Recognizing Slips of the Tongue With Kaldi ASR

Author:
Gaynora van Dommelen

Supervisor:
R. Ossewaarde

RESEARCH SEMESTER
TICT-OVRS-22

Content

IMPORTING PACKAGES

In [ ]:
```
import os
from os.path import isdir, join, dirname
import pandas as pd
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
import librosa
import librosa.display
import ipywidgets as wg
from IPython.display import display, Audio
import os
from typing import List, Callable, Tuple
from scipy.fftpack import fft, ifft
from spectrum import lpc
import matplotlib.transforms as mtransforms
from sklearn import metrics
import warnings
from src.file_frame import update_file_frame, get_samples
from src.sample import get_time_samples
from src.plot import plot_raw_wave, plot_fig, plot_spectrogram, plot_wind
from dotenv import load_dotenv
load_dotenv()
main_dir = os.getenv("DIGITSPATH")
warnings.filterwarnings('ignore')
```

Speech Errors

1. $s_1 : v \rightarrow p$ This person uses a *p* instead of a *v* consonant. So that the words: 'seven [s eh v ah n]' and 'five [f ay v]' now become 'sepen [s eh p ah n] ' and 'fipe [f ay p]'.

   > Do note that in this dataset the word 'fipe' does not represent anything, but closely resembles the word 'vipe' which in English is a word. So while this could give a low confidence score for this dataset, in a model trained on data where similar sounding words to 'fipe' exist, this doesn't necessarily happen.

2. $s_2 : w_1 \rightarrow w_2$ This person represents doubt. It is a fabrication of the situation where a sound is made belonging to another number while the intention and the end result is another digit, resulting in a combination of two digits. So, the words: 'one [w ah n]' $\rightarrow$ 'tw'one [t uw w ah n]', 'five [f ay v]' $\rightarrow$ 'fou'five [f ao f ay v]' and 'nine [n ay n]' $\rightarrow$ 'eigh'nine [ey n ay n]'.

The transcription for each of these words is the intended word. By using the intended word, the resulting transcription may show low confidence scores because of the uncertainty of combination of sounds.

To detect these 'anamolies' two approaches will be researched. The first approach involves determining the confidence scores in Kaldi, while the other approach involves using the time bound transcriptions to analyse the differences between the training and test group.

Step 1. Create project folder

In [ ]:
```
%%bash

cd ../git_workspace/kaldi/egs
[ ! -d digits ] && mkdir digits || echo "folder already exists"
```

Step 2. Gather data

Start by creating the folders that will contain the data.

In the `digits` folder, create the folder: `digits_audio` . Then, in the `digits_audio` folder, create the folders: `train` and `test` .

In [ ]:
```
%%bash -s "$main_dir"

cd $1;
[ ! -d digits_audio ] && mkdir digits_audio
cd digits_audio
[ ! -d train ] && mkdir train
[ ! -d test ] && mkdir test
tree .
```

```
.
├── test
└── train

2 directories, 0 files
```

TRAIN DATA

We use the Free Spoken Digit Dataset FSDD as training set. For each of the 6 speakers in the dataset, the digits 0 to 9 were retrieved by running:

```
./src/utils/get_train_data.sh
```

These files are locally stored in the current project folder under `data/train/raw` .

TEST DATA

Two people volunteered to take on the speakers s_1 and s_2. Both speakers are male and carry a Dutch accent.

The recordings were made and cut into seperate digits by following the guide left by the makers of the FSDD for adding new data.

These files are locally stores in the current project folder under `data/test/raw`

Step 3. Transform data

To follow along with the Kaldi tutorial for Dummies, the data will be transformed to form sequences of 3 digits: __\.wav.

To make sure that every digit appears at least once and have files to compare across speakers, we'll create 4 common files for each of the speakers:

`0_1_2.wav` , `3_4_5.wav` , `6_7_8.wav` and `9_0_1.wav`

In total, 10 files for each of the speakers of the train and test set, including the ones previously mentioned.

This is done by running the following commands:

```
./src/utils/transform_data.sh train/raw train/final
./src/utils/transform_data.sh test/raw test/final
```

The newly created files for the train set are stored in the current project folder: `data/train/final` . And the concatenated files for the test set are stored in: `data/test/final` .

Step 4. Move data to digits folder

After transforming the data, the data is now ready to be added to the Kaldi project folder `digits` for both the test and train set.

In [ ]:
```
%%bash -s "$main_dir"

cp -Rp data/train/final/* $1/digits_audio/train
cp -Rp data/test/final/*  $1/digits_audio/test
```

Step 5. Analyze test data

Though the files for the test data was created following the descriptions made by the makers of the FSDD set, looking at some of the measures of central tendancy could show whether for example the duration of the audio files is similar to that of the train set.

CREATE DATAFRAME TO EASILY COMPARE/ANALYZE/RETRIEVE FILES

To later analyze, compare and visualize files a `pandas.MultiIndex` will be used:

			samples	duration
speaker	**set**	**filename**		
jackson	train	0_1_2	[..]	1.23
...	...	...	...	...

In [ ]:
```
df_files = pd.DataFrame({'speaker':[], 'set':[], 'filename':[], 'samples'
df_files = df_files.set_index(['speaker', 'set', 'filename'])
df_files
```

Out[ ]:

			samples	duration
speaker	**set**	**filename**		

Add each of the audio files for each speaker to the dataframe. Since all audio is sampled at 8kHz, the files will be read and loaded using this sample rate.

In [ ]:
```
update_file_frame(df=df_files)
```

After filling the frame, it can be seen that all speakers and their files have been accounted for.

In [ ]:
```
df_files
```

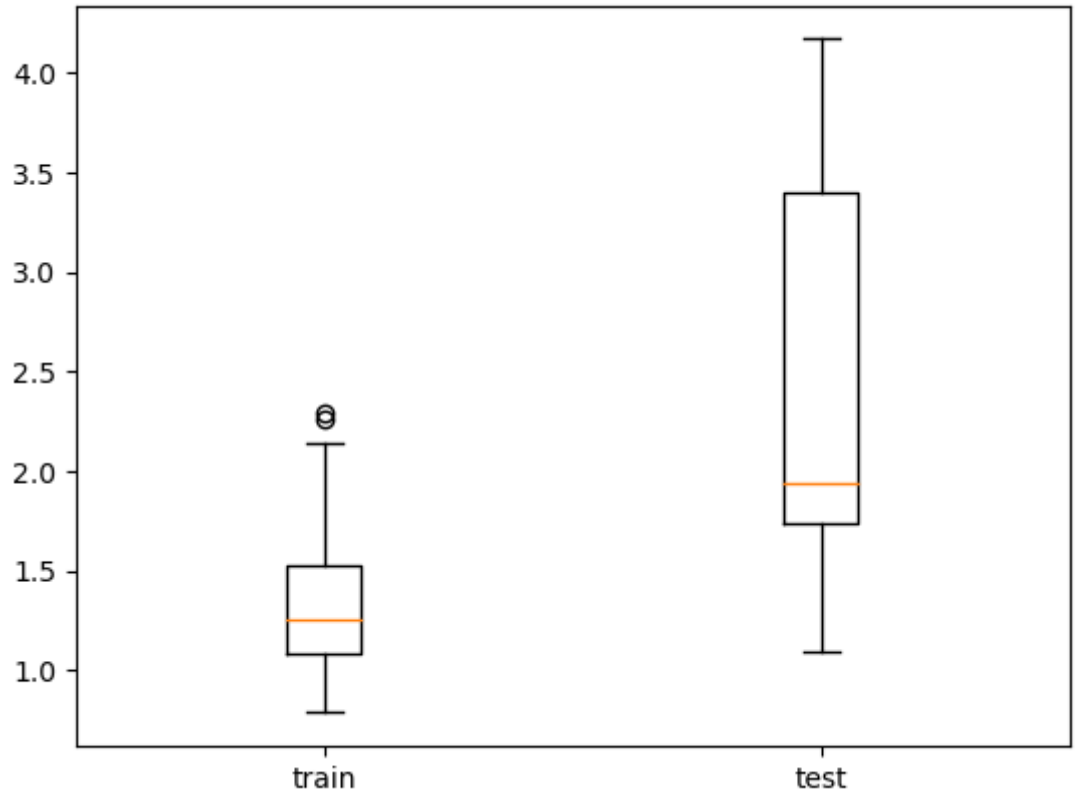
Out[ ]:

			samples	duration
speaker	**set**	**filename**		
jackson	**train**	**6_7_8.wav**	[0.007354736328125, -0.01104736328125, 0.01040...	1.607000
		9_0_1.wav	[-0.009765625, -0.00970458984375, -0.009857177...	1.764125
		2_8_3.wav	[-0.0128173828125, -0.01788330078125, 0.008117...	1.331500
		0_3_5.wav	[-0.011260986328125, -0.013153076171875, -0.01...	1.553500
		7_9_9.wav	[-0.00970458984375, 0.002349853515625, 0.00036...	1.638875
...	**...**	**...**	...	...
s1	**test**	**9_5_6.wav**	[0.005706787109375, 0.00518798828125, 0.005340...	1.821375
		3_4_5.wav	[0.0035400390625, 0.003509521484375, 0.0044555...	1.811000
		8_1_1.wav	[-0.003997802734375, -0.003936767578125, -0.00...	1.975375
		0_1_2.wav	[-0.00341796875, -0.000732421875, -0.002319335...	1.098500
		5_0_1.wav	[0.00543212890625, 0.005645751953125, 0.005157...	1.902500

80 rows × 2 columns

Comparing the different measures of central tendancy in the duration for the audio files for both the train and test set, shows whether the files are comparable in size.

In [ ]:
```
plt.boxplot([df_files.xs('train', level=1)["duration"].values, df_files.x
plt.show()
```



> As can be seen from the measurements of central tendancy like the min, mean, q3 and max values, the test set has an overall longer duration. One of the causes that could lead to this problem is the cuts that the FSDD script for adding data does to create singleton digit audio files. This could cause pauses to become very short or very long if the speech was not done in a constant manner. This also means that not necessarily both speakers in the test set have such long pauses. To first see whether it is necessary to look at both speakers, first examine the measurements of central tendancy for both speakers.

In [ ]:
```
df_files.loc["s1"].describe()
```

Out[ ]:

	duration
count	10.000000
mean	1.594600
std	0.318079
min	1.098500
25%	1.325594
50%	1.673812
75%	1.851281
max	1.975375

> As suggested earlier, it seems like the extreme distribution differences in duration are not caused by both speakers. From the central measurements it can be seen that they closely resemble that of the train set. Therefore, further analyzing the speaker s_1 is not necessary.

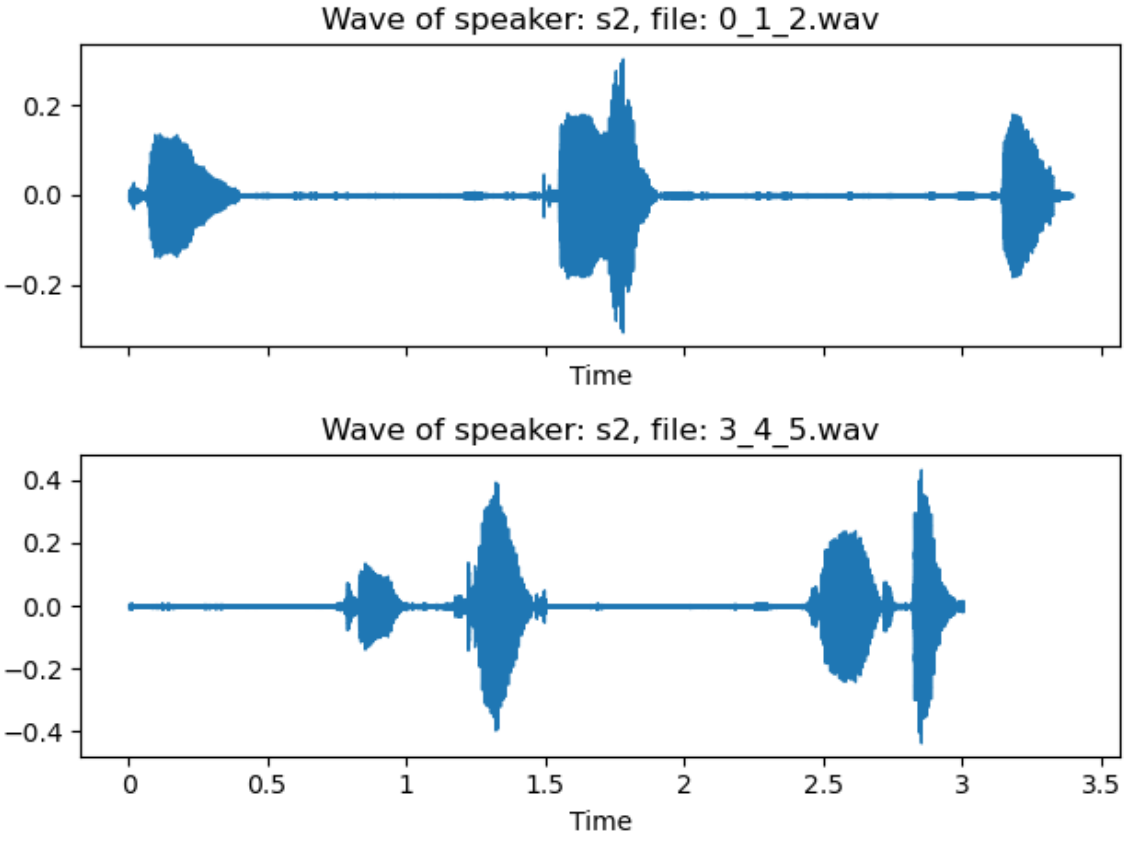
In [ ]:
```
df_files.loc["s2"].describe()
```

Out[ ]:

	duration
count	10.000000
mean	3.278837
std	0.566486
min	1.893375
25%	3.332000
50%	3.396375
75%	3.396375
max	4.168625

> This shows that the cause for the shift in central measurements are caused by the audio files of the speaker s_2. The minimum duration is almost the maximum duration of the train set and most of the files have a longer duration as the max duration of the train set. Therefore further analayzing the files of this speaker is necessary to conform to the speech audio files of the test set.

In [ ]:
```
fig, ax = plt.subplots(nrows=2, sharex=True)
librosa.display.waveshow(
    y=np.array(df_files.loc[("s2", "test", "0_1_2.wav"), ('samples')], dt
    sr=8000,
    ax=ax[0]
)
ax[0].set_title('Wave of speaker: s2, file: 0_1_2.wav')
librosa.display.waveshow(
    y=np.array(df_files.loc[("s2", "test", "3_4_5.wav"), ('samples')], dt
    sr=8000,
    ax=ax[1]
)
ax[1].set_title('Wave of speaker: s2, file: 3_4_5.wav')
fig.tight_layout()
plt.show()
```



> This shows that a very large portion of each file is probably near silence. Some silences take up to 1 second between spoken digits. To conform to the general duration of the train set, shortening large portions of silence is sufficient.

