Homework 1: Audio-based Multimedia Event Detection

11-775 Large-Scale Multimedia Analysis (Spring 2020)

Due on: Wednesday February 5, 2020 11:59 PM

1 Task

The task of homework 1 is to perform multimedia event detection (MED) with audio features. You will learn to build the whole MED pipeline, which can be used for HW2 and HW3. Please **START EARLY**! It takes time to understand different tools in the pipeline. Also, the feature extraction process may be time-consuming.

2 MED Pipeline Overview

The overview of MED pipeline is depicted in Fig. 1. In the first step, we extract features from the raw training data. In this homework, the audio features we will use are MFCCs and ASR transcripts (Additional audio features like the SoundNet features can also be used). The parsing part further does some processing on the extracted raw features to pack them into the final representation, so that each video is represented by a single feature vector. Specifically, in this homework, we ask you to implement the bag-of-words representation with k-means clustering. With the representations and the labels, one of the typical approaches to train a classifier is to use Support Vector Machines (SVMs). The trained SVM models are then to be used in the testing phase. For testing, we extract and pack video representations on the test data with the same parameters/models we use in the training phase. With the pre-trained classifiers, we then score the videos accordingly and calculate the average precision to evaluate our MED system.

We provide a simple exemplary framework to cover part of the pipeline. Please refer to **`hw1_code/run.feature.sh`** and **`hw1_code/run.med.sh`** in the GitHub repository of this course [1]. Briefly -

1. **`run.features.sh`** generate representations of videos. It extracts features, train k-means (k-words), and represent videos with bag-of-words representation.

2. **`run.med.sh`** trains and tests the SVM models and demonstrate the results as APs (Average Precision).

Note that the provided code is just an example setup helping you to understand the basics of the MED pipeline. Also, the original parameters in the example framework may not give you good mAP (mean Average Precision). Therefore, you are NOT required to follow the provided example pipeline. We encourage you to create your own framework and try different tools and parameter configurations to get better results. The only requirement is writing a **`README.md`** in your GitHub repository, explaining how to run your pipeline. **Please share the repository with the TA (guiliangke on GitHub).**

[1] https://github.com/11775website/11775-hws.git

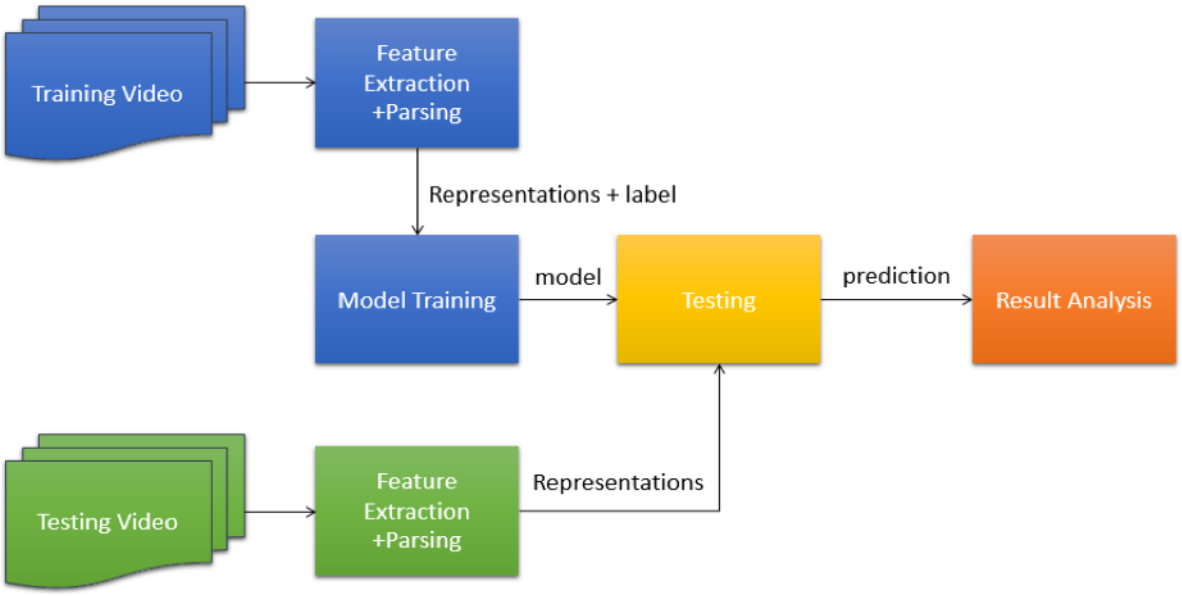

Figure 1: Overview of MED pipeline.

3 Dataset

The dataset will be the same for all three homeworks. It contains 2935 videos, with 3 positive events (P001: assembling_shelter; P002: batting_in_run; P003: making_cake) and 1 negative event class (NULL). The data is hosted at S3. Please read README.md for more details.

