EE627A Project Report

Utkarsh

Roll No.: 170767

15th May 2021

1 Digit recognition using Kaldi ASR toolkit

1.1 Problem Statement

We need to build an ASR (Automatic Speech Recognition) sytem using Kaldi software. We need to train the data and test the results using the metric **WER (Word Error Rate) and SER (Sentence Error Rate).** The training and testing set would consists of three digits spoken and stored in the form of **.wav** file.

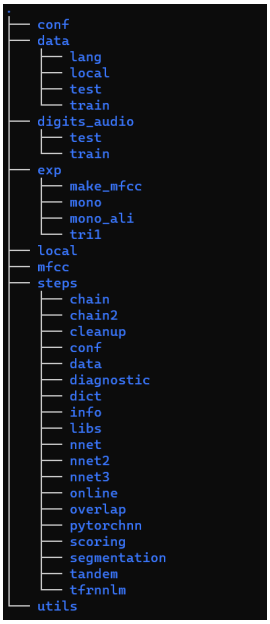
1.2 About Kaldi

Kaldi is a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. Kaldi is intended for use by speech recognition researchers. Kaldi enables to develop models using techniques like GMM (Gaussian Mixture Model), FST (Finite State Transducer) and DNN (Deep Neural Networks). The user does not have to code these models explicitly, but generate some essential codefiles which enables to run their data on these models.

1.3 Brief Explanation of code-files and working with Kaldi

I had mostly followed the tutorial: **Kaldi for dummies** to develop and run the relevant scripts.

1.3.1 Tree structure



1.4 Relevant Codefiles and Directories

1.4.1 digits_audio

It contains the data-set of the model. There are two folders **test** and **train**.
The test folder contains the testing audio files separated in the folder by their speakers.
The training folder contains the training audio files separated in the folder by their speakers.

1.4.2 data

The data folder contains relevant scripts and transcripts of the audio-files. It also has two folders, **test** and **train**.

Each of these folders are required to create these files:

wav.scp
This file contains all of the data-files name with the directory path.
text
This file contains the transcription of data-files alongwith the file name. Eg test-123 one two three.
spk2gender
This file contains all of the speakers along-with with their genders.
utt2spk
It maps all of the utterances, i.e. data-files to their speakers.

Miscellaneous files to be generated
corpus.txt
The corpus.txt should contain every single utterance transcription that can occur in your ASR system.

1.4.3 Language data

lexicon.txt
This file contains every word from your dictionary with its 'phone transcriptions'. The transcription is **phoneym** of the utterance.
nonsilence_phones.txt
This file lists nonsilence phones that are present in your project.
silence_phones.txt
This file lists silence phones.
optional_silence.txt
This file lists optional silence phones.

2 Experiments

We initially tested on smaller data-set around 50-70 files and the results were not good. We expected to increase the no. of training files.
Note: The left attached screenshot is for MONO testing and right attached screenshot is for TRIPHONE TESTING.

2.1 Experiment 1

num_iter = 40, max_iter_inc = 100, initial_beam = 6, Regular_beam = 10, Totgauss = 1250

```
%WER 10.49 [ 17 / 162, 4 ins, 0 del, 13 sub ]
%SER 24.07 [ 13 / 54 ]
exp/mono/decode/wer_11
%WER 10.49 [ 17 / 162, 4 ins, 0 del, 13 sub ]
%SER 24.07 [ 13 / 54 ]
exp/mono/decode/wer_12
%WER 8.64 [ 14 / 162, 2 ins, 0 del, 12 sub ]
%SER 20.37 [ 11 / 54 ]
exp/mono/decode/wer_13
%WER 8.64 [ 14 / 162, 2 ins, 0 del, 12 sub ]
%SER 20.37 [ 11 / 54 ]
exp/mono/decode/wer_14
%WER 8.64 [ 14 / 162, 2 ins, 0 del, 12 sub ]
%SER 20.37 [ 11 / 54 ]
exp/mono/decode/wer_15
%WER 8.64 [ 14 / 162, 2 ins, 0 del, 12 sub ]
%SER 20.37 [ 11 / 54 ]
exp/mono/decode/wer_16
%WER 8.64 [ 14 / 162, 2 ins, 0 del, 12 sub ]
%SER 20.37 [ 11 / 54 ]
exp/mono/decode/wer_17
%WER 8.64 [ 14 / 162, 2 ins, 0 del, 12 sub ]
%SER 20.37 [ 11 / 54 ]
exp/mono/decode/wer_7
%WER 14.20 [ 23 / 162, 11 ins, 0 del, 12 sub ]
%SER 29.63 [ 16 / 54 ]
exp/mono/decode/wer_8
%WER 13.58 [ 22 / 162, 10 ins, 0 del, 12 sub ]
%SER 27.78 [ 15 / 54 ]
```

```
%WER 26.54 [ 43 / 162, 16 ins, 1 del, 26 sub ]
%SER 61.11 [ 33 / 54 ]
exp/tri1/decode/wer_11
%WER 26.54 [ 43 / 162, 16 ins, 2 del, 25 sub ]
%SER 61.11 [ 33 / 54 ]
exp/tri1/decode/wer_12
%WER 24.69 [ 40 / 162, 14 ins, 2 del, 24 sub ]
%SER 57.41 [ 31 / 54 ]
exp/tri1/decode/wer_13
%WER 23.46 [ 38 / 162, 12 ins, 2 del, 24 sub ]
%SER 53.70 [ 29 / 54 ]
exp/tri1/decode/wer_14
%WER 23.46 [ 38 / 162, 12 ins, 2 del, 24 sub ]
%SER 53.70 [ 29 / 54 ]
exp/tri1/decode/wer_15
%WER 22.22 [ 36 / 162, 10 ins, 2 del, 24 sub ]
%SER 51.85 [ 28 / 54 ]
exp/tri1/decode/wer_16
%WER 21.60 [ 35 / 162, 6 ins, 2 del, 27 sub ]
%SER 50.00 [ 27 / 54 ]
exp/tri1/decode/wer_17
%WER 21.60 [ 35 / 162, 6 ins, 2 del, 27 sub ]
%SER 50.00 [ 27 / 54 ]
exp/tri1/decode/wer_7
%WER 32.72 [ 53 / 162, 27 ins, 1 del, 25 sub ]
%SER 66.67 [ 36 / 54 ]
exp/tri1/decode/wer_8
%WER 27.78 [ 45 / 162, 19 ins, 1 del, 25 sub ]
```