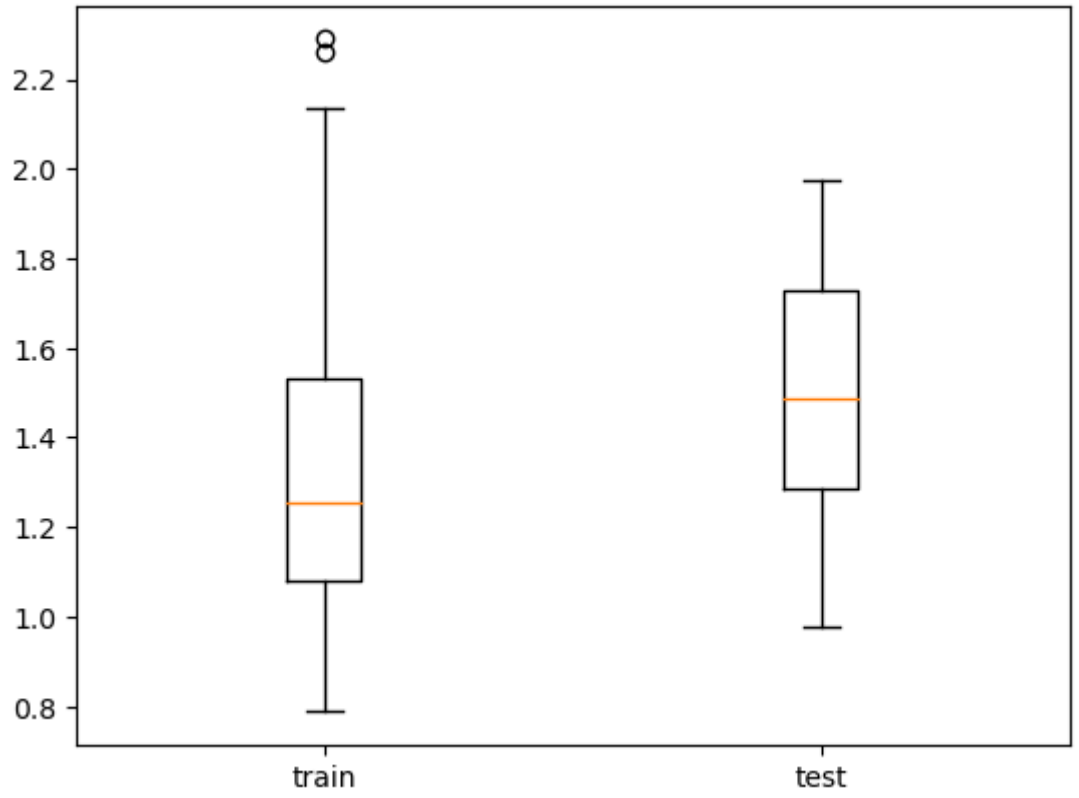
In [ ]:
```
%%bash -s "$main_dir"

cd $1/digits_audio/test/s2

for file in *;do # Shorten silences longer than 0.5 seconds to 0.3 secona
    sox $file ${file%".wav"}"_temp.wav" silence -l 1 0.1 1% -1 0.3 1%
    rm $file
    mv ${file%".wav"}"_temp.wav" $file
done
```

Since the files have been changed, the files in the dataframe need to be updated.

In [ ]:
```
update_file_frame(df_files)
```

Now, looking at the distributions of the durations of speech audio files for both the `test` and `train` set:

In [ ]:
```
plt.boxplot([df_files.xs('train', level=1)["duration"].values, df_files.x
plt.show()
```



> The range of duration lays nearer to that of the `train` set and thus is a better representation of input fitting the eventual trained models' data.

Step 6. Acoustic data

After gathering and transforming the audio data, the next step will be creating text files for, for example the transcription and which files belongs to which speaker.

Step 6.1 Setup folders for data

In the project folder of `digits` create another folder named `data`. In this folder, again create a `test` and `train` folder.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
[ ! -d data ] && mkdir data
cd data
[ ! -d train ] && mkdir train
[ ! -d test ] && mkdir test
tree
```

```
.
├── test
└── train

2 directories, 0 files
```

The following files need to be created for both the `train` and `test` folder of `data`. However, to prevent duplicating work, all files will first be made for the `train` folder after which the created files will be copy pasted into the `test` folder.

Step 6.2 Create spk2gender file

As the name suggests this file maps a speaker to their gender. Each speaker (ID) gets mapped to their gender (f/m).

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/train
test -e spk2gender || cat > spk2gender << EOF
george m
jackson m
lucas m
nicolas m
s1 m
s2 m
theo m
yweweler m
EOF
cd ..;tree
```

```
.
├── test
└── train
    └── spk2gender

2 directories, 1 file
```

Step 6.3 Create wav.scp file

The `wav.scp` file connects every utterance to an audio file containing the utterance.

In [ ]:
```
%%bash
./src/utils/generate_wav_file.sh
```

Step 6.4 Create folder `local`

In the data folder create a new folder named `local`.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data

[ ! -d local ] && mkdir local
tree
```

```
.
├── local
├── test
└── train
    ├── spk2gender
    └── wav.scp

3 directories, 2 files
```

Step 6.4 Create text, utt2spk and corpus file

The `text` file maps each utterance ID to their transcription, while the corpus keeps all the unique utterances from the transcriptions. Because the corpus contains every unique utterance that is first transcribed in the text file, it is easier to concatenate this to the corpus file during transcribing the audio for the text file. However, the corpus file only appears in the newly made folder in `digits/data/local`. The `utt2spk` file contains the utterance ID along with the transcription of the utterance. Because the utterance ID will be used for all these files it is easier to create this alongside the other files.

In [ ]:
```
%%bash
./src/utils/generate_text_utt2spk_corpus_file.sh
```

Step 6.5 Sort acoustic files

Kaldi works with sorted audio files. Therefore, all the audio files need to be sorted.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/train
for f in *;do
    echo "$(sort $f)" >| $f
done
```

Step 6.6 Copy files

Now that all the audio files have been created for the `train` set of the data folder, we can copy paste these to the `test` folder of data.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/
cp -R train/. test/
tree
```

```
.
├── local
│   └── corpus.txt
├── test
│   ├── spk2gender
│   ├── text
│   ├── utt2spk
│   └── wav.scp
└── train
    ├── spk2gender
    ├── text
    ├── utt2spk
    └── wav.scp

3 directories, 9 files
```

Step 7. Language Data

The language data includes files for Kaldi that describe the lexicon and other phonemes used for building for among others the HMM.

For this project the following created files are copied from the files shared by Kaldi in the Dummies tutorial

Step 7.1 Setup folders

Create the folder `dict` in the earlier created folder `data/local` . Then, in the newly created folder `dict` again create a `test` and `train` folder.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/local
[ -d dict ] || (mkdir dict;cd dict;mkdir test train);tree
```

```
.
├── corpus.txt
└── dict
    ├── test
    └── train

3 directories, 1 file
```

Step 7.2 Create lexicon file

In the folder of `dict` create the lexicon.txt file where every word from the dictionary with the possible phone transcriptions are noted as
\<word> <phone 1> <phone 2>.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/local/dict
test -e lexicon.txt || cat > lexicon.txt << EOF
!SIL sil
<UNK> spn
eight ey t
five f ay v
four f ao r
nine n ay n
one hh w ah n
one w ah n
seven s eh v ah n
six s ih k s
three th r iy
two t uw
zero z ih r ow
zero z iy r ow
EOF
cd ..;tree
```

```
.
├── corpus.txt
└── dict
    ├── lexicon.txt
    ├── test
    └── train

3 directories, 2 files
```

Step 7.3 Create nonsilence_phones file

This file lists other phones that are present in the audio that are not always part of a phone transcription.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/local/dict
test -e nonsilence_phones.txt || cat > nonsilence_phones.txt << EOF
ah
ao
ay
eh
ey
f
hh
ih
iy
k
n
ow
r
s
t
th
uw
w
v
z
EOF
cd ..;tree
```

```
.
├── corpus.txt
└── dict
    ├── lexicon.txt
    ├── nonsilence_phones.txt
    ├── test
    └── train

3 directories, 3 files
```

Step 7.4 Create silence_phones file

The following phones are silent.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/local/dict
test -e silence_phones.txt || cat > silence_phones.txt << EOF
sil
spn
EOF
cd ..;tree
```

```
.
├── corpus.txt
└── dict
    ├── lexicon.txt
    ├── nonsilence_phones.txt
    ├── silence_phones.txt
    ├── test
    └── train

3 directories, 4 files
```

Step 7.5 Create optional_silence file

Phones listed in this file are optional silence phones.

In [ ]:
```
%%bash -s "$main_dir"

cd $1/data/local/dict
test -e optional_silence.txt || cat > optional_silence.txt << EOF
sil
EOF
cd ..;tree
```

```
.
├── corpus.txt
└── dict
    ├── lexicon.txt
    ├── nonsilence_phones.txt
    ├── optional_silence.txt
    ├── silence_phones.txt
    ├── test
    └── train

3 directories, 5 files
```

Step 8. Finalizing setup

Last steps needed to finalize the setup.

Step 8.1 Attaching tools

From egs/wsj/s5 copy the folders `utils` and `steps` along with their content into the `digits` directory.

In [ ]:
```
%%bash

cd ../git_workspace/kaldi/egs
([ -d utils ] && [ -d steps ]) || cp -r wsj/s5/{utils,steps} digits
cd digits;tree -x
```

```
.
├── data
├── digits_audio
├── steps
└── utils

4 directories, 0 files
```

Step 8.2 Add scoring script

To get the decoding results, the scoring script must be added. from `egs/voxforge/s5/local` copy the `score.sh` script into a new folder `egs/digits/local` .

In [ ]:
```
%%bash -s "$main_dir"

# Create the folder `local` in the directory of `digits`
cd $1
[ -d local ] || mkdir local;
cd local
[ -f score.sh ] || cp ../../voxforge/s5/local/score.sh score.sh
```

Step 8.3 Create configuration files

In the `digits` folder, create a new folder named `conf`. Inside this folder create the files: `decode.config` an `mfcc.config`.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
[ -d conf ] || mkdir conf;
cd conf
test -e decode.config || cat > decode.config << EOF
first_beam=10.0
beam=13.0
lattice_beam=6.0
EOF
test -e mfcc.conf || cat > mfcc.conf << EOF
--use-energy=false
--allow-upsample=true
EOF
tree
```

```
.
├── decode.config
└── mfcc.conf

0 directories, 2 files
```

Step 8.4 Running scripts

To create an ASR model and test the created model, a couple of running scripts must first be created. Following the tutorial, 2 different training methods are used to create an ASR model and see the differences in decoding results. The first method is called MONO: monophone training. The second method is called TRI1: simple triphone training.

Then to create the running scripts create the `cmd.sh`, `path.sh` and `run.sh` files in the `digits` directory.

Step 8.4.1 Create the `cmd.sh` script

In [ ]:
```
%%bash -s "$main_dir"

cd $1
test -e cmd.sh || cat > cmd.sh << EOF
# Setting local system jobs (local CPU - no external clusters)
export train_cmd=run.pl
export decode_cmd=run.pl
EOF
```

Step 8.4.2 Create the `path.sh` script

These set all the relevant paths to useful tools and audio data, used by train, test e.g. scripts.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
test -e path.sh || cat > path.sh << EOF
#!/bin/bash

# Defining Kaldi root directory
export KALDI_ROOT=\`pwd\`/../..

# Setting paths to useful tools
export PATH=\$PWD/utils/:\$KALDI_ROOT/src/bin:\$KALDI_ROOT/tools/openfst/

# Defining audio data directory (modify it for your installation director
export DATA_ROOT="/home/ScoobyDoobyDoo/kaldi-trunk/egs/digits/digits_audi

# Variable that stores path to MITLM library
export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:\$(pwd)/tools/mitlm-svn/lib

# Enable SRILM
. \$KALDI_ROOT/tools/env.sh

# Variable needed for proper data sorting
export LC_ALL=C
EOF
```

Step 8.4.3 Create the `run.sh` script

Create the script that will build, train and test the ASR model.

1. Start by creating the file and setting the paths to the useful tools and variables for the terminal.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
test -e run.sh || cat > run.sh << EOF
#!/bin/bash

. ./path.sh || exit 1
. ./cmd.sh || exit 1
EOF
# Make file executable
chmod u+x run.sh
tree -x
```

```
.
├── cmd.sh
├── conf
├── data
├── digits_audio
├── local
├── path.sh
├── run.sh
├── steps
└── utils

6 directories, 3 files
```

1. Set global variables for the number of parallel jobs the building, training and testing gets as well as a number for the n-gram language model.

For a small dataset as this, the overall time complexion as well as space complexity is relatively small compared to full language corpuses meaning that using 1 thread: the main thread, is enough. This also applies to the language model. A total of 6 speakers are used for the training set, and each speaker utters 10 sentences. Of these sentences at least 4 of them are common across all speakers. This gives, assuming the other files are all unique, 4 (common files) $\times$ 6 (speakers) + 6 (unique files) = 40 unique 3-digit sequences. This is a small corpus and knowing that the unique set of words that can be used is $|\{0,..,9\}| = 10$ makes a 1-gram LM sufficient enough to describe the multitude of the 3 sequence digits.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
nj=1
lm_order=1
""" >> run.sh
```

1. If one wants to test with other arguments, the Kaldi for Dummies tutorial adds a safety mechanism to check the arguments added to the script call. This also means that when the `run.sh` is called more than once, the previously created data must be removed.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
. utils/parse_options.sh || exit 1
[[ \$# -ge 1 ]] && { echo \"Wrong arguments!\"; exit 1; }

rm -rf exp mfcc data/train/spk2utt data/train/cmvn.scp data/train/feats.s
""" >> run.sh
```

1. Prepare the last files for the acoustic data where each speaker is mapped to their utterance. A script to do this is already provided in the earlier copied `utils` directory.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== PREPARING ACOUSTIC DATA =====\"
echo

utils/utt2spk_to_spk2utt.pl data/train/utt2spk > data/train/spk2utt
utils/utt2spk_to_spk2utt.pl data/test/utt2spk > data/test/spk2utt
""" >> run.sh
```

1. Extract the MFCC feature vectors from the audio files for the train and test set. And then compute the cepstral mean and variance statistics per speaker.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== FEATURES EXTRACTION =====\"
echo

steps/make_mfcc.sh --nj \$nj --cmd \"\$train_cmd\" data/train exp/make_mf
steps/make_mfcc.sh --nj \$nj --cmd \"\$train_cmd\" data/test exp/make_mfc

steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train \$mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test \$mfccdir
""" >> run.sh
```

1. Prepare the last file of the language data.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== PREPARING LANGUAGE DATA =====\"
echo

utils/prepare_lang.sh data/local/dict \"<UNK>\" data/local/lang data/lang
""" >> run.sh
```

1. Create a language model using the ARPAbet lexicon created earlier.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== LANGUAGE MODEL CREATION =====\"
echo \"===== MAKING lm.arpa =====\"
echo

loc=`which ngram-count`;
if [ -z \$loc ]; then
        if uname -a | grep 64 >/dev/null; then
                sdir=\$KALDI_ROOT/tools/srilm/bin/i686-m64
        else
                sdir=\$KALDI_ROOT/tools/srilm/bin/i686
        fi
        if [ -f \$sdir/ngram-count ]; then
                echo \"Using SRILM language modelling tool from \$sdir\"
                export PATH=\$PATH:\$sdir
        else
                echo \"SRILM toolkit is probably not installed.
                      Instructions: tools/install_srilm.sh\"
                exit 1
        fi
fi

local=data/local
mkdir \$local/tmp
ngram-count -order \$lm_order -write-vocab \$local/tmp/vocab-full.txt -wb
""" >> run.sh
```

1. Create a Finite State Transducer of the language model.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== MAKING G.fst =====\"
echo

lang=data/lang
cat \$local/tmp/lm.arpa | arpa2fst - | fstprint | utils/eps2disambig.pl |
""" >> run.sh
```

1. Train the monophone Acoustic model.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== MONO TRAINING =====\"
echo

steps/train_mono.sh --nj \$nj --cmd "\$train_cmd" data/train data/lang ex
""" >> run.sh
```

1. Decode the test set on the MONO AM.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== MONO DECODING =====\"
echo

utils/mkgraph.sh --mono data/lang exp/mono exp/mono/graph || exit 1
steps/decode.sh --config conf/decode.config --nj \$nj --cmd \"\$decode_cm
""" >> run.sh
```

1. Align the latices of the MONO AM.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== MONO ALIGNMENT =====\"
echo

steps/align_si.sh --nj \$nj --cmd \"\$train_cmd\" data/train data/lang ex
""" >> run.sh
```