For training, the file **`all_trn.lst`** specifies 836 training videos and their labels. For validation, the file **`all_val.lst`** contains 400 videos and their ground-truth labels as well. You could use the validation set to tune hyper-parameters, conduct ablation studies and report your interesting findings in the report. For testing, there are additional 1699 videos specified in the **`all_test_fake.lst`**, in which their labels are all fake (deliberately set as NULL by us). You are expected to give classification scores on videos in this list using classifier of three event types with different features. For instance, if the first three lines in **`all_test_fake.lst`** are:

Example 1

```
HVC1012 NULL
HVC1060 NULL
NVC1062 NULL
```

Then, if you apply classifier P001 with MFCC feature on these videos, the first three lines of **`P001_mfcc.lst`** in your submission should look similar to this:

Example 2

```
0.2135  # confidence for video HVC1012
-0.1928 # confidence for video HVC1060
1.000   # confidence for video HVC1062
```

Each of the lines corresponds to the confidence with which each video is classified as P001 .

4 Features

4.1 MFCC Features

In class you will learn some popular audio features. Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel-scale of frequency. Mel-frequency cepstral coefficients (MFCCs)[2] are coefficients that collectively make up an MFC.

One of the well-known tools to extract MFCCs is openSMILE. In **`hw1_code/run.features.sh`**, the example script contains three steps:

1. ffmpeg : extract mono audio tracks (.wav) from the video files.
   `ffmpeg -y -i HVC5040.mp4 -ac 1 -f wav HVC5040.wav`
2. openSMILE : extract MFCC features
   `SMILExtract -C config/MFCC12_0_D_A.conf`
   `-I HVC1108.wav -O HVC1110.mfcc.csv`

We suggest you play with the openSMILE configuration file in **`hw1_code/config/MFCC12_0_D_A.conf`** after you finish the pipeline to get a better sense of MFCC features as well as better performance.

4.1.1 Bag-of-Words Representation

As stated in class, representing a video using bag-of-(audio-)words is one of the feasible approaches. In this part, we will train a k-means clustering model to cluster the MFCC vectors to represent a video using these cluster centers.

To speed up the clustering process, you can choose only a small portion of the MFCC vectors (f. ex., randomly select 20% MFCCs from each video). It will reduce the size of data for k-means clustering and thus speed up your pipeline development. It is up to you whether to use this speed up strategy.

Please implement functions to train a k-means model or using other clustering algorithms to represent videos (this part is not included in our exemplary pipeline code). There are many possible ways to do so. For example, you may set $k = 200$ to train 200 cluster centers using Python sklearn package for k-means. Once you have the clustering, you then represent each video in a k-dim vector according to the histogram where each dimension indicates the counts of MFCCs to the "closest" cluster center. You may also try different distance measurements to find out which approach is more suitable for audio features. **Please note that exception handling may be required since there are videos with no audio track**.

4.2 ASR Transcriptions

For your convenience, we provide you ASR transcriptions in **`hw1_code/asr/`**. It contains many different formats of the ASR output. The 2 most important ones are `CTM` and `TXT` files. If you use the bag-of-words representation, you may only need to care only about the `txt` files, that is, the words outputted by the ASR. If a file is missing assume that there is no output by the ASR.

However an ASR system contains more information than just the words. This information can be found in `CTM` file. Each line in the CTM files is composed of 6 fields. From left to right, these fields denote:

[2] More information can be found here.

1. Video ID
2. Channel ID. Always 1
3. Starting time of the word, in terms of seconds
4. Duration of the word, in terms of seconds
5. Word
6. Confidence score

For example, "`HVC3715 1 2.02 0.23 do 1`" means that the word "`do`" is spoken from $2.02s$ to $(2.02 + 0.23)s$ in the video `HVC3715.mp4`. These could potentially become really interesting features, for example, you can calculate how much part of the video contains speech that can explain which kind of video it is. The confidence score of the ASR can say how confident the ASR was for outputting the word.

4.3 SoundNet

SoundNet developed by the MIT CSAIL Lab learns rich natural sound representations by capitalizing on large amounts of unlabeled sound. It yields significant performance improvements over the state-of-the-art results on standard benchmarks for acoustic classification. You can extracted the raw features from different layers (conv3, conv4, conv6 and conv8) of SoundNet pre-trained model using an open-source Tensorflow implementation. You could load them in Python using numpy as follows (e.g. the conv3 feature of HVC1040):

Example 3

```
import numpy as np
raw_feat=np.load('HVC1040_conv3.npz')
raw_feat=raw_feat['arr_0']
```

The raw features are 4-dimensional arrays in (N, H, W, C) format, whereas N is the batch size of the feature, H and W is the size of feature maps and C is the number of filters. You can refer to the implementation in the original paper, employ various kinds of feature encoding strategies on these features and report your findings on the MED dataset. We also expect you to explore the performance of different SoundNet layer features in your report. As of the ASR and MFCC features, you can submit optional score files (**`P001_soundnet.lst`**, **`P002_soundnet.lst`**, **`P003_soundnet.lst`**) in your submission under the **`scores/`** folder. Your work on SoundNet will be graded as extra points (up to 20 pts based on the completeness) for your homework.

