Assigned: 10 November 2020

Deep Learning – Homework #5

EE 599: Fall 2020

Due: Wednesday, 25 November 2020 at 20:00. Submission instructions will follow separately on Canvas.

1 Objective

Spoken Language Identification (LID) is defined broadly as recognizing 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 features in the form of Mel-frequency cepstral coefficients (MFCCs) from spoken audio.
2. Implement and train a GRU/LSTM model for language classification.

3 Steps

3.1 Dataset

The dataset consists of number of wav files with filenames indicating class. The wav file names are anonymized. **Train your model with the following** map to assign integer class labels to each language.

```
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. Each is approximately 10 minutes long. Note: your classmates self-labeled these samples. You should investigate any samples that repeatedly defy accurate classification. If you identify a mislabeled sample you may choose to either remove it from the dataset or assign it to the correct category.

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

Speech processing applications such as LID widely employ MFCC features in classification models [2]. Use **64-dimensional MFCC features for this assignment**. This ensures that the input features are same for everyone and performance differences are independent of feature selection. Use all MFCC features with 25 msec frames width and 10 msec skip (or hop-length). This means that consecutive feature vectors start with FFT frame difference of 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, *etc.*

Librosa (`https://librosa.github.io/librosa/index.html`) is a helpful tool to extract the 64-dimensional MFCC features from each training utterance. The following snippet provides a template.

```
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 sets 25 msec frames and 10 msec skips. The TAs will provide a more detailed description of MFCC features and librosa in discussion.

3.5 Training Sequence Length

Each audio file is ∼ 10 minutes long. Choose a training sequence length that you think will be be long enough to capture the relevant time dependencies for this classification problem. Choosing a too long training sequence will lead to long training times. A too short training sequence will miss relevant temporal information during training and lead to poor inference. You will probably settle on a reasonable training sequence length of approximately 3-10 seconds for this task. The choice of sequence length is up to you and we encourage you to experiment for better choices. You may also consult the literature such as [1, 2]. Note: the *held out* test set might have short and as well as long utterances.

To extract features you should do the following:

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

2. Concatenate all 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 validation splits contains data from all 3 classes.

Note: if you concatenate the data across languages before reshaping into sequences, you may introduce contamination by introducing sequences with mixed languages.

3.6 Handling Silence

Each frame of your audio data will have a label. But recordings are also littered with short periods of silence as well. You may consider different ways of handling silence since silence in English, Mandarin, and Hindi is indistinguishable. Some reasonable options:

1. 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.

Regardless of your choice: plot the 3 softmax outputs versus time for several sample files to see how the classification accuracy varies with the type of speech and also during periods of silence.

Methods 2 and 3 above require a method of marking audio frames as silence or "speech." You could do this by computing the energy in the 25 msec frame and comparing against a threshold. Some filtering on this silence indicator is also typically since frames arrive every 10 msec and you probably only want to mark silent periods that last on the order of a second. Sox includes a simple energy based Voice Activity Detector (VAD) that may be helpful. But use care in this process since different samples will have different background noise levels. Also be careful not to chop the start and end of utterances. Note: A more encompassing solution (not required here) may use a trained preprocessing network 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. Revisit lecture notes to remember how to convert a sequence-trained to a streaming model. Your streaming model must three output scores (probabilities) for each 10 msec input feature-vector – i.e., the probability of English, Hindi, and Mandarin. In doing your own self-evaluation you should plot these scores as a function of time for several sample input files. You submission should also include a write-up the summarizes your model and provides a description of how you arrived your answer. This should include the way you handle silence, your observed performance, and plots/summaries of your model similar to HW 4.

4 Evaluation

We will use classification accuracy as the primary metric for testing evaluation. We will hard-threshold the classification decision based on the average of the probabilities over periods of non-silence.

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.