1. Train the first triphone pass AM.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== TRI1 (first triphone pass) TRAINING =====\"
echo

steps/train_deltas.sh --cmd \"\$train_cmd\" 2000 11000 data/train data/la
""" >> run.sh
```

1. Decode the test set using the TRI1 AM.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== TRI1 (first triphone pass) DECODING =====\"
echo

utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph || exit 1
steps/decode.sh --config conf/decode.config --nj \$nj --cmd \"\$decode_cm
""" >> run.sh
```

1. Print a note to show that the script has succesfully finished.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

echo """
echo
echo \"===== run.sh script is finished =====\"
echo
""" >> run.sh
```

Step 8.5 Execute `run.sh`

To build, train and evaluate the MONO and TRI1 training, run the `run.sh` script.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
./run.sh
```

```
===== PREPARING ACOUSTIC DATA =====


===== FEATURES EXTRACTION =====

steps/make_mfcc.sh --nj 1 --cmd run.pl data/train exp/make_mfcc/train
utils/validate_data_dir.sh: Successfully validated data-directory data/tr
ain
steps/make_mfcc.sh: [info]: no segments file exists: assuming wav.scp ind
exed by utterance.
steps/make_mfcc.sh: Succeeded creating MFCC features for train
steps/make_mfcc.sh --nj 1 --cmd run.pl data/test exp/make_mfcc/test
utils/validate_data_dir.sh: Successfully validated data-directory data/te
st
steps/make_mfcc.sh: [info]: no segments file exists: assuming wav.scp ind
exed by utterance.
steps/make_mfcc.sh: Succeeded creating MFCC features for test
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train
Succeeded creating CMVN stats for train
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test
Succeeded creating CMVN stats for test

===== PREPARING LANGUAGE DATA =====

utils/prepare_lang.sh data/local/dict <UNK> data/local/lang data/lang
Checking data/local/dict/silence_phones.txt ...
--> reading data/local/dict/silence_phones.txt
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> data/local/dict/silence_phones.txt is OK

Checking data/local/dict/optional_silence.txt ...
--> reading data/local/dict/optional_silence.txt
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> data/local/dict/optional_silence.txt is OK

Checking data/local/dict/nonsilence_phones.txt ...
--> reading data/local/dict/nonsilence_phones.txt
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> data/local/dict/nonsilence_phones.txt is OK

Checking disjoint: silence_phones.txt, nonsilence_phones.txt
--> disjoint property is OK.

Checking data/local/dict/lexicon.txt
--> reading data/local/dict/lexicon.txt
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> data/local/dict/lexicon.txt is OK

Checking data/local/dict/extra_questions.txt ...
--> data/local/dict/extra_questions.txt is empty (this is OK)
--> SUCCESS [validating dictionary directory data/local/dict]

**Creating data/local/dict/lexiconp.txt from data/local/dict/lexicon.txt
prepare_lang.sh: validating output directory
utils/validate_lang.pl data/lang
Checking existence of separator file
separator file data/lang/subword_separator.txt is empty or does not exis
t, deal in word case.
Checking data/lang/phones.txt ...
```

```
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> data/lang/phones.txt is OK

Checking words.txt: #0 ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> data/lang/words.txt is OK

Checking disjoint: silence.txt, nonsilence.txt, disambig.txt ...
--> silence.txt and nonsilence.txt are disjoint
--> silence.txt and disambig.txt are disjoint
--> disambig.txt and nonsilence.txt are disjoint
--> disjoint property is OK

Checking sumation: silence.txt, nonsilence.txt, disambig.txt ...
--> found no unexplainable phones in phones.txt

Checking data/lang/phones/context_indep.{txt, int, csl} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 10 entry/entries in data/lang/phones/context_indep.txt
--> data/lang/phones/context_indep.int corresponds to data/lang/phones/co
ntext_indep.txt
--> data/lang/phones/context_indep.csl corresponds to data/lang/phones/co
ntext_indep.txt
--> data/lang/phones/context_indep.{txt, int, csl} are OK

Checking data/lang/phones/nonsilence.{txt, int, csl} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 80 entry/entries in data/lang/phones/nonsilence.txt
--> data/lang/phones/nonsilence.int corresponds to data/lang/phones/nonsi
lence.txt
--> data/lang/phones/nonsilence.csl corresponds to data/lang/phones/nonsi
lence.txt
--> data/lang/phones/nonsilence.{txt, int, csl} are OK

Checking data/lang/phones/silence.{txt, int, csl} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 10 entry/entries in data/lang/phones/silence.txt
--> data/lang/phones/silence.int corresponds to data/lang/phones/silence.
txt
--> data/lang/phones/silence.csl corresponds to data/lang/phones/silence.
txt
--> data/lang/phones/silence.{txt, int, csl} are OK

Checking data/lang/phones/optional_silence.{txt, int, csl} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 1 entry/entries in data/lang/phones/optional_silence.txt
--> data/lang/phones/optional_silence.int corresponds to data/lang/phones
/optional_silence.txt
--> data/lang/phones/optional_silence.csl corresponds to data/lang/phones
/optional_silence.txt
--> data/lang/phones/optional_silence.{txt, int, csl} are OK

Checking data/lang/phones/disambig.{txt, int, csl} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 2 entry/entries in data/lang/phones/disambig.txt
--> data/lang/phones/disambig.int corresponds to data/lang/phones/disambi
```

```
g.txt
--> data/lang/phones/disambig.csl corresponds to data/lang/phones/disambi
g.txt
--> data/lang/phones/disambig.{txt, int, csl} are OK

Checking data/lang/phones/roots.{txt, int} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 22 entry/entries in data/lang/phones/roots.txt
--> data/lang/phones/roots.int corresponds to data/lang/phones/roots.txt
--> data/lang/phones/roots.{txt, int} are OK

Checking data/lang/phones/sets.{txt, int} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 22 entry/entries in data/lang/phones/sets.txt
--> data/lang/phones/sets.int corresponds to data/lang/phones/sets.txt
--> data/lang/phones/sets.{txt, int} are OK

Checking data/lang/phones/extra_questions.{txt, int} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 9 entry/entries in data/lang/phones/extra_questions.txt
--> data/lang/phones/extra_questions.int corresponds to data/lang/phones/
extra_questions.txt
--> data/lang/phones/extra_questions.{txt, int} are OK

Checking data/lang/phones/word_boundary.{txt, int} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 90 entry/entries in data/lang/phones/word_boundary.txt
--> data/lang/phones/word_boundary.int corresponds to data/lang/phones/wo
rd_boundary.txt
--> data/lang/phones/word_boundary.{txt, int} are OK

Checking optional_silence.txt ...
--> reading data/lang/phones/optional_silence.txt
--> data/lang/phones/optional_silence.txt is OK

Checking disambiguation symbols: #0 and #1
--> data/lang/phones/disambig.txt has "#0" and "#1"
--> data/lang/phones/disambig.txt is OK

Checking topo ...

Checking word_boundary.txt: silence.txt, nonsilence.txt, disambig.txt ...
--> data/lang/phones/word_boundary.txt doesn't include disambiguation sym
bols
--> data/lang/phones/word_boundary.txt is the union of nonsilence.txt and
silence.txt
--> data/lang/phones/word_boundary.txt is OK

Checking word-level disambiguation symbols...
--> data/lang/phones/wdisambig.txt exists (newer prepare_lang.sh)
Checking word_boundary.int and disambig.int
--> generating a 97 word/subword sequence
--> resulting phone sequence from L.fst corresponds to the word sequence
--> L.fst is OK
--> generating a 30 word/subword sequence
--> resulting phone sequence from L_disambig.fst corresponds to the word
sequence
--> L_disambig.fst is OK
```

```
Checking data/lang/oov.{txt, int} ...
--> text seems to be UTF-8 or ASCII, checking whitespaces
--> text contains only allowed whitespaces
--> 1 entry/entries in data/lang/oov.txt
--> data/lang/oov.int corresponds to data/lang/oov.txt
--> data/lang/oov.{txt, int} are OK

--> data/lang/L.fst is olabel sorted
--> data/lang/L_disambig.fst is olabel sorted
--> SUCCESS [validating lang directory data/lang]

===== LANGUAGE MODEL CREATION =====
===== MAKING lm.arpa =====

Using SRILM language modelling tool from /home/scoobydoobydoo/Documents/g
it_workspace/kaldi/egs/digits/../../tools/srilm/bin/i686-m64

===== MAKING G.fst =====


===== MONO TRAINING =====

steps/train_mono.sh --nj 1 --cmd run.pl data/train data/lang exp/mono
steps/train_mono.sh: Initializing monophone system.
steps/train_mono.sh: Compiling training graphs
steps/train_mono.sh: Aligning data equally (pass 0)
steps/train_mono.sh: Pass 1
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 2
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 3
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 4
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 5
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 6
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 7
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 8
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 9
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 10
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 11
steps/train_mono.sh: Pass 12
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 13
steps/train_mono.sh: Pass 14
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 15
steps/train_mono.sh: Pass 16
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 17
steps/train_mono.sh: Pass 18
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 19
steps/train_mono.sh: Pass 20
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 21
steps/train_mono.sh: Pass 22
```

```
steps/train_mono.sh: Pass 23
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 24
steps/train_mono.sh: Pass 25
steps/train_mono.sh: Pass 26
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 27
steps/train_mono.sh: Pass 28
steps/train_mono.sh: Pass 29
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 30
steps/train_mono.sh: Pass 31
steps/train_mono.sh: Pass 32
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 33
steps/train_mono.sh: Pass 34
steps/train_mono.sh: Pass 35
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 36
steps/train_mono.sh: Pass 37
steps/train_mono.sh: Pass 38
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 39
steps/diagnostic/analyze_alignments.sh --cmd run.pl data/lang exp/mono
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
18.75% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
10.0% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_alignments.sh: see stats in exp/mono/log/analyze
_alignments.log
250 warnings in exp/mono/log/update.*.log
21 warnings in exp/mono/log/align.*.*.log
2 warnings in exp/mono/log/analyze_alignments.log
exp/mono: nj=1 align prob=-77.96 over 0.03h [retry=0.0%, fail=0.0%] state
s=70 gauss=515
steps/train_mono.sh: Done training monophone system in exp/mono

===== MONO DECODING =====

WARNING: the --mono, --left-biphone and --quinphone options are now depre
cated and ignored.
-0.0424209 -0.0426144
[info]: LG not stochastic.
-0.0424209 -0.0426144
[info]: CLG not stochastic.
0.000381989 -0.0429688
HCLGa is not stochastic
steps/decode.sh --config conf/decode.config --nj 1 --cmd run.pl exp/mono/
graph data/test exp/mono/decode
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/mono/graph exp/mono/dec
ode
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
18.75% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
10.0% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_lats.sh: see stats in exp/mono/decode/log/analyz
e_alignments.log
Overall, lattice depth (10,50,90-percentile)=(1,1,1) and mean=1.0
steps/diagnostic/analyze_lats.sh: see stats in exp/mono/decode/log/analyz
e_lattice_depth_stats.log
exp/mono/decode/wer_10
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
```

```
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_11
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_12
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_13
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_14
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_15
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_16
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_17
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_7
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_8
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/mono/decode/wer_9
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]

===== MONO ALIGNMENT =====

steps/align_si.sh --nj 1 --cmd run.pl data/train data/lang exp/mono exp/m
ono_ali
steps/align_si.sh: feature type is delta
steps/align_si.sh: aligning data in data/train using model from exp/mono,
putting alignments in exp/mono_ali
steps/diagnostic/analyze_alignments.sh --cmd run.pl data/lang exp/mono_al
i
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
18.75% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
10.0% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_alignments.sh: see stats in exp/mono_ali/log/ana
lyze_alignments.log
steps/align_si.sh: done aligning data.

===== TRI1 (first triphone pass) TRAINING =====

steps/train_deltas.sh --cmd run.pl 2000 11000 data/train data/lang exp/mo
no_ali exp/tri1
steps/train_deltas.sh: accumulating tree stats
steps/train_deltas.sh: getting questions for tree-building, via clusterin
g
steps/train_deltas.sh: building the tree
WARNING (gmm-init-model[5.5.1057~1-be222]:InitAmGmm():gmm-init-model.cc:5
5) Tree has pdf-id 1 with no stats; corresponding phone list: 6 7 8 9 10
** The warnings above about 'no stats' generally mean you have phones **
** (or groups of phones) in your phone set that had no corresponding dat
a. **
** You should probably figure out whether something went wrong, **
```

```
** or whether your data just doesn't happen to have examples of those **
** phones. **
steps/train_deltas.sh: converting alignments from exp/mono_ali to use cur
rent tree
steps/train_deltas.sh: compiling graphs of transcripts
steps/train_deltas.sh: training pass 1
steps/train_deltas.sh: training pass 2
steps/train_deltas.sh: training pass 3
steps/train_deltas.sh: training pass 4
steps/train_deltas.sh: training pass 5
steps/train_deltas.sh: training pass 6
steps/train_deltas.sh: training pass 7
steps/train_deltas.sh: training pass 8
steps/train_deltas.sh: training pass 9
steps/train_deltas.sh: training pass 10
steps/train_deltas.sh: aligning data
steps/train_deltas.sh: training pass 11
steps/train_deltas.sh: training pass 12
steps/train_deltas.sh: training pass 13
steps/train_deltas.sh: training pass 14
steps/train_deltas.sh: training pass 15
steps/train_deltas.sh: training pass 16
steps/train_deltas.sh: training pass 17
steps/train_deltas.sh: training pass 18
steps/train_deltas.sh: training pass 19
steps/train_deltas.sh: training pass 20
steps/train_deltas.sh: aligning data
steps/train_deltas.sh: training pass 21
steps/train_deltas.sh: training pass 22
steps/train_deltas.sh: training pass 23
steps/train_deltas.sh: training pass 24
steps/train_deltas.sh: training pass 25
steps/train_deltas.sh: training pass 26
steps/train_deltas.sh: training pass 27
steps/train_deltas.sh: training pass 28
steps/train_deltas.sh: training pass 29
steps/train_deltas.sh: training pass 30
steps/train_deltas.sh: aligning data
steps/train_deltas.sh: training pass 31
steps/train_deltas.sh: training pass 32
steps/train_deltas.sh: training pass 33
steps/train_deltas.sh: training pass 34
steps/diagnostic/analyze_alignments.sh --cmd run.pl data/lang exp/tri1
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
18.75% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
10.0% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_alignments.sh: see stats in exp/tri1/log/analyze
_alignments.log
321 warnings in exp/tri1/log/update.*.log
2 warnings in exp/tri1/log/analyze_alignments.log
3 warnings in exp/tri1/log/align.*.*.log
1 warnings in exp/tri1/log/questions.log
1 warnings in exp/tri1/log/build_tree.log
65 warnings in exp/tri1/log/init_model.log
1 warnings in exp/tri1/log/mixup.log
exp/tri1: nj=1 align prob=-77.68 over 0.03h [retry=0.0%, fail=0.0%] state
s=104 gauss=495 tree-impr=7.24
steps/train_deltas.sh: Done training system with delta+delta-delta featur
es in exp/tri1

===== TRI1 (first triphone pass) DECODING =====
```

```
0 -0.0426144
[info]: CLG not stochastic.
0.000358309 -0.100727
HCLGa is not stochastic
steps/decode.sh --config conf/decode.config --nj 1 --cmd run.pl exp/tri1/
graph data/test exp/tri1/decode
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/tri1/graph exp/tri1/dec
ode
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
18.75% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only
10.0% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_lats.sh: see stats in exp/tri1/decode/log/analyz
e_alignments.log
Overall, lattice depth (10,50,90-percentile)=(1,1,1) and mean=1.0
steps/diagnostic/analyze_lats.sh: see stats in exp/tri1/decode/log/analyz
e_lattice_depth_stats.log
exp/tri1/decode/wer_10
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_11
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_12
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_13
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_14
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_15
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_16
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_17
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_7
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_8
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]
exp/tri1/decode/wer_9
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ]
%SER 2.50 [ 2 / 80 ]

===== run.sh script is finished =====
```

```
fstaddselfloops data/lang/phones/wdisambig_phones.int data/lang/phones/wd
isambig_words.int
arpa2fst -
LOG (arpa2fst[5.5.1057~1-be222]:Read():arpa-file-parser.cc:94) Reading \d
ata\ section.
LOG (arpa2fst[5.5.1057~1-be222]:Read():arpa-file-parser.cc:149) Reading \
1-grams: section.
tree-info exp/mono/tree
tree-info exp/mono/tree
fstpushspecial
fsttablecompose data/lang/L_disambig.fst data/lang/G.fst
fstdeterminizestar --use-log=true
fstminimizeencoded
fstisstochastic data/lang/tmp/LG.fst
fstcomposecontext --context-size=1 --central-position=0 --read-disambig-s
yms=data/lang/phones/disambig.int --write-disambig-syms=data/lang/tmp/dis
ambig_ilabels_1_0.int data/lang/tmp/ilabels_1_0.12525 data/lang/tmp/LG.fs
t
fstisstochastic data/lang/tmp/CLG_1_0.fst
make-h-transducer --disambig-syms-out=exp/mono/graph/disambig_tid.int --t
ransition-scale=1.0 data/lang/tmp/ilabels_1_0 exp/mono/tree exp/mono/fina
l.mdl
fstdeterminizestar --use-log=true
fstminimizeencoded
fsttablecompose exp/mono/graph/Ha.fst data/lang/tmp/CLG_1_0.fst
fstrmsymbols exp/mono/graph/disambig_tid.int
fstrmepslocal
fstisstochastic exp/mono/graph/HCLGa.fst
add-self-loops --self-loop-scale=0.1 --reorder=true exp/mono/final.mdl ex
p/mono/graph/HCLGa.fst
tree-info exp/tri1/tree
tree-info exp/tri1/tree
fstcomposecontext --context-size=3 --central-position=1 --read-disambig-s
yms=data/lang/phones/disambig.int --write-disambig-syms=data/lang/tmp/dis
ambig_ilabels_3_1.int data/lang/tmp/ilabels_3_1.14766 data/lang/tmp/LG.fs
t
fstisstochastic data/lang/tmp/CLG_3_1.fst
make-h-transducer --disambig-syms-out=exp/tri1/graph/disambig_tid.int --t
ransition-scale=1.0 data/lang/tmp/ilabels_3_1 exp/tri1/tree exp/tri1/fina
l.mdl
fsttablecompose exp/tri1/graph/Ha.fst data/lang/tmp/CLG_3_1.fst
fstdeterminizestar --use-log=true
fstrmsymbols exp/tri1/graph/disambig_tid.int
fstrmepslocal
fstminimizeencoded
fstisstochastic exp/tri1/graph/HCLGa.fst
add-self-loops --self-loop-scale=0.1 --reorder=true exp/tri1/final.mdl ex
p/tri1/graph/HCLGa.fst
```

