audio word (also referred to as a codeword) is the centroid of a cluster. This process is illustrated in Figure 2. I chose to use the k-means clustering algorithm based on the square euclidean distance measure to generate the codebook. I used MATLAB's implementation of k-means for my experiments. Across all videos, my dataset contains over 33 million MFCC feature vectors. Clustering all these vectors using the available computers was computationally demanding so I sampled at random 1% of the files with the idea that resulting clusters would still be representative of the sounds occurring in all the files. Performance of the overall system is dependent on the number of codewords and therefore the number of clusters, k, is one of the parameters I optimized for in my experiments.

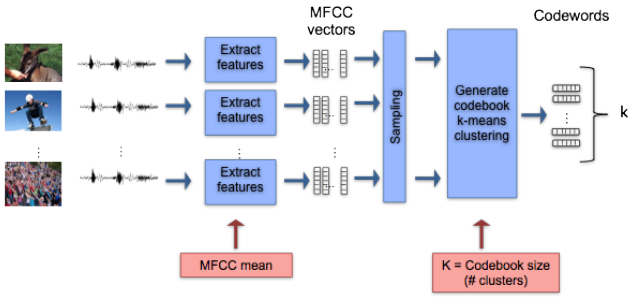

Figure 2: System diagram of how the codebook was generated from the audio.

The vector quantization (VQ) step consists of finding the codeword with the smallest distance to the original vector in Euclidean space and taking that codeword as the new feature vector. The bag-of-audio-words method uses the distribution of codewords in the file, not the actual values of the codewords. Therefore I instead kept track of the nearest codeword's index in the codebook and from there generated a word-vector to capture the distribution. Ideally, certain codewords would be more frequent in a given video event class and therefore a higher value in the word-vector for that codeword would be indicative of that class. This approach allows the video file to be represented by a single vector rather than a collection of MFCC vectors.

Since the videos vary in length they also vary in the number of MFCC vectors and in consequence the number of audio words. Without a normalization step, the magnitude of the word-vector would be proportional to the length of a file. Since the goal is to model the acoustic content, not the length of a video, this variation needs to be accounted for. I explored two different normalizing approaches which I refer to as *divide by max*, and *z-normalization* give by Equations 3 and 4 respectively.

$$c_{i,j} := \frac{c_{i,j}}{max_{k \epsilon |C|}(c_{k,j})} \tag{3}$$

$$c_{i,j} := \frac{(c_{i,j} - \mu_i)}{\sigma_i} \tag{4}$$

Here $c_{i,j}$ is number of codeword j occurences in document i, μ_i is the average value of word-vector elements for document i, and σ_i is the standard deviation of the codeword distribution for document i.

The *divide by max* divides all of the word counts in a given count representation vector by the maximum word count in that vector. This is equivalent to dividing the word-vector by the value of the largest element and ensures that all the elements in the vector are between 0 and 1. The *z-normalization* approach differs slightly by normalizing the word count distribution for a file to the standard normal Gaussian (mean 0, variance 1).

3. Experimental Setup

3.1. Dataset

I ran my experiments on a subset of user-submitted videos provided from NIST (National Institute of Standards and Technology) for the Text Retrieval Conferences Video Retrieval Evaluation [7]. More specifically, I used 2139 YouTube-quality videos ranging from approximately 5 seconds to 10 minutes in length. The audio component is sampled at 16KHz. Figure 3 shows the number of videos for each of the 15 event classes while Table 4.4 contains a mapping between the event ids and their description. Events 6 and 8 contain slightly more instances of videos and event 1 slightly less, but overall the the dataset is reasonably balanced.