5 Model Training and Testing

As provided in **`run.med.sh`**, please implement the **`scripts/train_svm.py`** and **`scripts/test_svm.py`** functionality accordingly with your own video representations. Again, you don't have to re-invent the wheel. There are lots of tools available such as scikit-learn and libsvm. Your

goal is to integrate these tools into your pipeline. You may also try writing functions to tune SVM parameters such as cost and kernels to get better results.

It should be noted that for MED, we are NOT performing a pure classification task. Instead, we are ranking "scores" of videos and performing a retrieval task. Here are the steps you need to follow:

- Apply the trained P00X classifier (using **`all_trn.lst`**) to the whole validation/testing list (i.e., **`all_val.lst`** and **`all_test_fake.lst`**). For each video, you get a decision score indicating how likely this video belongs to the event P00X. (A score, not a label!)
- Output the scores of the testing lists to a prediction file where each line corresponds to a testing video. The order of the videos in the original testing list should be kept in the prediction file (see Example in 3). Please attach all your prediction score results for videos in the **`all_test_fake.lst`** using MFCC and ASR features for event P001-P003 ($3 * 2 = 6$ score files) in your submission. These would be names as **`P00X_{MFCC|ASR}.lst`**
- You can also add other submissions which can be mix of MFCC and ASR or any other new features. You are also encouraged to try different classification models as well. This submission file will be called **`P00X_best.lst`**.

6 Evaluation

A simple tool for calculating Average Precision (AP) is provided in **`~/tools/mAP/ap`**. You may also use other tools such as average precision package by sklearn. However, please make sure the format is consistent (as shown in Section 3).

However, the results on the test set won't be able to be evaluated, since you do not have the real ground-truths of them. We will assess them using your submitted score files and grade you based on the results of **`P00X_best.lst`**. Note that you are expected to output results on the test set (**`all_tst_fake.lst`**).

For your reference, we provide a baseline implementation. A successful submission should at least aim at getting comparable results on the validation data (so that your final results on the test set won't be too bad). Submissions which significantly exceed our baseline with fancy works will get extra points.

7 What to submit

In Canvas: It is required to train and test MED detectors with two types of features: MFCC and ASR transcriptions. You are expected to submit 1) a report; 2) your individual prediction scores on the test set with two kinds of features for event P001-P003; 3) your GitHub username: 4) your Kaggle competition name. Please compress your submission into **`ANDREWID_HW1.zip`** and submit it through the turn-in link on Canvas.

In the report, first you are required to describe the detailed steps and parameters in your MED pipeline to get the results. Secondly, you should summarize your BEST average precision (AP) and mean average precision (mAP) on the validation set using two features on three events (P001-P003) respectively. In addition, we expect you to report the time it took (CPU time) to extract the features you submitted and tell us the amount of credit left on your AWS account. Please also specify your GitHub repository in your report and add a readme file under **`~/11775_code/hw1`** (on your repository), explaining how to run your pipeline.

In order to evaluate your submission easier, the overall contents of your **ANDREWID_HW1.zip** should follow the below structure:

- **report.pdf** // your pdf report for homework 1 using the template that we provide on canvas
- **scores/** // the root directory of prediction score files on the test set
- **github.txt** // link to your GitHub directory

Example 4

```
VideoID, Label
File0, 0  # classified as NULL
File1, 1  # classified as P001
File2, 2  # classified as P002
File3, 3  # classified as P003
...
```

8 AWS

In this course you will learn how to use Amazon Web Services (AWS). You will develop your MED pipeline on a t2.large instance with a provided Amazon Machine Image (AMI) (**20sp-11775hw**, the user name is **ubuntu** and the storage size is 100 GB).

If you cannot see the AMI, please switch your region to us-east-1 and make sure to be searching in the public AMI setting. Cost for t2.large is around $0.0928/hr. We have provided you with a $50 coupon, so you do not need to spend any of your own money for the homework. However, don't forget to stop your instance (**don't terminate it!**) when you don't need it. AWS charges per whole hour, regardless when you terminate, so stopping allows you to reuse the instance with no charges while it is stopped. For this class, we do not expect you to re-invent the wheel. We provide some basic tools for your homework under **~/tools/**. Anaconda has been setup with some basic environments, **python3**, **TensorFlow**, **PyTorch**, **MXNet**. In order to list the different environments, you can use **conda env list** to list the different conda environments. They contain the basic Python toolkits needed in multimedia event detection. Please note that you will have complete root access on your own AWS instance. Therefore, you can install any tools/packages you like for your MED pipeline to improve the performance and earn extra credits.