Step 9. Model evaluation

There are multiple ways to evaluate the ASR model, such as the transcription accuracy scores but also the time alignment scores. Both give different insights into the working and performance of the ASR model. Where the transcription shows the recognition of the different digits, the time alignment shows how good the model can recognise the exact moments where the word was uttered.

Step 9.1 Transcription accuracy scores

To evaluate the ASR models, the Character, Word and Sentence Error Rate are used. The Character Error Rate (CER) is given by:

$$CER = \frac{I + S + D}{T} \cdot 100\%$$

An insertion (I) stands for an extra character appearing in the transcription where it isn't supposed to. A substitution (S) is when a character is wrongly transcribed. And, contrary to an insertion, deletion (D) stands for removing or not transcribing a character. The sum of these is divided by the total number of characters in the correct transcription (T) and shows the similarity between the transcribed and actual string on character niveau.

The Word Error Rate (WER) shows the similarity between the transcribed string and the actual string on word level and is defined as:

$$CER = \frac{I + S + D}{T} \cdot 100\%$$

However, an insertion (I) now stand for an extra word appearing in the transcription. A substitution (S) stands for wrongly transcribing a word and a deletion (D) stands for not transcribing a word where it is expected. Then, the sum of these errors is divided by the total number of words in the original string (T).

The Sentence Error Rate (SER) shows how many of the transcribed sentences have been correctly transcribed by summing up the sentences that had at least one error (E) and dividing these over the total number of sentences (T):

$$SER = \frac{E}{T} \cdot 100\%$$

MONO

We start by analysing the MONO trained ASR model on their scores.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
# Gather WER scores for mono training
steps/scoring/score_kaldi_wer.sh data/test data/lang exp/mono/decode
# Show WER score of mono training
cat exp/mono/decode/scoring_kaldi/best_wer
# Gather CER score for mono training
steps/scoring/score_kaldi_cer.sh data/test data/lang exp/mono/decode
# Show CER score of mono training
cat exp/mono/decode/scoring_kaldi/best_cer
```

```
steps/scoring/score_kaldi_wer.sh data/test data/lang exp/mono/decode
steps/scoring/score_kaldi_wer.sh: scoring with word insertion penalty=0.
0,0.5,1.0
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ] exp/mono/decode/wer_7_0.0
steps/scoring/score_kaldi_cer.sh data/test data/lang exp/mono/decode
steps/scoring/score_kaldi_cer.sh: scoring with word insertion penalty=0.
0,0.5,1.0
%WER 0.85 [ 8 / 946, 0 ins, 8 del, 0 sub ] exp/mono/decode/cer_7_0.0
```

And from the output shown earlier from running the `run.sh` script, we could see that the SER of the MONO training is 2.50\%. The scores are already near perfect which begs the question whether the TRI1 training on using the alignments of the MONO training can still improve this score.

TRI1

Now looking at the scores of the TRI1 training, we can see if the extra context provided by the left and right phones of the HMM state provided more insight and thus lead to better scores.

In [ ]:
```
%%bash -s "$main_dir"

cd $1
# Gather WER scores for tri1 training
steps/scoring/score_kaldi_wer.sh data/test data/lang exp/tri1/decode
# Show WER score of tri1 training
cat exp/tri1/decode/scoring_kaldi/best_wer
# Gather CER score for tri1 training
steps/scoring/score_kaldi_cer.sh data/test data/lang exp/tri1/decode
# Show CER score of tri1 training
cat exp/tri1/decode/scoring_kaldi/best_cer
```

```
steps/scoring/score_kaldi_wer.sh data/test data/lang exp/tri1/decode
steps/scoring/score_kaldi_wer.sh: scoring with word insertion penalty=0.
0,0.5,1.0
%WER 0.83 [ 2 / 240, 0 ins, 2 del, 0 sub ] exp/tri1/decode/wer_7_0.0
steps/scoring/score_kaldi_cer.sh data/test data/lang exp/tri1/decode
steps/scoring/score_kaldi_cer.sh: scoring with word insertion penalty=0.
0,0.5,1.0
%WER 0.85 [ 8 / 946, 0 ins, 8 del, 0 sub ] exp/tri1/decode/cer_7_0.0
```

As can be seen, the scores have not improved. Though not completely unexpectedly seeing as the dataset is really small and the TRI1 builds forth on the alignments made on the MONO training. This gives us:

	CER	WER	SER
MONO	0.85	0.83	2.50
TRI1	0.85	0.83	2.50

That the ASR is so accurate is actually rather remarkable as we have added numerous speech errors to the audio files. To see what the mistakes are that have been made we first output the best found paths per utterance for each speaker starting with the MONO training.

MONO

In [ ]:
```
%%bash -s "$main_dir"

cd $1

# Create output file for mono training
../../src/latbin/lattice-best-path ark:'gunzip -c exp/mono/decode/lat.1.g
```

```
../../src/latbin/lattice-best-path 'ark:gunzip -c exp/mono/decode/lat.1.g
z |' 'ark,t:| utils/int2sym.pl -f 2- exp/mono/graph/words.txt > mono_outp
ut.txt'
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_0_1_2, best cost 17.3717 + 9867.92 = 9885.3 over 118
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_1_2_9, best cost 17.764 + 11192.4 = 11210.1 over 140
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_2_4_4, best cost 17.5028 + 8864.91 = 8882.41 over 11
8 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_4_5, best cost 19.2752 + 10651.4 = 10670.7 over 14
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_5_8, best cost 18.7793 + 11996.4 = 12015.2 over 15
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_6_4, best cost 19.3774 + 10411.7 = 10431 over 143
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_7_6, best cost 20.8224 + 12669 = 12689.8 over 164
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_6_7_8, best cost 19.785 + 13586.9 = 13606.7 over 167
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_9_0_1, best cost 18.1658 + 11208.2 = 11226.4 over 13
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_9_5_6, best cost 19.0078 + 11929.5 = 11948.5 over 15
8 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_0_1_2, best cost 18.7344 + 12434.3 = 12453.1 over 1
64 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_0_3_5, best cost 18.5092 + 10927.3 = 10945.8 over 1
53 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_2_3_8, best cost 19.321 + 9779.33 = 9798.66 over 13
1 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_2_8_3, best cost 19.4096 + 9781.12 = 9800.52 over 1
31 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_3_4_5, best cost 18.0661 + 9650.77 = 9668.83 over 1
35 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_5_4_5, best cost 18.0243 + 9277.14 = 9295.17 over 1
29 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_6_7_8, best cost 20.2702 + 12922.8 = 12943 over 159
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_7_9_9, best cost 19.7824 + 11851.8 = 11871.6 over 1
62 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_9_0_1, best cost 19.3838 + 12799.5 = 12818.9 over 1
74 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_9_6_6, best cost 23.6323 + 17181.2 = 17204.8 over 2
```

```
24 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_0_1_2, best cost 19.4201 + 11033.2 = 11052.6 over 137
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_0_4_0, best cost 22.2012 + 13294.8 = 13317 over 167 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_2_4_7, best cost 19.7743 + 11706.9 = 11726.7 over 144
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_2_5_6, best cost 18.3118 + 12101.8 = 12120.1 over 144
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_3_4_5, best cost 20.2302 + 13634.4 = 13654.7 over 162
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_6_7_8, best cost 27.6226 + 18199.2 = 18226.9 over 227
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_7_9_7, best cost 22.1824 + 14592 = 14614.2 over 182 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_8_3_1, best cost 25.836 + 17444.7 = 17470.5 over 212
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_9_0_1, best cost 20.153 + 11977.4 = 11997.6 over 150
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_9_3_2, best cost 18.4331 + 12256.9 = 12275.3 over 148
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_0_1_2, best cost 16.7967 + 8459.97 = 8476.77 over 1
14 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_2_9_0, best cost 16.8714 + 8674.01 = 8690.88 over 1
19 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_3_3_8, best cost 17.178 + 6641.15 = 6658.33 over 87
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_3_4_5, best cost 18.0697 + 7310.4 = 7328.47 over 96
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_4_8_1, best cost 17.4521 + 6673.49 = 6690.94 over 8
9 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_6_6_2, best cost 16.614 + 6449.47 = 6466.09 over 77
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_6_7_8, best cost 18.2528 + 6511.31 = 6529.56 over 8
0 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_7_3_6, best cost 17.9047 + 7386.79 = 7404.7 over 90
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_8_0_0, best cost 17.1209 + 7602.14 = 7619.26 over 1
09 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_9_0_1, best cost 17.1181 + 8803.83 = 8820.95 over 1
20 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_0_1_2, best cost 17.5063 + 8010.08 = 8027.59 over 108 fr
```

```
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_3_4_5, best cost 21.043 + 13450.9 = 13471.9 over 179 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_5_0_1, best cost 22.3681 + 13548.3 = 13570.6 over 188 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_6_3_1, best cost 14.0819 + 9688.27 = 9702.35 over 128 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_6_7_8, best cost 21.8318 + 10735.3 = 10757.1 over 138 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_8_1_1, best cost 22.4233 + 12854.6 = 12877 over 196 fram
es.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_8_6_2, best cost 20.1761 + 8724.16 = 8744.33 over 122 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_8_9_7, best cost 24.1689 + 13435.8 = 13460 over 184 fram
es.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_9_0_1, best cost 21.3615 + 10775.9 = 10797.2 over 152 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_9_5_6, best cost 23.1881 + 13136.5 = 13159.7 over 180 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_0_1_2, best cost 21.3143 + 10884.4 = 10905.7 over 153 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_0_2_9, best cost 20.451 + 10550.9 = 10571.3 over 149 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_1_2_0, best cost 22.7572 + 10585.5 = 10608.2 over 142 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_3_4_5, best cost 20.418 + 11828.3 = 11848.7 over 145 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_4_5_2, best cost 21.3547 + 12800.5 = 12821.8 over 168 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_5_4_9, best cost 21.3044 + 11371.3 = 11392.6 over 149 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_6_7_8, best cost 15.8028 + 8000.81 = 8016.62 over 96 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_9_0_1, best cost 19.9944 + 9183.2 = 9203.19 over 123 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_9_3_1, best cost 21.0097 + 10903.9 = 10924.9 over 143 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_9_6_1, best cost 19.9398 + 8269.31 = 8289.25 over 102 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_0_1_2, best cost 16.8281 + 7641 = 7657.83 over 85 fram
es.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_3_4_5, best cost 17.0044 + 6541.83 = 6558.84 over 80 f
```

```
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_4_5_5, best cost 17.3434 + 6280.71 = 6298.05 over 86 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_5_2_5, best cost 16.8475 + 6307.73 = 6324.58 over 83 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_6_3_6, best cost 18.0687 + 9544.36 = 9562.43 over 120
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_6_7_8, best cost 18.6526 + 9783.29 = 9801.94 over 126
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_7_8_7, best cost 19.6023 + 9653.61 = 9673.21 over 120
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_8_8_1, best cost 16.7596 + 7785.86 = 7802.62 over 94 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_9_0_1, best cost 17.3901 + 8428.81 = 8446.2 over 99 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_9_6_6, best cost 18.2585 + 10337.8 = 10356.1 over 135
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_0_0_6, best cost 17.2865 + 8182.71 = 8200 over 109
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_0_1_2, best cost 16.9315 + 8432.17 = 8449.11 over
106 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_0_2_0, best cost 17.6094 + 7529.39 = 7547 over 103
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_2_6_5, best cost 17.1133 + 6973.13 = 6990.24 over
89 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_2_7_9, best cost 17.9708 + 8356 = 8373.97 over 105
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_3_4_5, best cost 17.7051 + 8790.08 = 8807.78 over
108 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_6_7_8, best cost 20.1796 + 9030.99 = 9051.17 over
106 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_8_3_3, best cost 18.9834 + 8992.21 = 9011.2 over 1
08 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_9_0_1, best cost 18.27 + 8957.62 = 8975.89 over 11
5 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_9_4_8, best cost 19.4998 + 8443.89 = 8463.39 over
107 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:124)
Overall cost per frame is 77.2405 = 0.14413 [graph] + 77.0963 [acoustic]
over 10694 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:128)
Done 80 lattices, failed for 0
```

Then we print the results of the test set, to see where the mistakes were made.

In [ ]:
```
%%bash -s "$main_dir"

cd $1

grep 's2\|s1' mono_output.txt
```

```
s1_0_1_2 zero one two
s1_3_4_5 three four five
s1_5_0_1 five zero one
s1_6_3_1 six one
s1_6_7_8 six seven eight
s1_8_1_1 eight one one
s1_8_6_2 eight six two
s1_8_9_7 eight nine seven
s1_9_0_1 nine zero one
s1_9_5_6 nine five six
s2_0_1_2 zero one two
s2_0_2_9 zero two nine
s2_1_2_0 one two zero
s2_3_4_5 three four five
s2_4_5_2 four five two
s2_5_4_9 five four nine
s2_6_7_8 seven eight
s2_9_0_1 nine zero one
s2_9_3_1 nine three one
s2_9_6_1 nine six one
```

Looking at the transcription, there are two utterances where an error has occured: `s1_6_3_1` and `s2_6_7_8`. For these the WER and CER are:

Actual	six	THREE	one
ASR	six	*	one
Error		D	

which gives

$$WER = \frac{0+0+1}{3} \cdot 100\% \approx 33.3\%$$

and

$$CER = \frac{0+0+5}{10} \cdot 100\% = 50\%$$

As for the utterance `6_7_8` made by s_2, the WER and CER are:

Actual	SIX	seven	eight
ASR	***	seven	eight
Error	D		

which gives

$$WER = \frac{0+0+1}{3} \cdot 100\% \approx 33.3\%$$

and

$$CER = \frac{0+0+3}{13} \cdot 100\% \approx 23.1\%$$

Comparing this to the output of the TRI1 trained model, gives us:

In [ ]:
```
%%bash -s "$main_dir"

cd $1

# Create output file for tri1 training
../../src/latbin/lattice-best-path ark:'gunzip -c exp/tri1/decode/lat.1.g
grep 's2\|s1' tri1_output.txt
```

