**Speech to Text for Indian Accent English Using Deepspeech model.**

**Abstract:**

To create an own model for automatic speech recognition using Baidu’s pre-trained deep speech model. For this, I used the deep speech pre-trained model version 0.5.1. The ASR model should understand the voice of speakers who speaks Indian English and convert that speech into text. And Compare the accuracy of speech to text between the pre-trained model and our own trained model. To calculate the accuracy I use WACC, WER and BLUE Score metrics.

**Context:**

* Getting a pre-trained model.
* Steps for loading the dataset of Indian Accent English Speech.
* Steps to transfer learning with the released model.
* Comparing results between my model and the Original DeepSpeech model.

**Getting a pre-trained model:**

For getting a pre-trained model I cloned a git repository from Mozilla DeepSpeech GitHub page. That contains the pre-trained model files like .pbmm file, lm\_binary file and trie model. And It contains some sample audio files used for training while developing the original model. And the repository contains the python code files that are used for training deep speech model. The deepspeech.py is the code file that is used for deep speech model compiling. The bin folder contains the python code files for downloading datasets from common voice, vanilla voice and set a frame rate of 16000 for each audio file and export that dataset into CSV file that contains the audio filename, audio filesize, and audio transcript and also split the dataset into train and test and validation. And we will load these three CSV files while training the model. Here is the sample code for that.

Example : ./DeepSpeech.py --train\_files ../data/CV/en/clips/train.csv --dev\_files ../data/CV/en/clips/dev.csv --test\_files ../data/CV/en/clips/test.csv

**Steps for loading the dataset of Indian Accent English Speech:**

I have collected an Indian English speech dataset for around 50000 audios and the audio description. The deep speech only takes the audio file that has the audio frame rate of 16000. I created a python file that automatically set the frame rate for all the audio files into 16000hz and calculated the file size for those audios. Then I created a data frame that stores the audio file path and the audio file path size and the audio file. Split that data frame file into three train.csv and test.csv and dev.csv. The training file contains around 35000 files and the test and dev file contains around 7500 files each. The train.csv file is used for training the deep speech model. And test.csv is used for evaluating the STT results of the deep speech model. And I used the KenLM toolkit for generating binary file and trie model for my dataset.

**Steps to Transfer learning with released model:**

To do transfer learning with deep speech model we can do by using --checkpoint\_dir in DeepSpeech.py. I need to specify the path where I downloaded the checkpoint from release and the training will resume from the pre-trained model. For example: If I want to fine-tune the entire graph using my data in train.csv, dev.csv, and test.csv, for 3 epochs we can tune hyperparameters as below,

python3 DeepSpeech.py --n\_hidden 2048 --checkpoint\_dir path/to/checkpoint/folder --epochs 3 --train\_files my-train.csv --dev\_files my-dev.csv --test\_files my\_dev.csv --learning\_rate 0.0001

Here are the hyper tuning parameters I have done for my model training.

python -u DeepSpeech.py \

--train\_files /home/prem/ds\_project/datavoice/data\_voice\_train.csv \

--test\_files /home/prem/ds\_project/datavoice/data\_voice\_test.csv \

--dev\_files /home/prem/ds\_project/datavoice/data\_voice\_dev.csv \

--n\_hidden 2048 \

--epoch 100 \

--use\_seq\_length False \

--checkpoint\_dir /home/prem/ds\_project/datavoice/checkpoints/ \

--learning\_rate 0.0001 \

--export\_dir /home/prem/ds\_project/datavoice/model\_export/ \

--train\_batch\_size 64 \

--test\_batch\_size 32 \

--dev\_batch\_size 32 \

The export directory gave me the output file output\_graph.pbmm. And I loaded the .pbmm file in deepspeeh.model() function. Then I used deepspeech decoder function to get STT for my testing audio files.

**Comparing results between my model and the Original DeepSpeech model:**

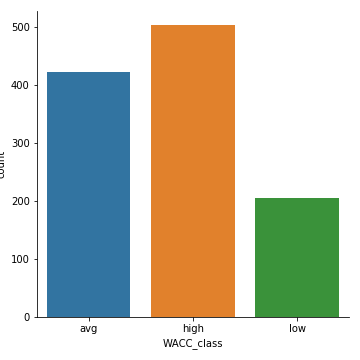
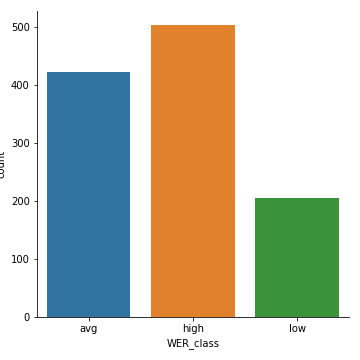
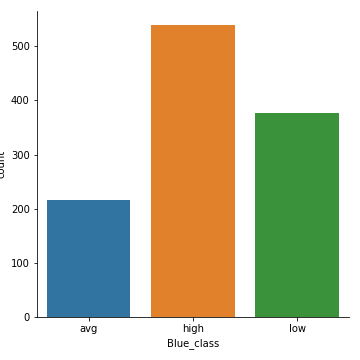
After training my model fro Indian Accent English I compared the results between my model and the original deep speech 0.5.1 model. For checking the accuracy of results I used metrics like **WER**, **WACC** and **BLUE SCORE**. For testing, I took around 1300 audio files of Indian Accent English and it’s voice transcript.

|  |  |  |
| --- | --- | --- |
| Metrics | Trained Model | Deepspeech Model |
| Word Error Rate | 0.157 | 0.44 |
| BLUE Score | 0.76 | 0.65 |
| Word Accuracy | 84.23 | 55.1 |

I have labelled values of Trained model and Deepspeech model.

1. **Trained Model**

I have labelled the values greater than 90 as high, greater than 70 as avg and less than 65 as low in **WACC**. In **WER, l**ess than 0.1 as high, less than 0.3 as avg and greater than 0.3 as low. In **BLUE Score,** greater than 0.85 as high, greater than 0.65 as avg and less than 0.65 as low.

**  **

1. **DeepSpeech model**

I have labelled the values greater than 90 as high, greater than 70 as avg and less than 65 as low in **WACC**. In **WER,** less than 0.1 as high, less than 0.3 as avg and greater than 0.3 as low. In **BLUE Score,** greater than 0.85 as high, greater than 0.65 as avg and less than 0.65 as low.