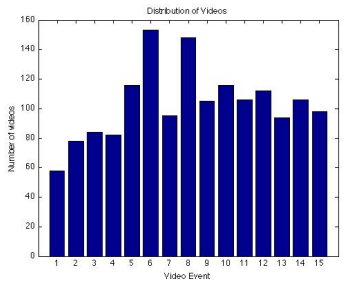

Figure 3: Data Distribution

Event Id	Brief Description
E001	Attempting a board trick
E002	Feeding an animal
E003	Landing a fish
E004	Wedding ceremony
E005	Working on a woodworking project
E006	Birthday party
E007	Changing a vehicle tire
E008	Flash mob gathering
E009	Getting a vehicle unstuck
E010	Grooming an animal
E011	Making a sandwich
E012	Parade
E013	Parkour
E014	Repairing an appliance
E015	Working on a sewing project

Table 1: Event ids with their corresponding descriptions

3.2. Classification

I chose to use a Support Vector Machine (SVM)-based classifier with a linear kernel using the LIBLINEAR SVM library in MATLAB [8]. SVMs have been found to perform well on audio and multimedia classification problems and are therefore commonly used in related research [1, 2, 3, 4].

My experiments consist of 15 binary classification experiments. For each video event class, such as E001, I generated binary labels as either "1" to indicate the video file was in the event class, or "0" to indicate it was not. This type of experiment is often referred to as *verification* or *one-against-all* [2] and is helpful for providing insight into the system performance for different event classes.

I chose to train on $\frac{4}{5}$ of the data and test on $\frac{1}{5}$, using the same training and test set for all 15 binary classification experiments.

3.3. Performance Metric

To measure the performance of the system, two metrics are commonly used in document classification tasks. Mean average precision (MAP), where precision is calculated as $\frac{tp}{tp+fp}$, provides insight into the reliability of a positive (1) label. I used tp to indicate the number of true-positives, tn true-negatives, fp false positives, and fn false negatives. F-measure, also referred to as F-score or F1, is another metric and can be computed using Equation 5. I used both of these metrics to measure the performance of my system.

$$Fscore = 2\frac{pr}{p+r} \tag{5}$$

Here p is the precision and r the recall ($\frac{tp}{tp+fn}$). F-measure balances between measuring the reliability of a positive label as well as how well the system is doing at capturing the positive samples.

4. Results and Discussion

There are many parameters in the bag-of-words classification system that can be adjusted to seek optimal performance. In this section I will discuss those that I explored in my project.

4.1. Front-end features

As mentioned previously, I tried two different types of MFCC vectors: one set of zero-mean and one of original-mean. When averaging across all 15 binary classification results, I found a MAP of 0.154 for MFCCs with no mean normalization and 0.142 for MFCCs normalized to mean zero. These were computed holding the codebook size fixed at 1000 and using *divide by max* normalization of the word-vectors. Although the two results are similar, this indicates the channel variability is somewhat correlated on the video event class. Table 2 shows the MAP for every experiment with a fixed codebook size.

Interestingly, some one-against-all experiments showed significant improvement with zero-mean MFCC vectors while for others the opposite was true.For example, getting a vehicle unstuck (E009) MAP value improved by almost 0.3 with zero-mean MFCCs while various classes such as feeding an animal (E002) had MAP values decrease by over 0.1 when using zero-mean MFCCs. This suggests that channel variability is informative for some video classes, while not for others. This is not so surprising. For example, E009 may of the sounds occurring are generated from wind, also common for many other event classes such as changing a vehicle tire . Therefore, normalizing the MFCC vector would help eliminate this common factor and place more emphasis on other more discriminative acoustics in the videos. E002 on the other hand generally has fewer sounds, and those that are present do not span the entire video. Therefore, discriminating between E002 and other classes would be easier with the original MFCC vectors, as all the other noisier ones could be eliminated.

4.2. Codebook size

In the bag-of-visual-words approach, the number of clusters to be generated in the k-means algorithm have varied from hundreds to ten-thousands [2]. However, the codebook size is a tradeoff. When generating a small set of clusters, dissimilar sounds (as modeled by the MFCC vector) can be grouped into the same audio-word and the codebook is therefore more general but not necessarily discriminative. On the other hand, larger codebook sizes result in similar sounds being assigned to different audio-words, resulting in a more discriminative but less general vocabulary. Further, the larger the codebook size, the longer both the codebook generation and vector quantization steps take, especially when clustering on over 33,000 vectors.