```
s1_0_1_2 zero one two
s1_3_4_5 three four five
s1_5_0_1 five zero one
s1_6_3_1 six one
s1_6_7_8 six seven eight
s1_8_1_1 eight one one
s1_8_6_2 eight six two
s1_8_9_7 eight nine seven
s1_9_0_1 nine zero one
s1_9_5_6 nine five six
s2_0_1_2 zero one two
s2_0_2_9 zero two nine
s2_1_2_0 one two zero
s2_3_4_5 three four five
s2_4_5_2 four five two
s2_5_4_9 five four nine
s2_6_7_8 seven eight
s2_9_0_1 nine zero one
s2_9_3_1 nine three one
s2_9_6_1 nine six one
```

```
../../src/latbin/lattice-best-path 'ark:gunzip -c exp/tri1/decode/lat.1.g
z |' 'ark,t:| utils/int2sym.pl -f 2- exp/tri1/graph/words.txt > tri1_outp
ut.txt'
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_0_1_2, best cost 16.828 + 9574.28 = 9591.11 over 118
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_1_2_9, best cost 17.5493 + 11031.8 = 11049.4 over 14
0 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_2_4_4, best cost 17.2521 + 8413.92 = 8431.17 over 11
8 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_4_5, best cost 19.1261 + 10612.4 = 10631.6 over 14
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_5_8, best cost 18.6341 + 12151.8 = 12170.4 over 15
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_6_4, best cost 19.1542 + 10346.4 = 10365.6 over 14
3 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_3_7_6, best cost 20.4018 + 12764.2 = 12784.6 over 16
4 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_6_7_8, best cost 19.3552 + 13415.2 = 13434.5 over 16
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_9_0_1, best cost 17.5184 + 10995.9 = 11013.4 over 13
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance george_9_5_6, best cost 18.8893 + 11827.8 = 11846.7 over 15
8 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_0_1_2, best cost 17.9866 + 12492.8 = 12510.8 over 1
64 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_0_3_5, best cost 17.749 + 10665.6 = 10683.4 over 15
3 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_2_3_8, best cost 19.0393 + 9678.27 = 9697.3 over 13
1 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_2_8_3, best cost 19.0651 + 9701.79 = 9720.86 over 1
31 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_3_4_5, best cost 17.7184 + 9584.9 = 9602.62 over 13
5 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_5_4_5, best cost 17.9344 + 9180.26 = 9198.2 over 12
9 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_6_7_8, best cost 22.0905 + 12953.2 = 12975.3 over 1
59 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_7_9_9, best cost 19.3221 + 11999.9 = 12019.2 over 1
62 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_9_0_1, best cost 18.6165 + 12816.1 = 12834.7 over 1
74 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance jackson_9_6_6, best cost 23.6554 + 17364.7 = 17388.4 over 2
```

```
24 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_0_1_2, best cost 19.1204 + 11211.7 = 11230.8 over 137
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_0_4_0, best cost 21.786 + 13230 = 13251.8 over 167 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_2_4_7, best cost 19.451 + 11914.1 = 11933.5 over 144
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_2_5_6, best cost 18.2902 + 12080.5 = 12098.8 over 144
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_3_4_5, best cost 20.2484 + 13492.8 = 13513 over 162 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_6_7_8, best cost 27.0631 + 18059.6 = 18086.6 over 227
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_7_9_7, best cost 21.5115 + 14466.6 = 14488.1 over 182
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_8_3_1, best cost 25.6244 + 17305.8 = 17331.5 over 212
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_9_0_1, best cost 19.8685 + 11805.2 = 11825 over 150 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance lucas_9_3_2, best cost 18.4738 + 12014.4 = 12032.9 over 148
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_0_1_2, best cost 16.6352 + 8274.26 = 8290.9 over 11
4 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_2_9_0, best cost 16.8235 + 8667.72 = 8684.55 over 1
19 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_3_3_8, best cost 17.0963 + 6630.84 = 6647.94 over 8
7 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_3_4_5, best cost 18.2014 + 7336.59 = 7354.79 over 9
6 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_4_8_1, best cost 17.337 + 6778.97 = 6796.3 over 89
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_6_6_2, best cost 16.5602 + 6308.74 = 6325.3 over 77
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_6_7_8, best cost 17.9369 + 6591.6 = 6609.54 over 80
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_7_3_6, best cost 17.5819 + 7470.13 = 7487.71 over 9
0 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_8_0_0, best cost 16.194 + 7393.45 = 7409.64 over 10
9 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance nicolas_9_0_1, best cost 17.0262 + 8834.99 = 8852.02 over 1
20 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_0_1_2, best cost 17.0623 + 7873.58 = 7890.65 over 108 fr
```

```
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_3_4_5, best cost 21.1976 + 13460.4 = 13481.6 over 179 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_5_0_1, best cost 21.9979 + 13344.9 = 13366.9 over 188 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_6_3_1, best cost 13.6613 + 9471.73 = 9485.39 over 128 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_6_7_8, best cost 21.6716 + 10724.3 = 10745.9 over 138 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_8_1_1, best cost 21.6103 + 12655.9 = 12677.5 over 196 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_8_6_2, best cost 20.2498 + 8736.32 = 8756.57 over 122 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_8_9_7, best cost 23.7798 + 13596.5 = 13620.3 over 184 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_9_0_1, best cost 20.7395 + 10689.5 = 10710.3 over 152 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s1_9_5_6, best cost 23.2625 + 13092.2 = 13115.5 over 180 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_0_1_2, best cost 20.5403 + 10533.2 = 10553.7 over 153 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_0_2_9, best cost 20.1339 + 10423.2 = 10443.3 over 149 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_1_2_0, best cost 21.866 + 10098.3 = 10120.1 over 142 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_3_4_5, best cost 20.0675 + 11970.1 = 11990.2 over 145 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_4_5_2, best cost 20.8456 + 12951.1 = 12971.9 over 168 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_5_4_9, best cost 20.7095 + 11588.3 = 11609 over 149 fram
es.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_6_7_8, best cost 15.5418 + 8103.83 = 8119.37 over 96 fra
mes.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_9_0_1, best cost 19.2524 + 8831.49 = 8850.74 over 123 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_9_3_1, best cost 20.4625 + 10797.6 = 10818 over 143 fram
es.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance s2_9_6_1, best cost 19.4438 + 8050.64 = 8070.08 over 102 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_0_1_2, best cost 16.6437 + 7659.22 = 7675.86 over 85 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_3_4_5, best cost 16.9055 + 6676.18 = 6693.08 over 80 f
```

```
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_4_5_5, best cost 17.1785 + 6467.02 = 6484.19 over 86 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_5_2_5, best cost 16.7951 + 6445.91 = 6462.7 over 83 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_6_3_6, best cost 18.0339 + 9738.9 = 9756.93 over 120 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_6_7_8, best cost 18.5983 + 9667.31 = 9685.91 over 126
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_7_8_7, best cost 18.9712 + 9362.86 = 9381.83 over 120
frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_8_8_1, best cost 16.123 + 7740.84 = 7756.96 over 94 fr
ames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_9_0_1, best cost 17.1433 + 8463.95 = 8481.09 over 99 f
rames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance theo_9_6_6, best cost 18.27 + 10488 = 10506.2 over 135 fram
es.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_0_0_6, best cost 16.8502 + 8067.75 = 8084.6 over 1
09 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_0_1_2, best cost 16.7233 + 8462.41 = 8479.13 over
106 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_0_2_0, best cost 17.3169 + 7512.48 = 7529.8 over 1
03 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_2_6_5, best cost 16.9486 + 7079.04 = 7095.99 over
89 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_2_7_9, best cost 17.7293 + 8263.96 = 8281.69 over
105 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_3_4_5, best cost 17.4745 + 8891.64 = 8909.12 over
108 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_6_7_8, best cost 19.781 + 9113.91 = 9133.69 over 1
06 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_8_3_3, best cost 18.8304 + 9186.29 = 9205.12 over
108 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_9_0_1, best cost 18.1106 + 8745.21 = 8763.32 over
115 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:99)
For utterance yweweler_9_4_8, best cost 19.1807 + 8613.72 = 8632.9 over 1
07 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:124)
Overall cost per frame is 76.9214 = 0.141988 [graph] + 76.7794 [acoustic]
over 10694 frames.
LOG (lattice-best-path[5.5.1057~1-be222]:main():lattice-best-path.cc:128)
Done 80 lattices, failed for 0
```

And the results were as expected from the scores the same. To further analyze the performance of the ASR, we look at the time aligned transcription.

Step 9.2 Time aligned transcription

Besides looking at the scores of guessing/recognising digits based on the trained acoustic model, it's also interesting to see how good the ASR can pinpoint when each word is uttered. Therefore, the next step is get the alignment for both the words and phones for both the MONO and TRI1 training.

In extracting the best lattice for a transcribed sentence in `src/utils/generate_word_ctm.sh` a parameter `--ls-scale` is given. This scaling factor for Kaldi is often between -0.1 and 0.1 for language model costs. This stems forth from the language model scaling factor or LMSF in short that is usually between 5 and 15. This scale was often applied on the language model probability $P(W)$, such that $P(W)^{LMSF}$ which resulted in a decrease for the value of the LM probability during decoding. Since Kaldi uses weights bigger than 1, the language scale is multiplied by the cost this results in a decrease for the value of the language weighted score.

MONO

We start by retrieving the word- and phone time aligned transcription.

In [ ]:
```
%%bash
./src/utils/generate_word_ctm.sh mono
./src/utils/generate_phone_ctm.sh mono
```

```
../../src/latbin/lattice-1best --lm-scale=0.1 'ark:zcat exp/mono/decode/l
at.1.gz|' ark:-
../../src/latbin/nbest-to-ctm ark:- -
../../src/latbin/lattice-align-words data/lang/phones/word_boundary.int e
xp/mono/final.mdl ark:- ark:-
LOG (lattice-1best[5.5.1057~1-be222]:main():lattice-1best.cc:103) Done co
nverting 80 to best path, 0 had errors.
LOG (lattice-align-words[5.5.1057~1-be222]:main():lattice-align-words.cc:
126) Successfully aligned 80 lattices; 0 had errors.
LOG (nbest-to-ctm[5.5.1057~1-be222]:main():nbest-to-ctm.cc:119) Converted
80 linear lattices to ctm format; 0 had errors.
../../src/latbin/nbest-to-ctm ark:- -
../../src/latbin/lattice-1best ark:- ark:-
../../src/latbin/lattice-align-phones --replace-output-symbols=true exp/m
ono/final.mdl 'ark:zcat exp/mono/decode/lat.1.gz|' ark:-
LOG (lattice-align-phones[5.5.1057~1-be222]:main():lattice-align-phones.c
c:105) Successfully aligned 80 lattices; 0 had errors.
LOG (lattice-1best[5.5.1057~1-be222]:main():lattice-1best.cc:103) Done co
nverting 80 to best path, 0 had errors.
LOG (nbest-to-ctm[5.5.1057~1-be222]:main():nbest-to-ctm.cc:119) Converted
80 linear lattices to ctm format; 0 had errors.
```

TRI1

Now, we do the same for the TRI1 training.

In [ ]:
```
%%bash
./src/utils/generate_word_ctm.sh tri1
./src/utils/generate_phone_ctm.sh tri1
```

