# 11-775 Large-Scale Multimedia Analysis

Prof. Alexander Hauptmann and Prof. Florian Metze
Homework 1

January 31, 2017

## 1 Collaboration Policy

You are allowed to discuss the assignment with other students and collaborate on developing it at a high level. However, your report and all the code must be entirely your own.

## 2 Task

In this homework, you are required to train and test a Multimedia Event Detection (MED). More specifically, your MED will be able to detect three different events: assembling a shelter, batting in a run and making a cake. You will build two different MED. The first one will use Mel-Frequency Cepstrum Coefficients (MFCC) and the second one will use Automatic Speech Recognizer (ASR) transcriptions. **This homework is due on Feb 6 (Mon) 23:59:59, 2017.**

In homework 2 and 3, we will assume that you will reuse a big part of the code generated for this homework, so it is very important to program this assignment in a reusable way.

## 3 Setup

t2.micro to start with

In this course you are required to learn how to setup and use an Amazon Web Server (AWS). You will develop your MED pipelines on a t2.medium instance with a provided Amazon Machine Image (AMI). The AMI provided will have the data and all the basic tools that you need to build both MED pipelines. One of the requirements of this homework is to build both MEDs with less than \$25.00. Take into consideration that a t2.medium on demand Linux instance have a cost of \$0.047 per hour approximately. We highly recommend to stop (**do not terminate it**[1]) your instance while you are not using it. When the instance is stopped Amazon will only charge you for the space that your data is using and for the public IP that your instance is using[2].

AWS offers the possibility to share operative systems images, also known as AMI. An AMI is a template that contains specific software configurations (operating system, application server, and applications). We have created a public AMI that contains the basic tools and data to develop this homework. To start your instance you have to access your AWS account and select the AMI section from the left menu. There you will be able to search our AMI by the following name:

[1]You can not recover the data of a terminated instance.
[2]IP cost is only applied if your instance has assigned an Elastic IP.

**11775sp2017**. Remember to search over the public images in US West (Oregon).

The Elastic Compute Cloud instances, also known as EC2, does not have a persistent public IP. Amazon can change your public IP in several situations (*e.g.* when your instance is stopped). To make your IP persistent you should associate it with an Elastic IP. This will allow your instance to be identified from outside always with the same IP. You are required to set an Elastic IP in your instance and leave your instance running for 24 hours after the submission.

## 4 Description

### 4.1 Data

The data is divided in three classes.

- **P001:** assembling a shelter.
- **P002:** batting in a run.
- **P003:** making a cake.

You can find all the parsed labels in `/home/ubuntu/hw1/list`. `P001_train`, `P002_train`, `P003_train` are the training set for each class. The training sets of each event contains 10 positive and 200 negative examples. `test` contains the complete test set. `P001_test`, `P002_test`, `P003_test` contain the test set of each event where 1 indicates a positive sample and 0 indicates a negative sample.

You can find the original video files (`.mp4`) in `/home/ubuntu/video`, you may take a look at them to get an idea of these events with the provided labels.

### 4.2 Proposed Pipeline (70 pts)

The overview of MED pipeline is depicted in Fig. 1. In the first step, we will extract features from the raw training data. In this homework, as we discussed before, MFCC and ASR transcriptions will be used as feature representation for our MED pipeline. In Fig. 1, parsing is the process of representing each video with a single feature vector. In this homework, we suggest you to implement a bag-of-word (BoW) approach to perform the parsing step. In order to create the centroids that BoW will use, we suggest you to use k-means clustering. Once each video is represented by one feature vector we will be able to train a classifier. One simple simple and elegant solution for classification is the Support Vector Machine (SVM).

We provide simple schemes scripts to cover the whole pipeline in `/home/ubuntu/hw1`. `/home/ubuntu/hw1/run.feature.sh` provides a basic pipeline for extracting features, train k-means (k-words), and generate a BoW representation for each video. `/home/ubuntu/hw1/run.med.sh` trains and tests the SVM models and shows the results as Average Precision (AP). All those scripts are called from `/home/ubuntu/hw1/hw1.sh`. All this scripts provided are just a guide and are not fully implemented.

Please note this does not mean that you have to follow our setup. In order to enhance your MED the default pipeline should be changed.

For testing, we perform the same feature extraction process (BoW with k-means) over the test set. With the trained classifiers, we can then score the videos and calculate the Mean Average

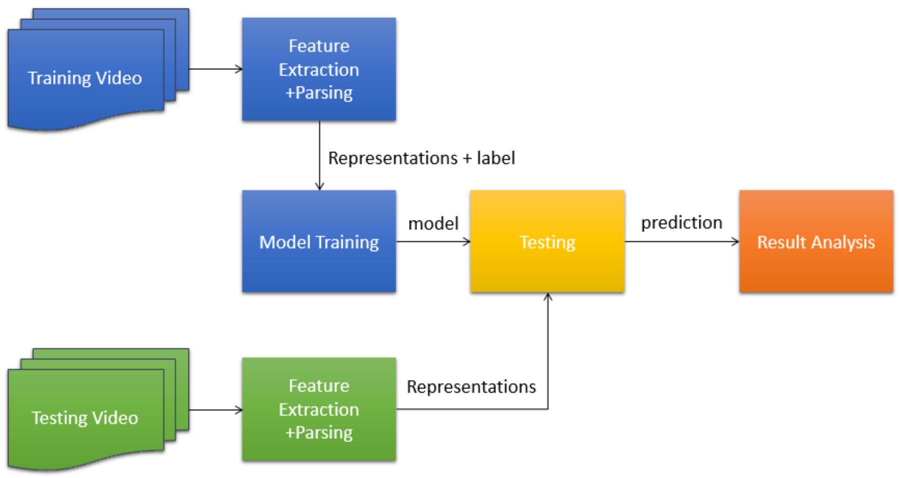

Figure 1: Overview of baseline MED.

