11-775 HW3 Pipelined MED Fusion

Xin Qian
xinq@cs.cmu.edu

Abstract

In this report, we conduct experiment and performance analysis on the performance of three Multimedia Event Detection (MED) tasks using video and audio features. Results show that feature learning with CNN are significantly stronger than other approaches, including ASR, MFCC, imtraj and SIFT features. We proposed three pipelines: early fusion with CNN Keras and MFCC features, late fusion with ensemble learning from three weak classifier from ASR and CNN features and late fusion with ensemble learning from three weak classifer from ASR, MFCC and CNN with an additional stacking layer. These three piplelines outperforms the best MAP reported in HW2 by at least 1 point. Differing in event difficulty, P001 has the most siginificant improment while P002 has the least, this is contrary to HW2 where we found that P002 was the easiest while P003 was the hardest.

For grading purpose, the running t2.medium instance ID i-0825fd2b3683ef880 has an associated Elastic IP of 52.24.150.222.

1 Implementation Choice

In this assignment, we implemented three improved fusion MED pipelines with different combination of video and audio frame feature setup. We begin with a review of available features for fusion. As the same setting in HW1 and HW2, we use ffmpeg to extract MFCC features and to cut the video to retain only the first 30 secs. This reduces the amount of data used to represent a video. The 30 secs is a suggested value without losing classification accuracy. We use tedlium configuration model to extract ASR text to train BOW features. In HW1, our feature extraction was conducted on 400 audios. This time, we expand our dataset to include 2935 videos. The average recoginition rate is 1 minues per audio. ASR features are selected as either tf-idf or tf from the training dataset vocabulary. Python NLTK library's default stopword lists, stemmer and lemmatizer are utilized to shrink vocabulary size and allow wording variations. The vocabulary was fixed with 2191 words.

Another available one is Imtraj pre-extracted feature. In the Imtraj/ directory, each video is represented as a single sparse vector. The sparse nature of the feature creates us some problem when we apply early fusion of them to other features. Compared to other feature that has no more than 5000 dimensions, the Imtraj feature occupies 32768 dimensions. During SVM training step, we convert the sparse vector into dense vector again (this is for code simplicity, although leaving it sparse for sklearn SVC is also supported.)

The forth feature is CNN. The suggested framework is Caffe with its wide selection of pre-trained model zoo. As in HW2, we pre-trained ImageNet-sized dataset models will greatly save time without losing effectiveness (since training from scratch is slow, expensive and tricky.) After preliminary exploration, we found that Keras is efficient in deploying with sufficient handy pre-trained model. My final choice is vgg16 model. The input layer for vgg16 require size to be 224*224*3.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Table 1: Baseline from HW2: 3 fold cross validation results using CNN Keras VGG16 pre-trained model features with AVG over vectors and rbf kernel

Event	MAP	Accuracy	TPR	TNR
P001	0.743173	0.9620	0.146822	1
P002	0.927116	0.9903	0.0455	1
P003	0.545770	0.9523	0	1
Average	**0.73869**	0.9682	0.1395	1

Each MFCC audio feature contributes a mixed bag of vocal words where we want to cluster k centroids to represent them well. To improve clustering efficiency while retaining clustering quality, I randomly sample 5% of vectors from all audio files. 5% is a heuristic number worth further examining, 5% is the comparative alternative choice from the 20% from HW1. Yet we argue this is a legitimate threshold as it yields around 100 MB aggregated vectors instead of GBs tiny vectors. These sampled vectors were fed into a skilearn KMeans++ function to learn cluster centroids. KMeans++ is a bit more robust than KMeans by overcoming the randomness in clustering seed initialization. Our default cluster number choice is 200, adopted from HW1.

We adopted the same 3-fold cross validation division to examine algorithm robustness as compared to HW2 best performance classifier (as baseline). For each event, there contains three rounds, each with 824 training videos and 412 validation videos. Our final test result are trained by the 1236 training videos. Sklearn's SVM function was invoked as the bottom level classifier in our pipeline, with the flexibility of altering series of training options, pipelining pre-processing such as standard normalization and training, etc.