```
../../src/latbin/nbest-to-ctm ark:- -
../../src/latbin/lattice-align-words data/lang/phones/word_boundary.int e
xp/tri1/final.mdl ark:- ark:-
../../src/latbin/lattice-1best --lm-scale=0.1 'ark:zcat exp/tri1/decode/l
at.1.gz|' ark:-
LOG (lattice-1best[5.5.1057~1-be222]:main():lattice-1best.cc:103) Done co
nverting 80 to best path, 0 had errors.
LOG (lattice-align-words[5.5.1057~1-be222]:main():lattice-align-words.cc:
126) Successfully aligned 80 lattices; 0 had errors.
LOG (nbest-to-ctm[5.5.1057~1-be222]:main():nbest-to-ctm.cc:119) Converted
80 linear lattices to ctm format; 0 had errors.
../../src/latbin/nbest-to-ctm ark:- -
../../src/latbin/lattice-align-phones --replace-output-symbols=true exp/t
ri1/final.mdl 'ark:zcat exp/tri1/decode/lat.1.gz|' ark:-
../../src/latbin/lattice-1best ark:- ark:-
LOG (lattice-align-phones[5.5.1057~1-be222]:main():lattice-align-phones.c
c:105) Successfully aligned 80 lattices; 0 had errors.
LOG (lattice-1best[5.5.1057~1-be222]:main():lattice-1best.cc:103) Done co
nverting 80 to best path, 0 had errors.
LOG (nbest-to-ctm[5.5.1057~1-be222]:main():nbest-to-ctm.cc:119) Converted
80 linear lattices to ctm format; 0 had errors.
```

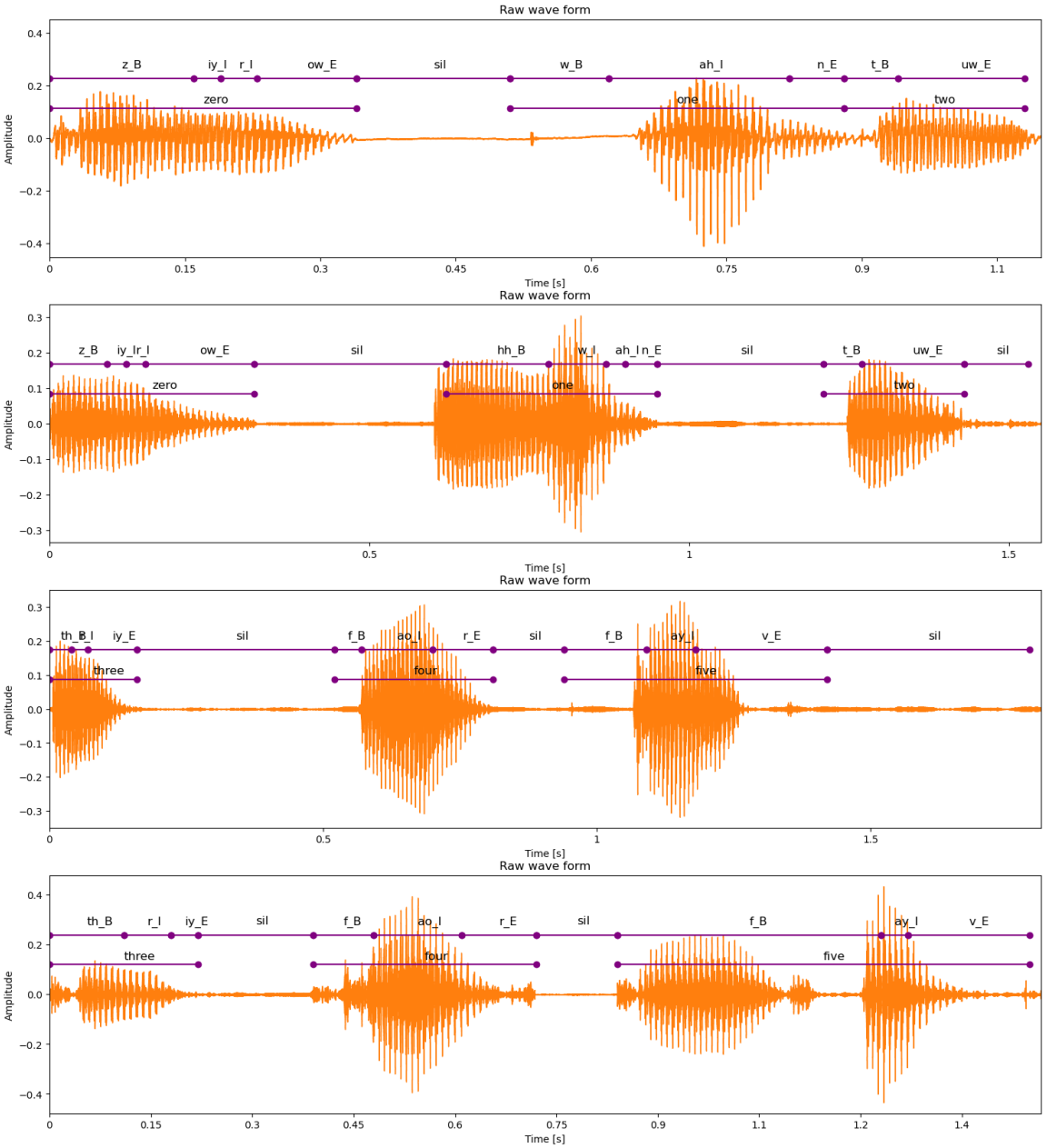
Since looking at these files with the naked eye and comparing everything by hand is needlessly hard, we visualize the time aligned transcriptions.

Step 9.2.1 Visual transcription along time

MONO

Visualize the time aligned transcription for both the phones and words for the files `0_1_2.wav` and `3_4_5.wav` for the speakers s_1 and s_2 using the MONO trained ASR.

In [ ]:
```
plot_fig(
    (18, 20),
    [plot_raw_wave, plot_raw_wave, plot_raw_wave, plot_raw_wave],
    [{"df":df_files, "speaker":"s1", "audio_file":"0_1_2.wav", "model":"m
    {"df":df_files, "speaker":"s2", "audio_file":"0_1_2.wav", "model":"mo
    {"df":df_files, "speaker":"s1", "audio_file":"3_4_5.wav", "model":"mo
    {"df":df_files, "speaker":"s2", "audio_file":"3_4_5.wav", "model":"mo
)
```


Raw wave form
z_B iy_I r_I ow_E sil w_B ah_I n_E t_B uw_E
zero one two
Amplitude
Time [s]
Raw wave form
z_B iy_I r_I ow_E sil hh_B w_I ah_I n_E sil t_B uw_E sil
zero one two
Amplitude
Time [s]
Raw wave form
th_B th_I iy_E sil f_B ao_I r_E sil f_B ay_I v_E sil
three four five
Amplitude
Time [s]
Raw wave form
th_B r_I iy_E sil f_B ao_I r_E sil f_B ay_I v_E
three four five
Amplitude
Time [s]

Something that is rather interesting is the accuracy of the model. Especially knowing that we introduced a lot of speech errors within these files. The speaker s_1 utters `fipe [f ay p]` instead of `five [f ay v]` . Though because of the small lexicon it is highly likely that there are no other known phone states possible for the consonant following `ay` after `f` . Finding the probability of this transition to this observation is what we are interested in. And the other speaker s_2 actually utters the words `two-one [t uw w ah n]` instead of just `two [t uw]` and `four-five [f ao (r) f ay v]` instead of `five [f ay v]` . Because the blended speech errors are so closely spoken and sometimes sound like two words, the expectation would be that the ASR would probably recognize two words instead of just one word. However, looking at the raw wave form where for the number `one` the split between the closely related word and intended word is not as clear and `five` where there is almost near silence between these two words the transcription is for both that of the intended word. Which is rather surprising. Cause looking and listening to the file `3_4_5.wav` of this speaker shows that the words are almost totally seperated from eachother. Since we used the intended words as the correct transcription, the WER and CER scores for all the sentences above are 100\%. However, suppose we used the actual heard speech and transcribed these, then the scores would be:
s_1 - `3_4_5.wav` | | | | :-- | :-- | | Actual | three four FIPE | | ASR | three four FIVE | | Error | S |

Which gives

$$WER = \frac{0+1+0}{3} \cdot 100\% \approx 33.3\%$$

and

$$CER = \frac{0+1+0}{13} \cdot 100\% \approx 7.7\%$$

s_2 - `3_4_5.wav` (Depending on what you hear)

Actual	three	four	FOUR	FIVE
ASR	three	four	FIVE	**
Error			S	D

Which gives

$$WER = \frac{0+1+1}{4} \cdot 100\% = 50\%$$

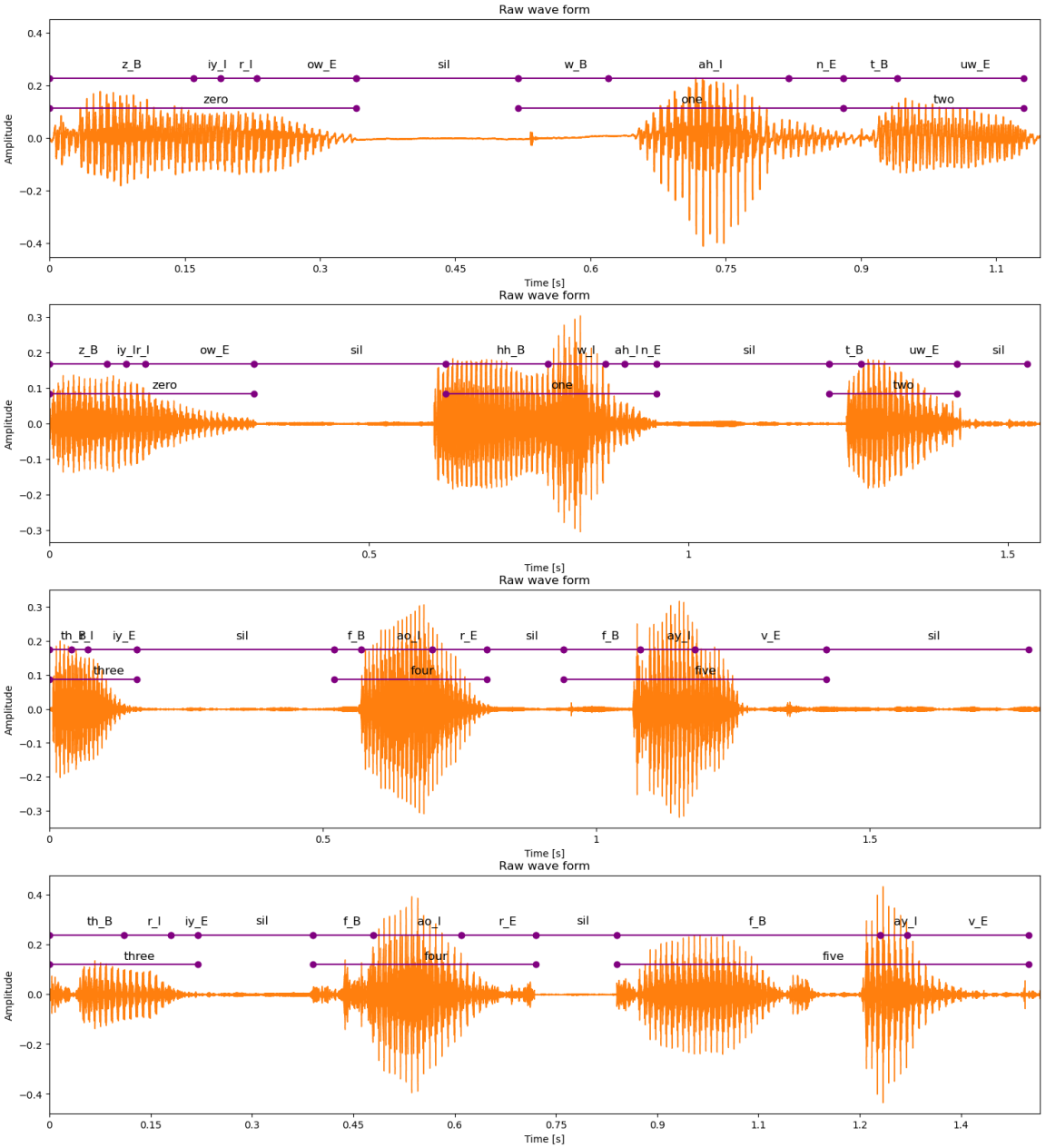
and

$$CER = \frac{0+3+4}{17} \cdot 100\% \approx 41.2\%$$

TRI1

Though the overall scoring of the MONO and TRI1 training are identical, this doesn't necessarily mean that their predicted transcription alignments are also the same. Especially since the TRI1 training further trains on the MONO alignments.

In [ ]:
```
plot_fig(
    (18,20),
    [plot_raw_wave, plot_raw_wave, plot_raw_wave, plot_raw_wave],
    [{"df":df_files, "speaker":"s1", "audio_file":"0_1_2.wav", "model":"t
    {"df":df_files, "speaker":"s2", "audio_file":"0_1_2.wav", "model":"tr
    {"df":df_files, "speaker":"s1", "audio_file":"3_4_5.wav", "model":"tr
    {"df":df_files, "speaker":"s2", "audio_file":"3_4_5.wav", "model":"tr
)
```



> Looking at these figures they are, to the naked eye, identical to those decoded using the MONO training. Only by looking at the files of the time marked transcription for TRI1 and MONO for both the words and phones are we able to see that there are minor differences in time alignment. Sine the MONO alignment was already very accurate on word level, the expectation that there could be even more improvement was limited. However, this does prove that even though the scores for both the MONO and TRI1 training are the same, they do recognize small differences in the time alignment.

Step 10. Recognizing speech errors

To recognise the speech errors introduced by speakers s_1 and s_2 using the ASR models' probability scores we need to retrieve these. However, after doing more research in the framework of Kaldi as well as studying the answers given by one of Kaldi's makers on retrieving the acoustic or language models scores per frame or transcribed word or phone seems not possible or is possible, but would be a rough approximation with a rough work-around approach and not something you should want (1).

Step 10.1 Time aligned transcription confidence scores

Otherwise obtained confidence scores calculated using the lattice-depth, word-categories, unigrams and the number of phones in a word for aligned phone and word lattices are computed.

MONO

In [ ]:
```
%%bash

# Generate the time aligned transcription on word level with confidence s
./src/utils/generate_word_ctm_conf.sh mono
# Generate the time aligned transcription on phone level with confidence
./src/utils/generate_phone_ctm_conf.sh mono
```

```
../../src/latbin/lattice-align-words data/lang/phones/word_boundary.int e
xp/mono/final.mdl ark:- ark:-
../../src/latbin/lattice-push 'ark:zcat exp/mono/decode/lat.1.gz|' ark:-
../../src/latbin/lattice-to-ctm-conf --acoustic-scale=0.8 ark:- -
LOG (lattice-push[5.5.1057~1-be222]:main():lattice-push.cc:98) Pushed 80
lattices, errors on 0
../../src/latbin/lattice-to-nbest ark:- ark:-
LOG (lattice-align-words[5.5.1057~1-be222]:main():lattice-align-words.cc:
126) Successfully aligned 80 lattices; 0 had errors.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_1_2_9-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_2_4_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_5_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_6_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_7_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_3_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_8_3-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_5_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_7_9_9-1, Bayes Risk 0, avg. confidence per-wor
```

```
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_6_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_4_0-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_4_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_7_9_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_8_3_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_3_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_2_9_0-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_4_8_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_6_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_7_3_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_8_0_0-1, Bayes Risk 0, avg. confidence per-wor
```

```
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_5_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_1_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_6_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_9_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_5_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_2_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_1_2_0-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_4_5_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_5_4_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_6_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_4_5_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_5_2_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_3_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_7_8_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_8_8_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance theo_9_6_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_0_6-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_1_2-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_2_0-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_6_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_7_9-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-nbest[5.5.1057~1-be222]:main():lattice-to-nbest.cc:125) D
one applying N-best algorithm to 80 lattices with n = 1, average actual #
paths is 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_3_4_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_6_7_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_8_3_3-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_9_0_1-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_9_4_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
184) Done 80 lattices.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
185) Overall average Bayes Risk per sentence is 0 and per word, 0
../../src/latbin/lattice-to-ctm-conf --acoustic-scale=0.8 ark:- -
../../src/latbin/lattice-align-phones --replace-output-symbols=true exp/m
ono/final.mdl 'ark:zcat exp/mono/decode/lat.1.gz|' ark:-
../../src/latbin/lattice-to-nbest ark:- ark:-
LOG (lattice-align-phones[5.5.1057~1-be222]:main():lattice-align-phones.c
c:105) Successfully aligned 80 lattices; 0 had errors.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_1_2_9-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_2_4_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_5_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_6_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance george_3_7_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_3_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_8_3-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_5_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_7_9_9-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_6_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_4_0-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_4_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_7_9_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance lucas_8_3_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_3_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_2_9_0-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_4_8_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_6_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_7_3_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_8_0_0-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-nbest[5.5.1057~1-be222]:main():lattice-to-nbest.cc:125) D
one applying N-best algorithm to 80 lattices with n = 1, average actual #
paths is 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_5_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_1_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_6_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_9_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_5_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
```

```
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_2_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_1_2_0-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_4_5_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_5_4_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_6_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_4_5_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_5_2_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_3_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_7_8_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_8_8_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_9_6_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_0_6-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_1_2-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_2_0-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_6_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_7_9-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_3_4_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_6_7_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_8_3_3-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance yweweler_9_0_1-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_9_4_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
184) Done 80 lattices.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
185) Overall average Bayes Risk per sentence is 0 and per word, 0
```

TRI1

In [ ]:
```
%%bash

# Generate the time aligned transcription on word level with confidence s
./src/utils/generate_word_ctm_conf.sh tri1
# Generate the time aligned transcription on phone level with confidence
./src/utils/generate_phone_ctm_conf.sh tri1
```

```
../../src/latbin/lattice-push 'ark:zcat exp/tri1/decode/lat.1.gz|' ark:-
../../src/latbin/lattice-to-nbest ark:- ark:-
../../src/latbin/lattice-to-ctm-conf --acoustic-scale=0.8 ark:- -
../../src/latbin/lattice-align-words data/lang/phones/word_boundary.int e
xp/tri1/final.mdl ark:- ark:-
LOG (lattice-push[5.5.1057~1-be222]:main():lattice-push.cc:98) Pushed 80
lattices, errors on 0
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_1_2_9-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_2_4_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_5_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_6_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_7_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_3_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_8_3-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-align-words[5.5.1057~1-be222]:main():lattice-align-words.cc:
126) Successfully aligned 80 lattices; 0 had errors.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_5_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_7_9_9-1, Bayes Risk 0, avg. confidence per-wor
```

```
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_6_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_4_0-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_4_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_7_9_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_8_3_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_3_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_2_9_0-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_4_8_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_6_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_7_3_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_8_0_0-1, Bayes Risk 0, avg. confidence per-wor
```

```
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_5_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_1_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_6_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_9_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_5_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_2_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_1_2_0-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_4_5_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_5_4_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_6_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_4_5_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_5_2_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_3_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_7_8_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_8_8_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance theo_9_6_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_0_6-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_1_2-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_2_0-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_6_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_7_9-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_3_4_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_6_7_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-nbest[5.5.1057~1-be222]:main():lattice-to-nbest.cc:125) D
one applying N-best algorithm to 80 lattices with n = 1, average actual #
paths is 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_8_3_3-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_9_0_1-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_9_4_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
184) Done 80 lattices.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
185) Overall average Bayes Risk per sentence is 0 and per word, 0
../../src/latbin/lattice-align-phones --replace-output-symbols=true exp/t
ri1/final.mdl 'ark:zcat exp/tri1/decode/lat.1.gz|' ark:-
../../src/latbin/lattice-to-ctm-conf --acoustic-scale=0.8 ark:- -
../../src/latbin/lattice-to-nbest ark:- ark:-
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_1_2_9-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_2_4_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_5_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_6_4-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_3_7_6-1, Bayes Risk 0, avg. confidence per-word
1
```