Previous research using bag-of-audio-words explored codebook sizes ranging from 500-2000 [4] so I also tried sizes in this range. Results are presented in Figure 4. The plot shows the F-measure, MAP, as well as average recall across all 15 binary classification experiments, using original-mean MFCC vectors and *divide by max* normalization of the word-vector. Generating as many of 3000 clusters from 33 thousand 13-dimensional vectors was computationally demanding and the performance for codebook sizes beyond 1000 did not suggest there would be improvement by trying larger values for k, so I did not explore larger sizes. Since 1000 codewords showed the highest average recall and second greatest MAP values, resulting in the highest f-measure, I used 1000 codewords for the remainder of the experiments.

4.3. Word-vector normalization

As described in Section 2.2, I tried two approaches for normalizing the word-vector. Using original-mean MFCC vectors and codebook size 1000, I found that the MAP calculated across all 15 binary classification experiments was 0.154 for the *divide by max* approach and 0.094 for the *z-normalization*.

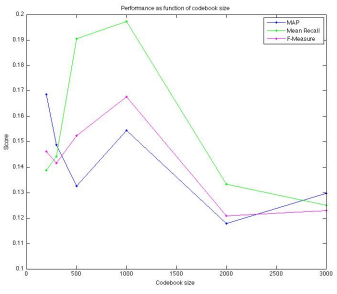

Figure 4: F-measure for various codebook sizes

4.4. Performance by event

In addition to optimizing performance across all experiments, it is also interesting to look how the classification system does for individual events. Figure 5 shows the F-measure for each of the binary classification experiments for codebook sizes of 500 and 1000. Original-mean MFCC vectors and *divide-by-max* word-vector normalization was held constant.

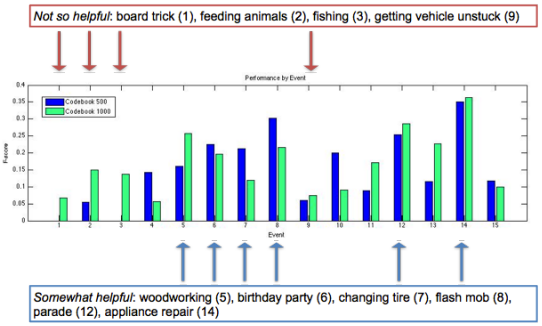

Figure 5: F-score (or F-measure) by event

Some of the event classes such as landing a fish performed poorly while others such as appliance repair best. Likely videos with little acoustic variety perform better with the bag-of-audio-words than others. For example, appliance repair videos are generally instructional and predominately speech with little background noise. Therefore, the codewords modeling speech would be very discriminative for this event class. Landing a fish videos, on the other hand, often contain sounds ranging from loud wind to motor boats to people screaming. Although these trends changed slightly when exploring other MFCC normalization and word-vector normalization techniques, the lower-performing and higher-performing events remained generally consistent. As mentioned above, the results for all the MFCC normalization and word-vector normalization experiments are presented in Table 2

	Orig. Mean		Zero-Mean		
Cbk Norm	DivM	Z	DivM	Z	Average
E001	0.071	0.058	0.000	**0.087**	0.054
E002	**0.188**	0.081	0.050	0.131	0.112
E003	**0.095**	0.044	0.067	0.043	0.062
E004	0.042	0.029	**0.118**	0.034	0.056
E005	**0.225**	0.141	0.120	0.114	0.150
E006	**0.163**	0.099	0.146	0.083	0.123
E007	0.097	0.078	**0.148**	0.091	0.103
E008	0.205	0.196	**0.229**	0.176	0.201
E009	0.095	0.070	**0.360**	0.155	0.170
E010	0.077	0.110	0.108	**0.159**	0.113
E011	**0.140**	0.103	0.111	0.066	0.105
E012	**0.292**	0.094	0.200	0.096	0.170
E013	**0.217**	0.100	0.103	0.105	0.132
E014	**0.286**	0.135	0.167	0.137	0.181
E015	0.125	0.069	**0.211**	0.113	0.129
Average	**0.154**	0.094	0.142	0.106	0.124

Table 2: MAP for each event with zero-mean or original-mean MFCC normalization, and word-vector divide by max (DivM) normalization or word-vector z-normalization (Z). These were computed for codebook size 1000. The highest MAP value for a given video class is shown in bold.