One of the focus for HW3 is fusion technique. We attempted early fusion (which concatenates features from different modals into a unified feature vector), late fusion and SVM classifier ensembles with the help of brew toolkit in Python. Experiment results show that early fusion between MFCC and Keras features. SVM ensembles or ensemble together with other weak classifiers also improves the classification accuracy. However, late fusion with an AVG combination strategy is less promising.

In short, this section briefly introduces the pipeline implementation choice that I made during the pipeline designing process. Next section shows detailed control experiment result where we alter one variable in the pipeline to monitor metric value change.

2 Experiment Result

Below is a list of experiment that explores three pipeline variations. Table 1 is a reference implementation of Keras feature SVM classifier from HW2. Our best performing fusion classifier for P001 was shown in Table 2 with early fusion of MFCC and Keras feature. For P002, table 3 shows that late fusion by concatenating ASR and Keras followed by mean combinator yields the best(only marginal) improvement. For P003, late fusion of ASR, MFCC and CNN Keras followed by classifier ensemble and stacking yields best performance.

2.1 Pipeline 1: Early Fusion

Our first pipeline fuse modals before the learning process. It concatenates each selected unimodal features. Keras features have 4096 dimensions, imtraj has 32768 dimensions, MFCC has BOW size dimensions (default as 200) and ASR has 2191 dimensions as vocabulary size. This creates the sparsity unbalanced problem especially for imtraj features. Beyond simple concatenation, applying PCA to reduce dimension would be another way to learn fine-grained multimodal feature representation. One highlight of the early fusion pipeline is the high reusability of HW1 code – without modifying pre-existing learning phase code.

2.2 Pipeline 2: Early Fusion of ASR and Keras with ensemble

This pipeline has the same input features as pipeline 2. The difference is the learning phase, where we implement several weak classifier to learn from the same data using Python's brew

Table 2: Pipeline 1: Early fusion with MFCC and CNN Keras VGG16 pre-trained model feature

Event	MAP	Accuracy	TPR	TNR
P001	**0.818935**	0.9668	0.25396	1
P002	0.937117	0.9895	0.76767	0.99916
P003	0.573618	0.9531	0.0175	1
Average	**0.77609**	0.9698	0.34638	0.99972

Table 3: Pipeline 2: Early fusion by first concatenating ASR and CNN Keras VGG16 pre-trained model features as input, ensemble with mean combinator

Event	MAP	Accuracy	TPR	TNR
P001	0.754765	0.9749	0.4365	1
P002	**0.937274**	0.9895	0.7828	0.99833
P003	0.574956	0.9587	0.26183	0.9941
Average	**0.755277**	0.9744	0.49371	0.99747

toolkit. These weak classifiers are then ensembled together to give out predictions on the test set. From Table 2 and 3, we see that ensemble is good at predicting true-positive cases correct, where we see an increase of TPR for P003 (0.26183). P003 was originally one of the hardest class in HW2. Now we see a significant improvment on metrics, especially the anchor metric TPR (How many positive classess are classified correctly.) It is because the inclusion of ASR features. For P003, making a cake event, the image modal tends to be general indoor scene w/wo a cake object. Even with cake object being in place, a baby eating cake is not to our interest. This time, the tutorial stype dense colloquial records play an important roles. As long as the script contains words like "make" and "cake", our classifier could learn from these ASR features.

We chosed MEAN combinator to combine evidences from different weak classifier as one of the default setting. As we could see, the performance improves except for the slightly inferior performance on P001. Making a tent is less audio oriented and would have more image frame concentration. It will be possible that MFCC features in Pipeline 1 are completely ignored or obsecurely weighted.

Pipeline 3 will re-examine the setting with stacking.