```
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance george_9_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_0_3_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_2_8_3-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_5_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_7_9_9-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance jackson_9_6_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_1_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-align-phones[5.5.1057~1-be222]:main():lattice-align-phones.c
c:105) Successfully aligned 80 lattices; 0 had errors.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_0_4_0-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_4_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_2_5_6-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_3_4_5-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_6_7_8-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_7_9_7-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

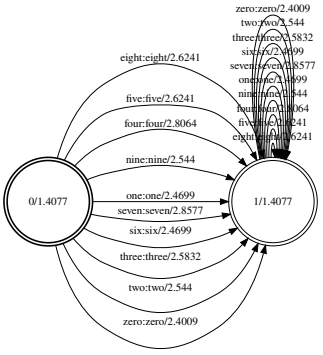
```
175) For utterance lucas_8_3_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_0_1-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance lucas_9_3_2-1, Bayes Risk 0, avg. confidence per-word
1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_0_1_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_2_9_0-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_3_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_3_4_5-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_4_8_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_6_2-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_6_7_8-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_7_3_6-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_8_0_0-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance nicolas_9_0_1-1, Bayes Risk 0, avg. confidence per-wor
d 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_5_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_1_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_6_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_8_9_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s1_9_5_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_0_2_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance s2_1_2_0-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_4_5_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_5_4_9-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_3_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-nbest[5.5.1057~1-be222]:main():lattice-to-nbest.cc:125) D
one applying N-best algorithm to 80 lattices with n = 1, average actual #
paths is 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance s2_9_6_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_0_1_2-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_3_4_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_4_5_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_5_2_5-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_3_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_6_7_8-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_7_8_7-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_8_8_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_9_0_1-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance theo_9_6_6-1, Bayes Risk 0, avg. confidence per-word 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_0_6-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_1_2-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_0_2_0-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_6_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_2_7_9-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_3_4_5-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_6_7_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_8_3_3-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
```

```
175) For utterance yweweler_9_0_1-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
175) For utterance yweweler_9_4_8-1, Bayes Risk 0, avg. confidence per-wo
rd 1
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
184) Done 80 lattices.
LOG (lattice-to-ctm-conf[5.5.1057~1-be222]:main():lattice-to-ctm-conf.cc:
185) Overall average Bayes Risk per sentence is 0 and per word, 0
```

> Looking at these files it quickly becomes apparent that the confidence of each transcription is always 1. Looking at the LM this is not something you would immediately expect since:

In [ ]:
```
%%bash -s "$main_dir"

cd $1

fstdraw --portrait=true --isymbols=exp/tri1/graph/words.txt --osymbols=ex
```



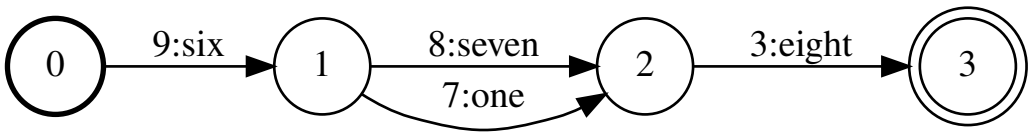
> However as discussed earlier when deciding on the number of n-grams for the LM, we explained why using a simple approach is sufficient enough. The data is very simple in both size and uniqueness. So the connections we see here that seem possible, are not always within the dataset. This causes the lattices to only contain single paths resulting in the answer always being the same. To further show this, we export a couple of the utterance (6-7-8, 9-0-1) for the speakers in the test set to a .svg file.

(Since we now know that the MONO and TRI1 trained models both output the same confidence scores, we only show the lattices for the MONO trained model.)

In [ ]:
```
%%bash -s "$main_dir"

cd $1
./utils/show_lattice.sh --mode save --format svg s1_6_7_8 exp/mono/decode
./utils/show_lattice.sh --mode save --format svg s2_6_7_8 exp/mono/decode
./utils/show_lattice.sh --mode save --format svg s1_9_0_1 exp/mono/decode
./utils/show_lattice.sh --mode save --format svg s2_9_0_1 exp/mono/decode
```

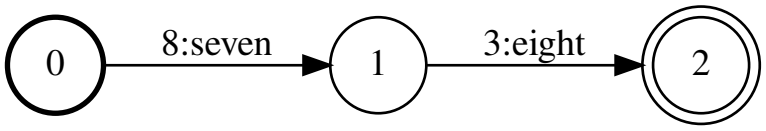
```
Saving to s1_6_7_8.svg
Saving to s2_6_7_8.svg
Saving to s1_9_0_1.svg
Saving to s2_9_0_1.svg
lattice-to-fst --lm-scale=0.0 --acoustic-scale=0.0 ark:- 'scp,p:echo s1_6
_7_8 /tmp/kaldi.AZbU/s1_6_7_8.fst|'
LOG (lattice-to-fst[5.5.1057~1-be222]:main():lattice-to-fst.cc:89) Done c
onverting 80 lattices to word-level FSTs
lattice-to-fst --lm-scale=0.0 --acoustic-scale=0.0 ark:- 'scp,p:echo s2_6
_7_8 /tmp/kaldi.Weh7/s2_6_7_8.fst|'
LOG (lattice-to-fst[5.5.1057~1-be222]:main():lattice-to-fst.cc:89) Done c
onverting 80 lattices to word-level FSTs
lattice-to-fst --lm-scale=0.0 --acoustic-scale=0.0 ark:- 'scp,p:echo s1_9
_0_1 /tmp/kaldi.LWLP/s1_9_0_1.fst|'
LOG (lattice-to-fst[5.5.1057~1-be222]:main():lattice-to-fst.cc:89) Done c
onverting 80 lattices to word-level FSTs
lattice-to-fst --lm-scale=0.0 --acoustic-scale=0.0 ark:- 'scp,p:echo s2_9
_0_1 /tmp/kaldi.altf/s2_9_0_1.fst|'
LOG (lattice-to-fst[5.5.1057~1-be222]:main():lattice-to-fst.cc:89) Done c
onverting 80 lattices to word-level FSTs
```

S_1 - [6 7 8]



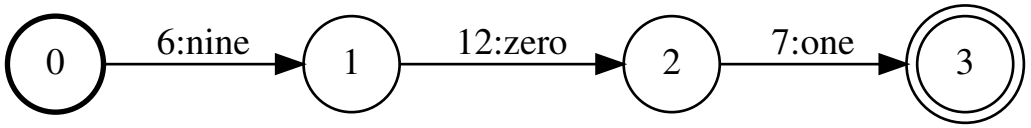
In this picture we can also see the 'mistake' of the earlier shown sentence `6_3_1` where the 3 is deleted. This is caused by the best path found in the lattice of `6` is always followed by either 1 or 7.

S_2 - [6 7 8]

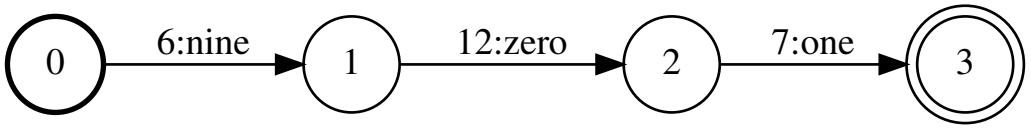


For this person the same situation. However, there is absolutely no other possibility that any other number than 8 follows after a `7`.

S_1 - [9 0 1]



S_2 - [9 0 1]



And for the other images again the lattices show always one optimal path, namely the only existing path. So for a confidence score consisting of among others the lattice depth and n-gram which in our case are both 1 results in a confidence score of 1.00

Conclusion

Because of the small dataset there are no competing paths for transcribing words resulting in a confidence score of 1. Therefore, for further research it is recommended to use a larger dataset. Another option could be to use an existing ASR model and research the confidence scores on an audio file with speech errors to determine whether a model trained fully on healthy speech can properly recognize broken speech. If the transcription results are bad, deciding to further train the model on broken speech could improve the model. Then, the original idea of this study can be implemented to research the possibility of using confidence scores for recognizing speech errors by for example setting a simple threshold.

Step 10.2 Analysing speech errors

Besides trying to determine the differences in speech errors and normal speech using the confidence scores, we can also look at other attributes of the speech audio files.

Such as the difference in vowel formants of speakers for the same words and their spectrograms.

Step 10.2.1 Vowel formants

Vowel formants, also known as formant (resonant) frequencies, are spikes in energy that show the vowel height, advancement and rounding over time. They are retrieved by looking at the highest spikes (somtimes a combination of 2 spikes) visible on the spectrum or by looking at the intensity of vowels on a spectrogram.

> ***Note*** that in the `src.sample.get_formants()` function the bandwith is not considered when confirming a formant. This means that a rather large bandwidth (>400) is also considered to be a formant which can cause actual formants, much more clear in the spectrogram, to not be included or considered as for example F3/F4.

Step 10.2.2 Plots

To study different aspects of the audio files, the:

- raw wave (with predicted word and phone transcription),
- spectrogram (of full file),
- windowed samples (of time range),
- spectrum (with vowel formants of windowed samples),
- spectrogram (of time range with vowel formants)

will be used.

As examples we use the speaker s_1 and the utterance from `6_7_8.wav` .

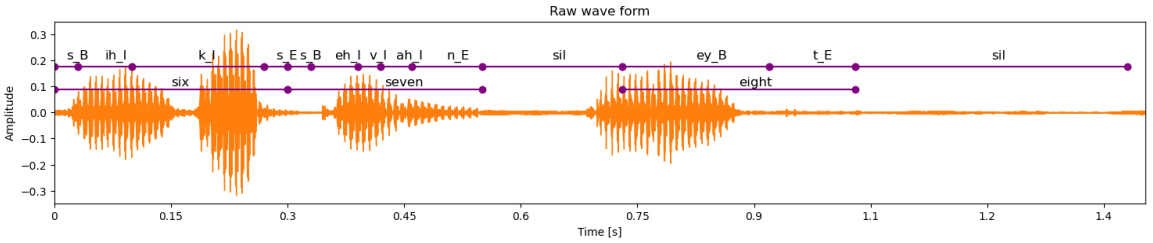
In [ ]:
```
s1_678_samples = get_samples(df_files, "s1", "6_7_8.wav")
```

RAW WAVE

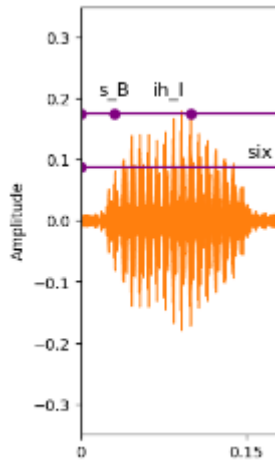
As shown earlier, the raw wave form of an audio signal shows at first sight whether there is sound at certain time intervals or not. We can use this to see if the \<SIL> phone has been correctly placed.

> ***Note*** that this dataset contains almost no noise, and therefore the assumption can be made that the sound detected is always spoken sound. This is of course not applicable to data with noise.

In [ ]:
```
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
plot_raw_wave(
    df=df_files,
    speaker="s1",
    audio_file="6_7_8.wav",
    model="mono",
    ax=ax,
    with_phones=True,
    with_words=True)
```



Looking at the alignment for the word 'eight', we see that the beginning of the alignment is off and a large portion of the near silence part is also part of the alignment. When listening to the audio in Audacity and zooming in on the parts where speech is found, the six sounds very rushed, but the number is only spoken in the first part of the audio wave:

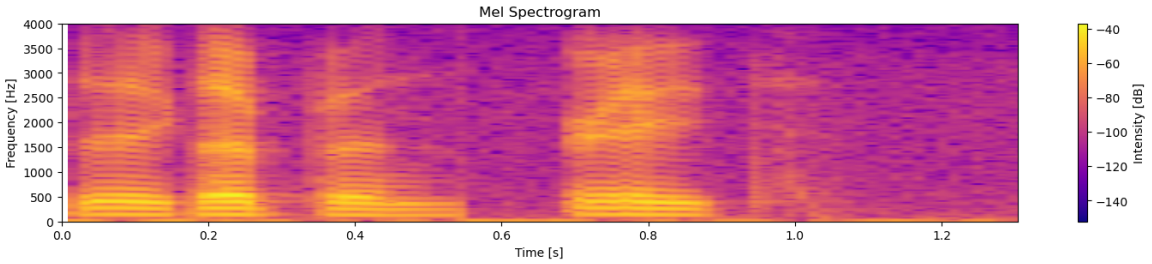


This means that the alignment of the word six and thus seven is also very off.

SPECTROGRAM

A spectrogram as plotted here, is a representation of the Frequency in Hertz over time (s) of an audio signal. The closeness of bands in frequency show the intensity of the sound.

In [ ]:
```
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
plot_spectrogram(
    samples=s1_678_samples,
    ax=ax,
    dt_0=.0,
    dt_end=get_time_samples(s1_678_samples)[-1] - 0.1,
    formants=False
    )
```



The confusion on where six ends and seven begins may have something to do with the relatively small moment of silence between [s ih k s] and [s eh p ah n] and probably also the common 's' sound. The other misalignment for the word 8 could be caused by the seemingly visible intentisity of some sound around t=0.95 but which is not quite high enough in intensity to actually mean something.

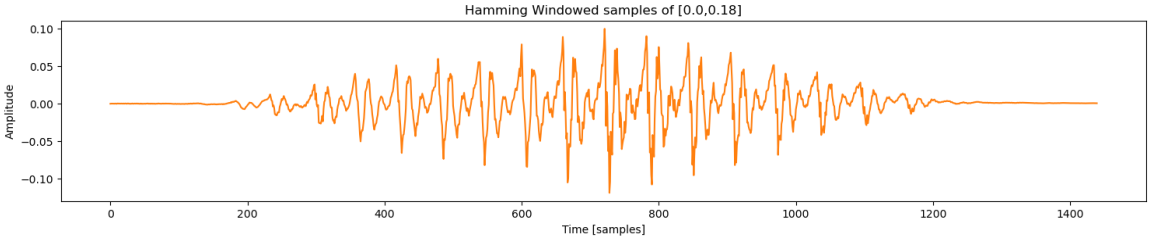
WINDOWED SAMPLE

Windowing is the act of cutting up audio into samples that partly overlap to get a stationary portion of a signal. This is done because the spectrum of audio is non-stationary: constantly changes over time. By extracting spectral features for MFCC vectors for example, this would cause the features to be a bad representation of the original signal.

However, because we work with speech audio we first apply a highpass filter to boost the amount of enery in the lower frequencies. For vowels there is often a lot of energy in the lower frequencies. By boosting these frequencies, the information of the formants for these vowels are more available to the acoustic model which in turn improves phone detecten accuracy.

For the range we look at the range [0.0 - 0.18], to fully focus on the word 6.

In [ ]:
```
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
plot_windowed_samples(
    samples=s1_678_samples,
    ax=ax,
    dt_0=.0,
    dt_end=0.18
)
```

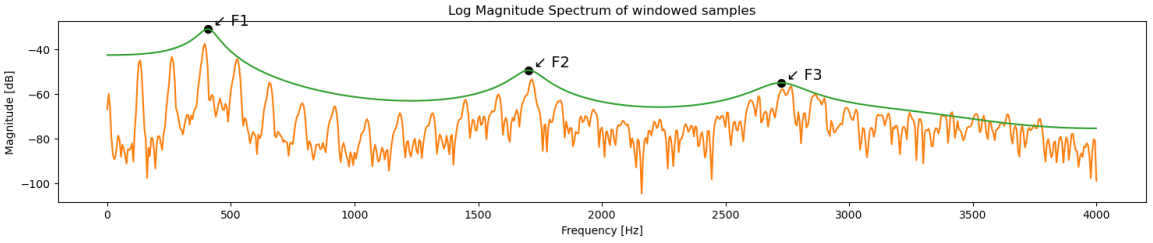


As we can see the signal of the range [.0, .18] has been evenly divided into an almost symmetrical signal, suggesting that the signal has come close to a stationary form. Looking very carefully however, at the first part [200, 600], it seems as though there is a slightly small portion of the signal that is slightly different from the rest of the windowed samples causing two curves to appear by only looking at the highest spikes in the Amplitude. This suggests that there may be some changes in the audio signal, that cause the non-stationary behaviour. Perhaps a change in phone as is to be expected from taking the audio of a six [s ih k s].