2.2 Experiment 2

num_iter = 40, max_iter_inc = 100, initial_beam = 6, Regular_beam = 10, Totgauss = 1050

```
%WER 5.23 [ 8 / 153, 2 ins, 0 del, 6 sub ]
%SER 13.73 [ 7 / 51 ]
exp/mono/decode/wer_11
%WER 5.23 [ 8 / 153, 2 ins, 0 del, 6 sub ]
%SER 13.73 [ 7 / 51 ]
exp/mono/decode/wer_12
%WER 3.92 [ 6 / 153, 1 ins, 0 del, 5 sub ]
%SER 11.76 [ 6 / 51 ]
exp/mono/decode/wer_13
%WER 3.92 [ 6 / 153, 1 ins, 0 del, 5 sub ]
%SER 11.76 [ 6 / 51 ]
exp/mono/decode/wer_14
%WER 3.92 [ 6 / 153, 1 ins, 0 del, 5 sub ]
%SER 11.76 [ 6 / 51 ]
exp/mono/decode/wer_15
%WER 4.58 [ 7 / 153, 1 ins, 1 del, 5 sub ]
%SER 13.73 [ 7 / 51 ]
exp/mono/decode/wer_16
%WER 5.23 [ 8 / 153, 1 ins, 1 del, 6 sub ]
%SER 15.69 [ 8 / 51 ]
exp/mono/decode/wer_17
%WER 5.23 [ 8 / 153, 1 ins, 1 del, 6 sub ]
%SER 15.69 [ 8 / 51 ]
exp/mono/decode/wer_7
%WER 7.84 [ 12 / 153, 4 ins, 0 del, 8 sub ]
%SER 15.69 [ 8 / 51 ]
exp/mono/decode/wer_8
%WER 6.54 [ 10 / 153, 3 ins, 0 del, 7 sub ]
```

```
%WER 15.03 [ 23 / 153, 2 ins, 0 del, 21 sub ]
%SER 35.29 [ 18 / 51 ]
exp/tri1/decode/wer_12
%WER 14.38 [ 22 / 153, 1 ins, 0 del, 21 sub ]
%SER 35.29 [ 18 / 51 ]
exp/tri1/decode/wer_13
%WER 14.38 [ 22 / 153, 1 ins, 0 del, 21 sub ]
%SER 35.29 [ 18 / 51 ]
exp/tri1/decode/wer_14
%WER 14.38 [ 22 / 153, 1 ins, 0 del, 21 sub ]
%SER 35.29 [ 18 / 51 ]
exp/tri1/decode/wer_15
%WER 14.38 [ 22 / 153, 1 ins, 0 del, 21 sub ]
%SER 35.29 [ 18 / 51 ]
exp/tri1/decode/wer_16
%WER 15.03 [ 23 / 153, 1 ins, 0 del, 22 sub ]
%SER 37.25 [ 19 / 51 ]
exp/tri1/decode/wer_17
%WER 15.03 [ 23 / 153, 1 ins, 0 del, 22 sub ]
%SER 37.25 [ 19 / 51 ]
exp/tri1/decode/wer_7
%WER 17.65 [ 27 / 153, 5 ins, 0 del, 22 sub ]
%SER 37.25 [ 19 / 51 ]
exp/tri1/decode/wer_8
%WER 17.65 [ 27 / 153, 5 ins, 0 del, 22 sub ]
%SER 37.25 [ 19 / 51 ]
exp/tri1/decode/wer_9
%WER 17.65 [ 27 / 153, 5 ins, 0 del, 22 sub ]
%SER 37.25 [ 19 / 51 ]
```

2.3 Further experiments:

1. I further tried with tri2 and FST(Finite State Tranducers) and MLLT testing. I was able to train the files not able to decode it.

2. I also tried to train it with the DNN Model but it required CUDA setup from which I was unfamiliar with.

3. Since the mono results were coming within the acceptable range i.e 10 %, I decided to go with it.

2.4 Final Result

I was able to get **WER of 3.92%** and **SER of 11.76%** on MONO testing.
I was able to get **WER of 14.38%** and **SER of 35.29%** on TRIPHONE testing.

2.5 Training the data

First we need to generate the training data-set files which can communicate with Kaldi. This can be done by using:

```
python3 gen_files.py train
```

Same would be done for testing data too:

```
python3 gen_files.py train
```

It will generate the required files for training. Train using bash script:

```
./run.sh
```

2.6 Running the code

The code can by run by using:

```
./transcribe.sh file.wav
```

The file should be in the format like: 1_2_3.wav

3 References

1. https://kaldi-asr.org/doc/kaldi for dummies.html
2. http://kaldi-asr.org/doc/data prep.html
3. http://jrmeyer.github.io/asr/2016/01/26/Installing-Kaldi.html