**AUDIO TEST SET’S WER CALCULATION AGAINST KALDI SWAHILI RECOGNIZER**

**Purpose**

We have used the recognizer created by Laurent Besacier & Company to test if this recognizer works with short audios. Our questions’ answers are very short and simple. We want to compare default WER values with our short audios WER results.

**Source**

Github version

<https://github.com/kaldi-asr/kaldi/tree/master/egs/swahili>

How to install in a Debian server

<https://drive.google.com/open?id=150kINwJUkchIrpssnKOjE6gdiPVRMcbxvwGI_csla6I>

**How to test with our audio test**

Prerequisite

-Set of testing audios recorded in a wav format

-Transcription per each audio

- (Optional) Speaker per each audio

**Steps:**

1. Kaldi needs to create some files: (see <http://kaldi-asr.org/doc/data_prep.html> “Files you need to create by yourself”)
   1. Text

‘Audio\_id transcription’

* 1. wav.scp

‘Audio\_id audio\_absolute\_pathname’ . Pathname like “/root/path\_to\_our\_audio\_folder”

* 1. Audio\_folder : folder which contains our audios
  2. Utt2spk

‘Audio\_id speaker\_id’ . If we don’t have speaker\_id write audio\_id again

* 1. spk2utt

Convert with the script:

utils/utt2spk\_to\_spk2utt.pl data/train/utt2spk > data/train/spk2utt

IMPORTANT: all files (text, wav.scp, utt2spk and spk2utt) must be sorted, sorting command:

s*ort text > text\_sorted*

*Mv text\_sorted > text*

1. Audio format

All the audios must be stored with some specific features: RAW 16 kHz, 16-bit, signed, little-endian (server native), linear PCM. See for the steps to be executed to convert them at: <https://docs.google.com/document/d/13IoOGc00dasvT4eb89VqJGWq3_IRaXxZ23zUpcqfA7w/edit>

1. Create our new folder at kaldi/egs/swahili/s5/data/ (f.e **new\_folder**)
2. Store all the files and folder defined at step 2. In that new folder
3. Create (after saving the original one) a new cmd.sh script located at kaldi/egs/swahili/s5 which includes those lines:

*# JHU cluster options*

*export train\_cmd="run.pl -l arch=\*64\*"*

*x=***new\_folder**

*steps/make\_mfcc.sh --nj 4 --cmd "$train\_cmd" data/$x exp/make\_mfcc/$x mfcc*

*steps/compute\_cmvn\_stats.sh data/$x exp/make\_mfcc/$x mfcc*

*steps/decode\_fmllr.sh --nj 4 --cmd "$train\_cmd" exp/system1/tri3b/graph data/$x exp/system1/tri3b/decode\_$x*

1. Execute cmd.sh script with:

Nohup ./cmd.sh > cmd.log.date

1. Analyze WER values. WER values are stored by the recognizer at (/exp/system1/tri3b/decode\_**newfolder). How to see WER values?** Go to that decode\_**newfolder** folder and execute:

Cat WER\* | grep “%WER”

Our test set’s WER values are:

%WER 114.29 [ 8 / 7, 3 ins, 0 del, 5 sub ]

%WER 114.29 [ 8 / 7, 3 ins, 0 del, 5 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 0 del, 5 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 0 del, 5 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 0 del, 5 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 1 del, 4 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 1 del, 4 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 2 del, 3 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 3 del, 2 sub ]

%WER 100.00 [ 7 / 7, 2 ins, 3 del, 2 sub ]

Default swahili recognizer’s WER values are: (they are stored at decode\_test)

%WER 13.10 [ 11 / 84, 1 ins, 1 del, 9 sub ]

%WER 14.29 [ 12 / 84, 1 ins, 1 del, 10 sub ]

%WER 14.29 [ 12 / 84, 1 ins, 1 del, 10 sub ]

%WER 15.48 [ 13 / 84, 1 ins, 1 del, 11 sub ]

%WER 15.48 [ 13 / 84, 1 ins, 1 del, 11 sub ]

%WER 15.48 [ 13 / 84, 1 ins, 1 del, 11 sub ]

%WER 15.48 [ 13 / 84, 1 ins, 1 del, 11 sub ]

%WER 15.48 [ 13 / 84, 1 ins, 1 del, 11 sub ]

%WER 15.48 [ 13 / 84, 1 ins, 1 del, 11 sub ]

%WER 16.67 [ 14 / 84, 1 ins, 1 del, 12 sub ]