SPECTRUM

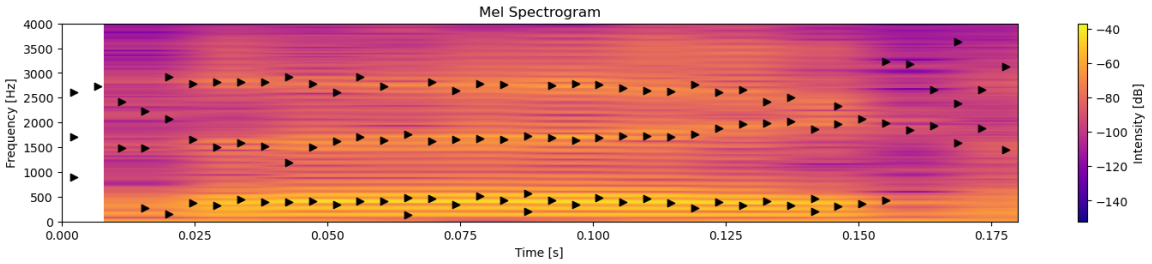
The spectrum is given by the Magnitude in decibels for the frequencies found in the windowed samples of a portion of the original audio signal.

In [ ]:
```
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
plot_magnitude_spectrum(
    samples=s1_678_samples,
    fig=fig,
    ax=ax,
    dt_0=.0,
    dt_end=.18,
    eps=0.028
)
```



The spectrum is taken over the full portion of the audio signal containing the digit 6. This means that the formants shown above are found after sorting the frequency and are a generalization. This spectrum found that the first three formants, respectively lie around 490-500 Hz, 1740-1750 Hz and 2740-2750 Hz. This probably differs from the actual formants since the formants for the word six / s ih k s / can differ per phone. To see whether the differences are there or minimal we zoom in on the spectrogram for the word six and calculate the formants in small steps. Keep in mind, that the range is already relatively small, so it is possible that the formants will be slightly off because of the small range the go over, to make predictions about the formants.

In [ ]:
```
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
plot_spectrogram(
    samples=s1_678_samples,
    ax=ax,
    dt_0=.0,
    dt_end=.18,
    formants=True
)
```



As mentioned earlier, the formants are slightly off at some points, and when there is almost no sound, the formants are also not correct because there is nothing to make formants of. As we can see the formants at the end, around t=0.17, at least for the second and third formant are moving towards eachother. The second formant moves in an increasingly incremental way to just short of the third formant before stabilizing and fading out. This confirms the earlier found inconsistency in the signal.

INTERACTIVE PLOTS

Because comparing the common files of the speakers from the test set to those of the training set will be unreadable if each of these graphs are placed underneath eachother, interactive plots will be used for the common files to easily switch between speakers. Then, underneath the plot a sommation of the findings with screenshots of the most important findings will be showed.

RAW WAVE

Starting by comparing the common files for the speakers of the test set to those of the train set in terms of the raw wave form is the first step. From these wave files the timestamps of certain words and consonants will be used to later on dig deeper into the differences between the normally spoken words and the speech errors.

In [ ]:
```
@wg.interact(file=["0_1_2.wav", "3_4_5.wav", "6_7_8.wav", "9_0_1.wav"],
    speakers=df_files.index.get_level_values(0).unique(),
    model=["mono", "tri1"], phones=True, words=True)
def plot_interactive_raw_wave(file, speakers, model, phones, words):
    fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
    plot_raw_wave(df=df_files, speaker=speakers, audio_file=file,
        model=model, ax=ax, with_phones=phones, with_words=words)
    plt.show()
```

```
interactive(children=(Dropdown(description='file', options=('0_1_2.wav',
'3_4_5.wav', '6_7_8.wav', '9_0_1.wav'…
```

So speaker s_1 makes the speech error on phone level, where the consonant /v/ is traded for a /p/, leading to five / f ay v / becoming fipe / f ay p / and seven / s eh v ah n / becoming / s eh p ah n /. As we have seen earlier however, the transcription does not include the changes in consonants because of the lack of data for alternative paths. Therefore, we use the consonant described as /v/ for the speaker s_1. This gives the following timestamps for the speakers: | | five | / f ay v / | seven | / s eh v ah n / | | :-- | :--: | :--: | :--: | :--: | | s_1 | 0.940-1.420 | 1.180-1.420 | 0.300-0.550 | 0.390-0.420 | | george | 0.900-1.470 | 1.160-1.470 | 0.490-1.230 | 0.670-0.830 | | jackson | 0.920-1.350 | 1.220-1.350 | 0.780-1.270 | 0.880-1.030 | | lucas | 1.120-1.620 | 1.360-1.620 | 0.460-1.150 | 0.840-0.930 | | nicolas | 0.610-0.960 | 0.860-0.960 | 0.200-0.550 | 0.300-0.440 | | theo | 0.500-0.800 | 0.750-0.800 | 0.580-0.930 | 0.690-0.800 | | yweweler | 0.770-1.080 | 1.030-1.080 | 0.280-0.770 | 0.410-0.550 |

Then, for the speaker s_2, mistakes were made that lead to the word two / t uw / becoming one'two / (hh) w ah n /, five / f ay v / becoming fou'five / f ao (r) f ay v / and nine / n ay n / becoming eigh'nine / ay (t) n ay n /. Because the focus does not lie on a particular phone, we take the intended transcripted word for the time ranges. This leads to the following information: | | two | five | nine | | :-- | :--: | :--: | :--: | | s_2 | 1.120-1.430 | 0.840-1.450 | 0.000-0.420 | | george | 0.830-1.180 | 0.900-1.470 | 0.000-0.550 | | jackson | 1.130-1.340 | 0.920-1.350 | 0.000-0.620 | | lucas | 0.990-1.370 | 1.120-1.620 | 0.000-0.490 | | nicolas | 0.800-1.140 | 0.610-0.960 | 0.000-0.420 | | theo | 0.600-0.850 | 0.500-0.800 | 0.000-0.410 | | yweweler | 0.800-1.060 | 0.770-1.080 | 0.000-0.320 |

> ***Note*** that the MONO training was used for these timestamps, because of the small differences in alignments between the TRI1 and MONO training. Moreover, since the alignments are an approximation made by the ASR model there can be differences between the timestamps noted here for the alignment of phones and words and that of the actual alignment. This is further emphasized by the misalignment of the near silence belonging often to the alignment of that of words following or preceeding the near silence.

WINDOWED SIGNAL

As mentioned earlier, looking at the hamming windowed signal of a full word, will most likely result in the signal not becoming a stationary signal. This is caused by the different sounds being created within a single word caused by the different phones. So the primary focus will be the consonants of / v /.

In [ ]:
```
@wg.interact(file=["0_1_2.wav", "3_4_5.wav", "6_7_8.wav", "9_0_1.wav"],
    speakers=df_files.index.get_level_values(0).unique(),
    dt_0=(0.0, 3, 1e-3), dt_end=(0.0, 3, 1e-3))
def plot_interactive_win_signal(file, speakers, dt_0, dt_end):
    fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
    plot_windowed_samples(samples=get_samples(df_files, speakers, file),
    plt.show()
```

```
interactive(children=(Dropdown(description='file', options=('0_1_2.wav',
'3_4_5.wav', '6_7_8.wav', '9_0_1.wav'…
```

From the hamming windowed samples of s_1 for the consonant / v / in five, we can see that there is no symmetry at all. Which means that this signal part is very non-stationary. Comparing this to the other speakers, the speakers 'George', 'Jackson', 'Lucas' are also very non-stationary whereas the speaker 'Nicolas' has a a much more clear symmetric windowed signal and are the signals of 'Theo' and 'Yweweler' also approaching a more stationary signal. Concluding anything on whether it is because of the misalignment, pronounciation or just length in duration used for windowing are all possible causes, but can not be proved because of the lack of data. The dataset consists of multiple dialects each one not appearing more than a handfull of times which makes proving the dialect as a factor an impossible task. The misalignment is also hard to prove, since the lack of linguistic background to correctly define when each phone/word starts and ends is missing. And lack of duration is plausible, however the smaller the windows are taken the less likely a window is to contain any context about surrounding windows. Usually windows of 25 ms are taken with a 10 ms overlaying shift. So it could be that this greatly affects the windowed signal of the very non-stationary signals. But, creating windows using a constant timeframe and timeshift doesn't mean that each window can only contain one phone. It is highly more likely for phones to last multiple frames. So, it may also just be that the pronunciation of a certain phone quickly changes over time causing the non-stationary behaviour. Or more than one phone was taken in with the sampled signal for windowing.

For the hamming windowed samples of the consonant / v / in seven for the test speaker, we can see that what happens when a window is taken from a time frame which in itself would normally create a window. Namely, the oversimplification of an audio signal which in this case makes the windowed signal approach a symmetric form. The speakers from the train set, however, all seem to become rather symmetrical with Theo and Yweweler in the lead.

SPECTRUM

Though the spectrum for non-stationary signals is not very interesting, let alone the signal of a word, it could for the consonant / v / in seven maybe show some insight into the different pitches used to create the consonant.

In [ ]:
```
@wg.interact(file=["0_1_2.wav", "3_4_5.wav", "6_7_8.wav", "9_0_1.wav"],
    speakers=df_files.index.get_level_values(0).unique(),
    dt_0=(0.0, 2, 1e-3), dt_end=(0.0, 3, 1e-3), eps=(.0, 1e3, 1e-2))
def plot_interactive_win_signal(file, speakers, dt_0, dt_end, eps):
    fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
    plot_magnitude_spectrum(samples=get_samples(df_files, speakers, file)
    plt.show()
```

```
interactive(children=(Dropdown(description='file', options=('0_1_2.wav',
'3_4_5.wav', '6_7_8.wav', '9_0_1.wav'…
```

As mentioned before on the time range for the speaker s_1 on the consonant / v / which is very small and the possible consequences of that, we can clearly see that in the magnitude spectrum. The peaks are very wide indicating that the information within the used time frame is on the small side. The First formant lies seemingly on 500Hz, the second around 1500Hz and the third formant around 2600Hz. Retrieving this for the other speakers, gives the following information:

	F1	F2	F3
s_1	500	1500	2600
george	500	1750	3400
jackson	500	1700	3400
lucas	500	1500	2450
nicolas	500	1750	2800
theo	550	1600	2500
yweweler	500	1600	2750

Except for Jackson's third formant the other formants for all speakers all lay rather closely together.

SPECTROGRAM

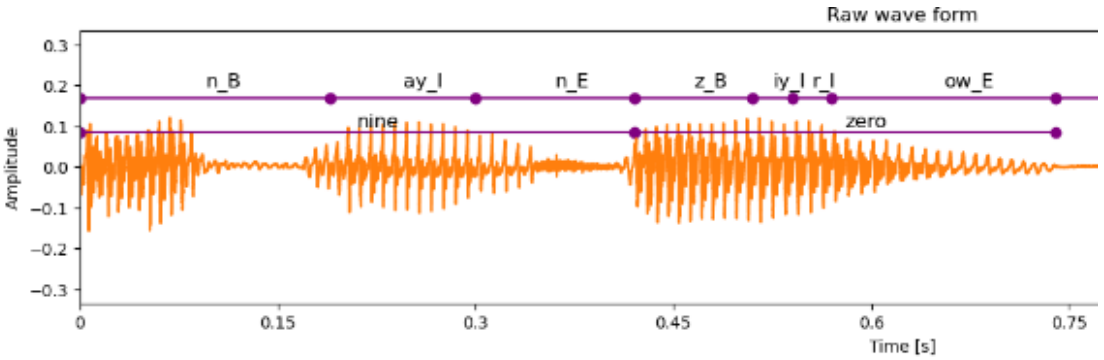
Looking at the spectrograms with the formants drawn in them for both the words and phones can provide insight into the movement of sound for vowels and consonants within words. Though the formants lay near eachother, within this small time frame it does not necessarily mean that this generalization holds for the full signal that is windowed.

In [ ]:
```
@wg.interact(file=["0_1_2.wav", "3_4_5.wav", "6_7_8.wav", "9_0_1.wav"],
    speakers=df_files.index.get_level_values(0).unique(),
    dt_0=(0.0, 2, 1e-3), dt_end=(0.0, 3, 1e-3), formants=True)
def plot_interactive_win_signal(file, speakers, dt_0, dt_end, formants):
    fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(18,3))
    plot_spectrogram(samples=get_samples(df_files, speakers, file), ax=ax
    dt_0=dt_0, dt_end=dt_end, formants=formants
    )
    plt.show()
```

```
interactive(children=(Dropdown(description='file', options=('0_1_2.wav',
'3_4_5.wav', '6_7_8.wav', '9_0_1.wav'…
```

Starting with where we left of for the speaker s_1. If we zoom in on the digit seven for this speaker and then in specific the consonant / v /, we see that the first and second formant are rather constant whereas the third formant seems to slowly increment before fading out. Looking at the spectrogram of george the first thing that shows is the overall high intensity of the sound in contrast to that of s_1 as well as the many narrow bands. For the speaker jackson the bands are wider. The formants however with the exception of the second formant are not very steady. Especially zooming in on the consonant. Looking at the break in the third formant which is included in the alignment for the consonant makes it highly likely that the alignment is incorrect. Formants fading in and out during a phone is not out of the ordinary, but having a formant dissapear for a duration of time only to reappear for the same phone is not really believable. This again shows that the alignments approach the actual alignments, but don't always do a good job of starting or ending at the right moment.

Moving on to speaker s_2 and the digit 9, while keeping in mind that



the approximation made by the ASR model includes both near silence and a portion of the next digit it is easily explainable why we seem to see three words. When looking at the spectrogram it is more likely that the digit 9 ends around 0.34s from looking at the intensity and fading formants. For this speaker the blend speech error looks more like two seperate words, which would make using this as information for recognizing speech errors very challenging if you did not know that that wasn't supposed to be there. Looking at the speaker yweweler, we see that this speaker also has very narrow bands and his alignment also includes a lot of near silence at the end. From the form of the second and third formant creating an open space because of the first and second formant meeting eachother we can see that the sound made here is in the lower vowel space. After the vowel in nine is formed it seemingly opens again to form the sound of the consonant n before fading. Which is a contrast to formants of the speaker jackson where the band are both wider and more seperated and very constant over time. Again, this could be caused by the difference in pronunciation, but the data is too small to draw such a definite conclusion.

Conclusion

The overall conclusion of recognizing speech errors using this approach seems to be that the dataset is too varied in dialects as well as too small to accurately determine or exclude cases to form a vast conclusion on whether a speech error was really found or the differences found are just the cause of differently used mic's or dialects. Therefore, there is no real conclusion on finding speech errors using this type of analysis, but rather

the advice to enlarge the dataset using speakers whose mother language corresponds to the language targeted. And, cutting the audio files using some constant time value results in audio being cut between utterances and on utterances resulting in vague sounds appearing. Rather, using a full sentence based audio is more natural and perhaps would lead to more realistic conclusions in findings as opposed to concatenated files that may not have been connected properly resulting in long silences or peaks in sounds.

Perhaps a better approach to the problem of recognising speech errors is to start by focusing on recognising one type of speech error. And from there, use the results to move on to trying to use the same method for recognising other speech errors.

Another problem is that recognising these errors by eye is very biased in the sense that we know where they occur, so the focus in this research was heavily on the part where the speech error occured. During conversations with the supervisor and listening to actual conversations of aphasic patients it is highly more likely that moments where a speech error occurs can be recognized not by that moment but rather by the sounds made before the speech error or the sounds made after the speech errors.