Precision (MAP) to evaluate our MED system.

### 4.3 Tools Proposed

In this section we will introduce some of the tools that we suggest to use. Again, those are just suggestions, we encourage you to try other tools and methods. All tools presented have been compiled from the source code and are not installed (but added to `$PATH`) and python is located inside anaconda (`/home/ubuntu/tools/anaconda2`). Remember that you have `root` access and you can install or remove everything that you need.

#### 4.3.1 Feature Extraction

Ffmpeg is a popular audiovisual encoder/decoder. It will allow you to easily extract the audio track from a video file. Ffmpeg can be used in the following way:

```
ffmpeg -y -i video.mp4 -f wav audio.wav
```

In class, we learned some popular audio features such as MFCC. MFCC is a feature 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. One well-known tool for feature extraction from audio files is OpenSMILE. You can find an OpenSMILE configuration file in `/home/ubuntu/hw1/config/`. To perform the feature extraction of `sample.wav` file you can use the following command:

```
SMILExtract -C config/MFCC12_0_D_A.conf -I sample.wav -O mfcc.csv
```

We encourage you to use some of the large feature sets that OpenSMILE offers (*e.g.* PLP, ZCR, Prosodic). Here you can find some of them.

#### 4.3.2 Bag-of-Word Representation

As stated in class, a common approach to represent the audio modality of a video is to use a BoW approach. To do so, we suggest to train a k-means clustering model with the MFCC feature representation extracted from the audio track.

When performing k-means, it is very important to only select a small portion of the vectors that represents the audio track of a video, since we will be able to perform our training in a reasonable portion of time without affecting the final accuracy of our MED. We provide a simple python function (`scripts/select_frames.py`) to randomly select 20% of the feature representation of the audio track of each video.

One important parameter when performing a BoW is the number of clusters used. The number of clusters will define the number of features used to represent each video. One popular python package for machine learning that has k-means implemented is scikit. In `run.features.sh` you will find that `scripts/train_kmeans.py` and `scripts/create_kmeans.py` are called. However, they are just provided in terms of scheme, you should implement them yourself.

After the clusters are created you will be able to compute the histograms to represent each audio track from each video with just one feature vector.

#### 4.3.3 ASR Transcriptions

First, you will transcribe all the videos using the deep learning based ASR EESEN. You can find a compiled version in `/home/ubuntu/tools/eesen-offline-transcriber`. With the following command you can transcribe an audio track:

```
./speech2text.sh /home/ubuntu/hw1/audio/HVC1103.wav
```

Then, since the transcriptions provide a text representation of each video, we can use Natural Language Processing techniques to generate a unique vector representation based on text for each video. Here you can find different ways of creating such vectors.

#### 4.3.4 Evaluation

We provided you with a simple tool for calculating MAP. You can find it in `/home/ubuntu/hw1/mAP`. You may also use other tools or metrics to evaluate your implementations. scikit has some tools to evaluate classifiers.

## 5 Submission and Grading

### 5.1 Report (30 pts)

You should turn in a 2-3 page report[3]. You should clearly describe what implementation choices you made, especially what feature and techniques you used. Report your performance with the metric proposed (MAP) and other relevant metrics that you consider important, and provide an error analysis. The error analysis and comparison between methods/features will be highly valued.

### 5.2 Submission

You will send a compressed file with the following name: `11775_$andrewid_hw1.tar.gz` (*e.g.* `11775_ramons_hw1.tar.gz`) to ramons@andrew.cmu.edu that will contain your report, your AWS bill, and the ssh key of your AWS instance before Feb 6 (Mon) 23:59:59, 2017. They key should be

[3]We suggest to use the NIPS 2016 LaTeXtemplate that can be found here.

named as `$(whoami)_$elasticip_$andrewid.pem` (*e.g.* `ubuntu_35.164.167.95_ramons.pem`).

You will leave your AWS in "running" state 24 hours after the submission. And `/home/ubuntu/hw1/hw1.sh` will test (no feature extraction nor training) your models (audio and ASR transcriptions) with highest MAP.

Please, note that your assignment will be evaluated through a script. If the submission format is not correct we will not be able to evaluate your assignment.

## 5.3 Grading

For this assignment, the following points are required for successful completion of the homework:

- Your AP of all classes should be higher than 10% for the MFCC pipeline and 15% for the ASR transcription features pipeline.
- Your AWS bill should be less than $25.00 in the moment of submission.
- You are required to build this system with less than 100 GB of persistent storage.
- You instance should have a public IP assigned and your submission should have the proper format.

The highest-scoring submissions will be those that perform substantially better than the minimum requirements. Further investigation (*e.g.* error analysis) and the usage of different tools than the ones proposed on the basic pipeline will be highly valued.

## 5.4 Extra Credits (20 pts)

Trying some other tools (*e.g.* Kaldi for ASR), features (*e.g.* ZCR, PLP), different Kernel functions, etc. and it's further discussion and error analysis will result in a higher score. Finally, 3 best scores (MAP) among all the class will obtain full Extra Credit.