5. Conclusion and Future Directions

My experiments provide initial insight into the performance of using only the bag-of-audio-words approach for MED tasks. In this project I have explored normalization of both the MFCC vector as well the word-vector representation of the audio file. Results on an event-class basis indicate that certain classes perform better with the bag-of-audio-words classification system than others.

There are many possible variations that can be explored to seek performance improvement. On the acoustic feature level, MFCC's velocity and acceleration coefficients are often included in the initial feature vector so this would be one of the first things to try. Further, a different number of MFCCs could be extracted. Both of these variations would result in a larger initial feature vector and would likely have an impact on the results. After exploring possible improvements on the bag-of-audio-words approach, the word-vector representation will be able to be combined with visual features to further improve MED systems.

On the modeling side, the bag-of-words algorithm for document classification often incorporates a weighting scheme to account for the fact that some words (i.e. spam, httpaddr, unsubscrib for spam email detection) are more meaningful than others (i.e. the, a, and) [2]. Since I generated my words with a clustering algorithm, the audio-words are more likely to be evenly distributed, but this may still be an interesting approach to explore. In addition, feature selection algorithms are often used to eliminate codewords that are not discriminative. Finally, recent research has found improvement by modeling sequence of audio-words with N-gram models [4]. This approach exponentially increases the dimension vectors inputted to the classifier, thus requiring a smaller codebook size, but would better model the acoustics in the file and would therefore also be a possibly beneficial technique to incorporate into future experiments. After exploring possible improvements on the bag-of-audio-words approach, the word-vector representation will be able to be combined with visual features to further improve MED systems.

6. Acknowledgements

I thank Murat Akbacak and Eric Yeh from SRI International for their invaluable guidance in my project. Akbacak provided me with the idea to explore bag-of-audio-words as well as taught me the MFCC feature extraction process while Yeh was very helpful in suggesting classification and word-vector normalization techniques.

7. References

[1] Jiang, Y. Zang, X., Ye, G. , Bhattacharya, S., Ellis, D. Shah, M. Chang, S., "Columbia-UCF TRECVID2010 Multimedia Event Detection: Combining Multiple Modalities, Contextual Concepts, and Temporal Matching," 2001.

[2] J. Yang, Y. Jiang, A. Hauptmann and C. Ngo, "Evaluating bag-of-visual words representations in scene classification," in *Proceedings of the international workshop on multimedia information retrieval*, pp. 197-206, 2007.

[3] G. Checkik, E. Le, M. Rehn, S. Bengio, D. Lyon, *Large-scale Content-Based Audio Retrieval from Text Queries*, 2008.

[4] Kim, S., Sundaram, S. Panayiotis, G. and Narayana, S. "An N-gram model for unstructured audio signals toward information retrieval" Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on , vol., no., pp.477-480, 4-6, 2010

[7] "TRECVID multimedia event detection 2011 evaluation," *http://www.nist.gov/itl/iad/mig/med11.cfm*

[5] Rabiner, L. R. and Schafer R. W., *Introduction to digital speech processing*, 2007.

[6] Sanchez, M. H., *Nuisance Compensation and Prosodic Modeling on High-Level Speech Tasks*, Ph.D. Thesis, Stanford University, 2010.

[7] Gersho, A. and Gray, R. *Vector Quantization and Signal Compression*, Boston: Kluwer Academic Publishers, 1992.

[8] Fan, R., Chang, K. , Hsieh, C., Wang, X., and Lin, C., "LIBLINEAR: A library for Large Linear Classification," *Journal of Machine Learning Research*, pp. 1871-1874, 2008. Software available at http://www.csie.ntu.edu.tw/ cjlin/liblinear/