Assignment-3

ELL-888

Submitted By-

Ramadev Sai Teja (2015EE10466)

Sant Prasad Mishra (2017EEZ8208)

Shantanu (2016EET2646)

1. Methodology:

Connectionist Temporal Classification (CTC) with LSTM has been used in this assignment to learn the word boundaries. The problem has been attempted as generation of character-level transcriptions from the spoken samples. Although world level and phoneme level transcriptions can also be done but character-level transcriptions have following advantages over other two methods- it is easier to detect the word boundaries, alleviates the problem of having different time lengths of same word spoken by different speakers and better generalization in terms of predicting the new words not seen during training. The inputs to the model are 1) audio sample file and 2) English transcription file. The audio is sampled at 22050Hz and then Mel-frequency Cepstral Coefficients (MFCCs) are extracted from it and used as features of speech data. We have used input sequence length of 336 with each sequence having 13 MFCC features. The reason of taking the MFCC sequence is that it closely approximates the human auditory system's response. From the transcription file the spoken sentence is read and all the punctuations are removed except space. Also, the uppercase letters are converted to lowercase. The initial configuration of labels is generated by randomly repeating the characters of input transcript with blank and space characters without changing the alignment of sentence (inbuilt function of tensorflow: `tf.nn.ctc_loss`) so that sequence lengths of input and output are same.

2. Implementation Details:

We used Tensorflow library to built LSTM and CTC loss (both are built-in functions in Tensorflow). Input sequence is extracted MFCC features as discussed above. The label for each instance of sequence is 28-dimensional one-hot vector (26 lower alphabets, 01 space and 01 blank character). Punctuations in the output have been removed. After testing we compute the Correct Detection Rate, Missed Detection Rate, False Detection Rate and Histogram of deviations of start of word boundaries. A brief description of enclosed source code files is given below:

utils.py - It contains helper functions for extracting MFCC features and generating one-hot vectors.

trainandtest.py- It contains code for training data and detecting boundaries on test data. The detected boundaries are saved to file.txt. By using online JSON editor, file.txt was converted to JSON format, which contains all sentences of test data and the starting word boundary for every word in each sentence. The final test result is stored in file.json.

accuracy.py- It reads file.json and computes Correct Detection Rate, Missed Detection Rate, False Detection Rate and plots Histogram of deviations of start of word boundaries.

3. Steps for Training and Testing:

Change TRAIN_DIR and TEST_DIR variables to train and test data directory locations in train.py and then run train.py. After running the model on test data, stored under test data directory, the detected word boundaries are stored in file.txt. The file.txt is converted to file.json for further computing test accuracies using accuracy.py

4. Result:

4.1. Training-

Single hidden layer LSTM with 50 nodes was trained using CTC loss for 1000 epochs with batch size of 256, which took approximately 34 hours on Laptop with i5 2.2 GHz CPU and 8GB RAM. Model is trained to recognizing text from audio. (Speech recognition)

4.2. Testing-

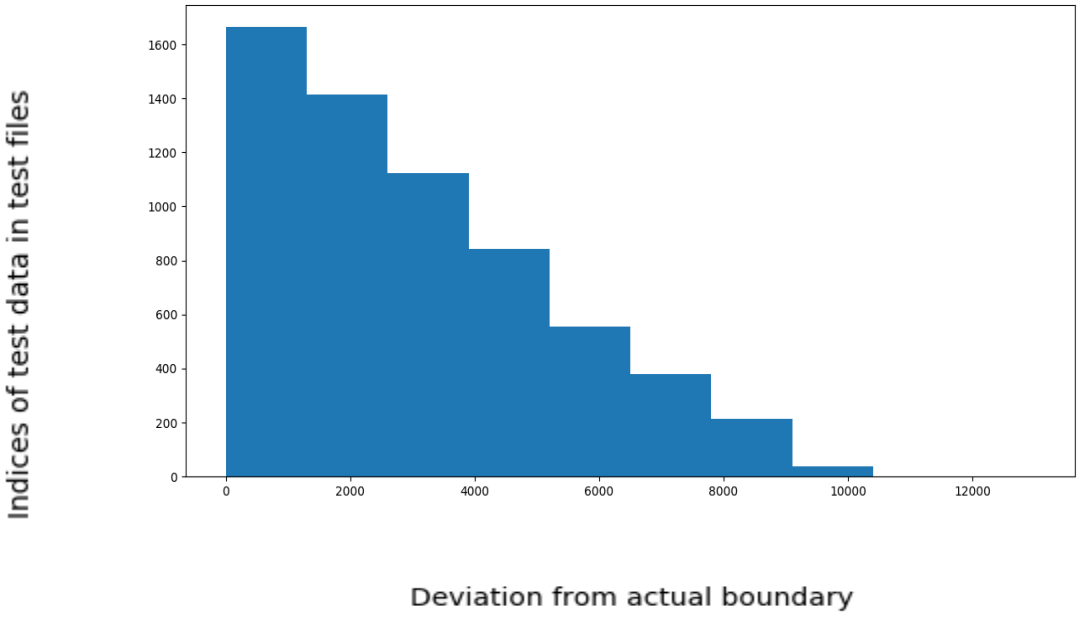
During testing we find out at which point in output sequence the space is recognized and converted it to 16000 Hz from 22050Hz scale. The accuracies on test data are computed as-

Correct detection rate: 50.23%

Missed detection rate: 45.28%

False detection rate:4.49%

Histogram:



5. Discussion:

With single layer LSTM and CTC loss the correct detection rate achieved on the given test data in this assignment is approximately 50%. This may be due to following reasons

1. Certain characters for example space, a, e, r, s and t are more common
2. Consonant-vowel-consonant patterns occur more frequently in English language

By incorporating following modifications, the accuracy might be improved further

1. Using Gram-CTC as a loss function.
2. Using deep LSTM
3. Using bi-directional LSTM with attention model
4. Using language models on top of character based RNN
5. Using CNN for feature extraction from audio data

6. References:

1. https://arxiv.org/pdf/1512.02595.pdf
2. https://www.svds.com/tensorflow-rnn-tutorial/
3. https://github.com/ugnelis/tensorflow-rnn-ctc
4. https://www.tensorflow.org/api_docs/python/tf/nn/ctc_loss