11775 Homework 1 Report

Wen He
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
wenhe@cs.cmu.edu

Abstract

This report describes the MED pipeline used to classify 2935 videos provided into three classes. Different approaches will be discussed and compared based on the average accuracy on the validation dataset and Kaggle submission result.

1 MED Pipeline Description

In this section, I will describe the detailed steps in the MED pipeline including feature extraction and model training. I will include parameters with the best performance in each step. Detailed comparison among different settings will be discussed in the next section.

1.1 Feature Extraction

The first step of the MED pipeline is to extract features from the videos. I have tried three different ways to extract features: using MFCC feature only, using ASR feature only, using the combined feature of MFCC and ASR. The following three sections will describe each approach in detail.

1.1.1 MFCC

I first used the shell script provided to do the MFCC feature extraction with the default MFCC configuration. The extracted MFCC features are then randomly selected with the selection ratio of 0.2. The selected features are concatenated and dumped into a single CSV file.

After the MFCC features are generated, they are then fed into the KMeans model. I used `MiniBatchKMeans` with 1000 clusters. The other parameters are of default settings. The KMeans model is then used to create histograms for each video. I used bag-of-words representation with normalization applied in creating the histogram. If there is no MFCC feature associated with certain file, I simply assigned the same normalized value to each entry.

1.1.2 ASR

Before extracting ASR feature, I first created a `vocab` file containing unique words from all transcript files with stop words and gibberish moved. The final `vocab` file contains 8546 unique words.

I then read in `TXT` files of ASR transcripts and created a histogram recording the term frequency of each word based on the `vocab` file generated previously. I applied normalization to each entry so that the term frequencies of all words for one transcript file add up to one.

1.1.3 Combined Feature

In order to make use of different features of videos to the greatest extent, I also tried to combine MFCC features and ASR features into one. For the combination, I simply concatenated the ASR features to the MFCC features for each video.

1.2 Model Training

The second step of the MED pipeline is to train a model to classify videos based on the features extracted in the first step. I used `sklearn.svm.SVC` class provided by scikit-learn to do the training. I trained three `SVM` classifier for each event (P001, P002, P003). The training data for each classifier is the features created in the first step and the labels are binary depending on whether the event matches with the input event or not.

I have tried many different kernels for `SVM`, and found that the chi-squared kernel (using the library `sklearn.metrics.pairwise.chi2_kernel` in scikit-learn) gives the best performance. The other parameters are of the default settings. Performance on other configurations will be discussed in the next section.

2 Result Evaluation

In this section, I will summarize the best average precision and mean average precision on the validation set with three different features and different configurations. I will also describe how I find out the best combination of parameters throughout the experiments.

2.1 MFCC Only

I first tuned the configuration of KMeans with different combinations of parameters. As can be observed from Table 1, the validation set has better mAP on all three classifier when `cluster_num=1000`, `batch_size=100` and `init_size=3000`. I also found that tuning KMeans parameters did not bring significant improvement on the validation set.

Table 1: Mean average precision on the validation set with different KMeans settings

Configuration	P001 mAP	P002 mAP	P003 mAP
Baseline	0.1992	0.2087	0.2724
`cluster=400,batch_size=100,init_size=1200`	0.1725	0.4486	0.2615
`cluster=1000,batch_size=2000,init_size=6000`	0.2727	0.1531	0.2747
`cluster=1000,batch_size=100,init_size=3000`	0.3383	0.3279	0.2189

I then shifted to the SVM parameter tuning. I first tried to change the `gamma` value to `scale` instead of `auto`. From the result shown in Table 2. It is clear that the performance is significantly better when `gamma=scale`.

Table 2: Mean average precision on the validation set with different gamma value in SVM

Configuration	P001 mAP	P002 mAP	P003 mAP
`gamma=auto`	0.3383	0.3279	0.2189
`gamma=scale`	0.3448	0.5495	0.2387

Based on the setting of `gamma=scale`, I tried different values of penalty parameter `C`. As can be observed from Table 3, the model gives best performance when the penalty parameter `C=1.0` which is the default setting.

Table 3: Mean average precision on the validation set with different penalty parameter values

Configuration	P001 mAP	P002 mAP	P003 mAP
`C=0.01`	0.3032	0.3491	0.2527
`C=0.1`	0.3313	0.4063	0.2325
`C=1.0`	0.3448	0.5495	0.2387
`C=10`	0.3084	0.5401	0.2342

In order to further improve the performance, I tried to change different kernels of the SVM. I first tried the built-in kernels provided by the scikit-learn including linear and sigmoid. All of them

did not seem to improve significantly. I then explored other customized kernels and found out that chi-squared kernel outperformed other kernels in the case of histogram feature vectors [1, 2]. The results are shown in Table 4.

Table 4: Mean average precision on the validation set with different kernels

Configuration	P001 mAP	P002 mAP	P003 mAP
`kernel=rbf`	0.3448	0.5495	0.2387
`kernel=linear`	0.3012	0.4547	0.2598
`kernel=sigmoid`	0.3267	0.2789	0.2133
`kernel=chi-squared`	**0.3576**	**0.7288**	0.2919

2.2 ASR Only

With the result obtained from experiments for MFCC feature, I jumped directly to comparing performance of different SVM kernels. Using chi-squared kernel significantly improved the mAP of P003 classifier while maintained same performance on P001 and P002.

Table 5: ASR mean average precision on the validation set with different kernels

Configuration	P001 mAP	P002 mAP	P003 mAP
`kernel=rbf`	0.2211	0.2373	0.3303
`kernel=chi-squared`	0.2273	0.2463	**0.6156**

2.3 MFCC + ASR

For the combined feature, I used the chi-squared kernel for SVM and the results are reported in Table 6.

Table 6: Combined feature mean average precision on the validation set

Configuration	P001 mAP	P002 mAP	P003 mAP
`kernel=chi-squared`	0.3014	0.6176	0.3726

From the above experiments, we can see that the best average precision on the validation set is `P001mAP=0.3576`, `P002mAP=0.7288`, `P003mAP=0.6156`, which are bolded in Table 4 and Table 5.

3 Deployment

The pipeline was developed on AWS EC2 instance of `t2.large` type with `2vCPUs` and `8GB RAM`. The total cost of this project is $9.06 and there are $140.94 credit remaining. The time taken for feature extraction in the pipeline is recorded in Table 7.

All the scripts are included in the GitHub repository `https://github.com/hw9603/11775-hws`. The README file in the `hw1_code` folder describes how to run the pipeline.

4 Conclusion

In this project, we developed a MED pipeline that explored the audio and text features of videos and did the classification on the videos. I learned the bag-of-words representation of the data. Different combinations of parameters for SVM will cause significant difference in the final performance of the model. The performance on the test set is 0.70552 based on the result on Kaggle.

In the future, there could be more to do with the sound feature. SoundNet could be one possible direction to work on. I could also utilize other information in the ASR transcript including the duration of words, confidence score, etc.

Table 7: CPU time taken for feature extraction in the MED pipeline

Module	CPU time
MFCC feature extraction	3 hrs
`select_frames`	30 min
`train_kmeans`	5 min
`create_kmeans`	34 min
`create_asr`	4 min

References

[1] `https://stackoverflow.com/questions/12779190/linear-svm-with-chi-squared-kernel-vs-rbf-kernel`

[2] `https://www.researchgate.net/post/Why_Chi-squared_kernel_SVM_outperform_other_kernels_in_image_recognition_based_on_histograms`