2.3 Pipeline 3: Pipeline 2 plus Stacking

This pipeline adds to the foundation of Pipeline 2 another stacking layer. The essential idea of stacking allows us to properly weight classifiers trained from features of different space. As an example, Keras feature itself is outstanding, so it wins over larger weight. Stacking layer learns local misclassification mistakes made by individual classifier. We did not explicitly split our training set to perform stacking layer cross validation.

In most common settings, the three old folds in HW2 will be used to train Tier 1 classifers. We would need another forth fold to train the stacking layer.

Table 4: Pipeline 3: Late fusion by inputing concatenated ASR, MFCC and CNN Keras VGG16 pre-trained model features as input, ensemble with mean combinator and apply a stacking layer

Event	MAP	Accuracy	TPR	TNR
P001	0.77299	0.9741	0.4206	1
P002	0.93887	0.9895	0.7706	0.99916
P003	**0.58366**	0.9596	0.22143	0.9966
Average	**0.76517**	0.9744	0.4709	0.99859

Table 5: Failure case: Late fusion by AVG probability score from individual SVM trained from MFCC, CNN Keras, imtraj features

Event	MAP	Accuracy	TPR	TNR
P001	0.746119	0.96035	0.0912	1
P002	0.90577	0.96278	0.03737	1
P003	0.65116	0.9523	0	1
Average	**0.76768**	0.95848	0.0428	1

2.4 Failed Pipeline 4: Late Fusion

Late Fusion requires minimum tweaks on original feature extraction and training phase. They assume the existence of several indivial unimodal classifier, with differed strength. In our case, CNN with Keras features are the strongest. Although we combine classifiers in a uniform probability way (MEAN combination.) However, from Table 5, we see a decrease in accuracy, especially for P002. We would need extra meta-learning effort to improve upon this. We were also worring losing correlation between mixed feature space (e.g. correlation between MFCC and ASR features, or between Keras and imtraj features.) It would also worth looking of different combination strategy. Currently we examined MEAN, on the belief that it is most unbiased combination strategy.

3 Evaluation

3.1 Metrics

In this experiment, we used MAP, Accuracy, TPR and TNR to evaluate model effectiveness. Accuracy measures the percentage of decisions that are correct. MAP is more indicative than Accuracy in our dataset. This reveals a shortback of Accuracy. When the class distribution is highly skewed, Accuracy could be high even it cannot classify one single correct positive instance. This is because the majority of training instances are negative cases, which makes TP (True Positive) case extremely rare and TN case dominant in both training and test set. It makes no surprise that Accuracy stabilize at over 90% for all methods since all the methods have little confidence in classify the positive case right.

On the other hand, MAP treats MED as a search task, where a list of possible relevant document are returned with relevancy probability (in our case, it's the probability of classify the case as positive). In all above tables, MAP values are indicative in terms of classification/retrieval effectiveness.

The introduction of TPR and TNR in this assignment mitigates the problem. TPR and TNR serves as supplementary metrics for us to diagnose. Since our ultimate goal is to classify positve class correct, it would be better if one method has comparatively higher TPR with slightly lower MAP. However, in most senario as show in the tabels above, the correlations between TPR and MAP are high. Comparing Table 1 and Tabe 3 or Table 4, we see a significant improvement of TPR for P003. This is because the event, making a cake includes a lot of verbal features not captured by video. Including ASR features largely increases the decisiveness of Positive cases.

4 Future work

4.1 Handle missing modal

For some videos, they do not have audio in place, e.g. HVC1012. In my implementation, I set modal corresponding feature dimensions all as zero. There could be other solutions, for example, set the corresponding audio feature to be the average of all audio features. This is similar to the cold-start problem in recommender system, when ratings are missing for new incoming items or users, we assign a rating automatically, based on the ratings assigned by similar users within the community or from other similar items. Similarity would be determined according to user profile or item content.

4.2 Fusion of strong CNN features

Due to time limit, we were not able to examine more strong features, e.g. features extracted from different layers in a pre-trained CNN model. As different layes in CNN captures different granularity level of details, fusion on these features could give a more robust performance improvement.

References