EE599 Deep Learning – Homework 5

March 30, 2020

1 Objective

Spoken Language Identification (LID) is broadly defined as recognizing the language of a given speech utterance [1]. It has numerous applications in automated language and speech recognition, multilingual machine translations, speech-to-speech translations, and emergency call routing. In this homework, we will try to classify three languages (English, Hindi and Mandarin) from the spoken utterances that have been crowd-sourced from the class.

2 Goals

In this homework you will learn to:

1. Extract Mel-frequency cepstral coefficients (MFCCs) from audio, which will be employed as features.
2. Implement a GRU/LSTM model, and train it to classify the languages.

3 Steps

3.1 Dataset

The dataset has a bunch of wav files and a csv file containing labels. The wav file names are anonymized, and class labels are provided as integers. You are supposed to train with the provided integer class labels.

The following mapping is used to convert language IDs to integer labels:

```
mapping = dict{'english': 0, 'hindi': 1, 'mandarin': 2}
```

3.2 Audio format

The wav files have 16KHz sampling rate, single channel, and 16-bit Signed Integer PCM encoding.

3.3 Data split

We provide a training set after holding out a non-overlapping test set for evaluation. The test and training sets have no speakers in common, so that we can *truly* verify if the DNN can recognize language independent of speakers. You are free to create any validation split from the training set.

3.4 MFCC Features

MFCC features are widely employed in various speech processing applications including LID [2]. You will all use 64-dimensional MFCC features for this homework, so that the input features are same for everyone and the differences in performance do not depend on the choice of features. In addition, you will all use MFCC features generated using 25 msec frames with a 10 msec skip (or hop-length). This means that, for consecutive feature vectors, the start of the FFT frame differs by 10 msec and the FFT frame size is 25 msec – e.g., the first feature vector is based on audio in [0, 25] msec of audio and the second feature vector is based on audio from [10, 35] msec.

You can utilize **Librosa** (`https://librosa.github.io/librosa/index.html`) to extract 64 dimensional MFCC features for all utterances. A sample code snippet is provided below:

```
import librosa
y, sr = librosa.load('audio.wav',sr=16000)
#sr should return 16000, y returns the samples
mat = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=64, n_fft=int(sr*0.025), hop_length=
    int(sr*0.010))
print(y.shape, sr, mat.shape)
```

Note that this configured for 25 msec frames and 10 msec skip. A more detailed overview of MFCC features and librosa usage will be given in discussion.

3.5 Training Sequence Length

The full audio files are $\sim$ 10 minutes long. The training sequence length that you use should be long enough to capture the relevant time dependencies for the classification problem. Making teh training sequence length too long will make the training process take too long. Making the training sequence too short will cause relevant temporal information to be omitted in training the network and therefore worse inference performance. A reasonable training sequence length for this application is approximately 3-10 seconds. The choice of sequence length is up to you, and you are encouraged to experiment. You can refer to the the literature such as [1, 2]. The *held out* test set might have short and as well as long utterances.

Thus, to create your features you should do the following:

1. For each audio file of a given class (e.g., 'english'), you will compute a 2D array of features with shape `(M, 64)` where `M` is the number of 10 msec frames in the audio file. For a 10 second audio file, this would be 1000. Create a matching label array of dimension `(1, 64)`.

2. Concatenate all of these features for a given class into a single array of shape `(N, 64)` where `N` is the number of features for this language.

3. Once you decide on a training sequence length S, reshape this data into `(N_english, S, 64)` where `N_english` is the number of sequences that you have from your English data. Note that you should have an integer number of training sequences for each language.

4. Concatenate the feature and label data for all three languages to get a data set of shape `(N_english + N_hindi + N_mandarin, S, 64)` and similarly concatenate the labels.

5. Shuffle the data across the number-sequences axis. This will ensure that your val split contains data from all 3 classes.

Note that if you concatenate the data across languages before reshaping into sequences, you may get some sequences with languages mixed which is a form of contamination.

3.6 Handling Silence

Your audio data will have a label for every frame, but the recordings will also have short periods of silence in them as well. Since silence in English, Mandarin, and Hindi is indistinguishable, you may consider different ways of handling silence. These are reasonable options:

1. Just label the silence as the language of the file and don't worry about it. Your accuracy during silence will be low and will affect the overall accuracy you see in a file – e.g., the average accuracy over the entire file.

2. Pre-process your data to remove the silence. If there is useful information in the structure of the silence – e.g., different pauses between words in different languages – then you will lose this information.

3. Mask the silence. In this approach, you preprocess the audio to compute a mask when the audio is silent. You can use this mask in a custom loss function that you write. For example, you can use the mask to omit features from silent periods from the loss computation. This is essentially telling your netwrok that you "don't care" what it does during periods of silence.

Whichever of these methods you use, you should plot the 3 softmax outputs versus time for some sample files to see how the classification accuracy varies with the type of speech and also during periods of silence.

If you chose methods 2 or 3 above, you will need a method of marking audio frames as silence or "speech." This can be done by computing the energy in the 25 msec frame and comparing against a threshold. Some filtering on this silence indicator is also typically uses as the frames arrive every 10 msec and you probably only want to mark silent periods that last on the order of a second. Sox has simple energy based Voice Activity Detector (VAD) of this type that you can use. Care must be used in this process to handle different levels of background noise and also not to chop the start and end of utterances. In a more complete solution to the problem, another network could be trained to classify speech over other noises and used for this masking.

For the evaluation of your models after submission, we will use a masking script and compute the average accuracy over all non-silent periods.

3.7 Submission Material

You will submit a "streaming model" based on your training. The conversion for the sequence-trained model and the streaming model can be done as shown in lecture. Your streaming model should give three output scores (probabilities) for each 10 msec feature vector that it accepts – i.e., there are the probability of English, Hindi, and Mandarin. As mentioned above, In doing your own self-evaluation, you should plot these scores as a function of time for sample input files. You submission should also include a write-up that includes a description of how you arrived at your final model, the way you handled silence, the performance you observed, and a plot of your model.

4 Evaluation

Classification accuracy will be the primary metric for test set evaluation. The classification decision will be based on the average of the probabilities over periods of non-silence.

5 Submission

Please keep an eye on Piazza for submission related information. The above is a guideline and the specifics of the submission format and materials will be provided as the date approaches.

References

[1] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[2] A. Lozano-Diez, O. Plchot, P. Matejka, and J. Gonzalez-Rodriguez, "Dnn based embeddings for language recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5184–5188.