Assignment 3-Kaldi

1. Installation

I use the linux OS.

1. Run the command

```
git clone https://github.com/kaldi-asr/kaldi
```

2. Check the Dependencies in `kaidi`

```
cd tools
extras/check_dependencies.sh
```

And I need

```
sudo apt-get install sox gfortran libtool subversion
extras/install_mkl.sh
```

3. Compile:

compile tools.

```
cd tools
make -j 8 #I have 8 cpus on my linux machine.
```

It is a long time to wait.

compile src.

We want to use cuda. In file `src/configure`, change default configuration: use_cuda=true

```
cd ../src/
./configure --shared --cudatk-dir=/usr/local/cuda

make depend -j 8

make -j 8
```

4. Test:

   run yes/no example.

```
cd ../egs/yesno/s5
sh run.sh
```

5. Preparation: thchs30

```
wget https://openslr.magicdatatech.com/resources/18/data_thchs30.tgz

tar zxvf data_thchs30.tgz

wget https://openslr.magicdatatech.com/resources/18/resource.tgz

tar zxvf resource.tgz
```

2. Running yesno and THCHS-30 examples

yesno

run yes/no example.

```
cd ../egs/yesno/s5
sh run.sh
```

If we run successfully, the WER (Word Error Rate) result will appear as shown in the following

```
HCLGa is not stochastic
add-self-loops --self-loop-scale=0.1 --reorder=true exp/mono0a/final.mdl exp/mono0a/graph_tgpr/HCLGa.fst
steps/decode.sh --nj 1 --cmd utils/run.pl exp/mono0a/graph_tgpr data/test_yesno exp/mono0a/decode_test_yesno
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd utils/run.pl exp/mono0a/graph_tgpr exp/mono0a/decode_test_yesno
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0a/decode_test_yesno/log/analyze_alignments.log
Overall, lattice depth (10,50,90-percentile)=(1,1,2) and mean=1.2
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0a/decode_test_yesno/log/analyze_lattice_depth_stats.log
local/score.sh --cmd utils/run.pl data/test_yesno exp/mono0a/graph_tgpr exp/mono0a/decode_test_yesno
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 0.00 [ 0 / 232, 0 ins, 0 del, 0 sub ] exp/mono0a/decode_test_yesno/wer_10_0.0
```

wer_10_0.0

syf > kaldi > egs > yesno > s5 > exp > mono0a > decode_test_yesno > wer_10_0.0

```
compute-wer --text --mode=present ark:exp/mono0a/decode_test_yesno/scoring_kaldi/test_filt.txt ark,p:-
%WER 0.00 [ 0 / 232, 0 ins, 0 del, 0 sub ]
%SER 0.00 [ 0 / 29 ]
Scored 29 sentences, 0 not present in hyp.
```

thchs30

In kaldi,

```
cd egs/thchs30/s5
```

Modify cmd.sh under egs/thchs30 dir

```
#export train_cmd=queue.pl
#export decode_cmd="queue.pl --mem 4G"
#export mkgraph_cmd="queue.pl --mem 8G"
#export cuda_cmd="queue.pl --gpu 1“

export train_cmd=run.pl
export decode_cmd="run.pl --mem 4G"
export mkgraph_cmd="run.pl --mem 8G"
export cuda_cmd="run.pl --gpu 1"
```

Modify run.sh under egs/thchs30 dir

```
#n=8      #parallel jobs
n=4      #parallel jobs

#thchs=/nfs/public/materials/data/thchs30-openslr
thchs=/mnt/workspace/syf/thchs30/data_thchs30  #your path to data_thchs30
```

According to

> You can test these four models and comment the rest tests in run.sh.

And we just run the cmd:

```
#monophone here
steps/train_mono.sh --boost-silence 1.25 --nj $n --cmd "$train_cmd"
data/mfcc/train data/lang exp/mono || exit 1;
#test monophone model
local/thchs-30_decode.sh --mono true --nj $n "steps/decode.sh" exp/mono
data/mfcc &

#monophone_ali here
steps/align_si.sh --boost-silence 1.25 --nj $n --cmd "$train_cmd"
data/mfcc/train data/lang exp/mono exp/mono_ali || exit 1;

#triphone
```

```
steps/train_deltas.sh --boost-silence 1.25 --cmd "$train_cmd" 2000 10000
data/mfcc/train data/lang exp/mono_ali exp/tri1 || exit 1;
#test tri1 model
local/thchs-30_decode.sh --nj $n "steps/decode.sh" exp/tri1 data/mfcc &

#triphone_ali
steps/align_si.sh --nj $n --cmd "$train_cmd" data/mfcc/train data/lang exp/tri1
exp/tri1_ali || exit 1;

#lda_mllt
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=3 --
right-context=3" 2500 15000 data/mfcc/train data/lang exp/tri1_ali exp/tri2b ||
exit 1;
#test tri2b model
local/thchs-30_decode.sh --nj $n "steps/decode.sh" exp/tri2b data/mfcc &


#lda_mllt_ali
steps/align_si.sh  --nj $n --cmd "$train_cmd" --use-graphs true data/mfcc/train
data/lang exp/tri2b exp/tri2b_ali || exit 1;

#sat
steps/train_sat.sh --cmd "$train_cmd" 2500 15000 data/mfcc/train data/lang
exp/tri2b_ali exp/tri3b || exit 1;
#test tri3b model
local/thchs-30_decode.sh --nj $n "steps/decode_fmllr.sh" exp/tri3b data/mfcc &

#sat_ali
steps/align_fmllr.sh --nj $n --cmd "$train_cmd" data/mfcc/train data/lang
exp/tri3b exp/tri3b_ali || exit 1;
```

monophone

The result is

```
exp/mono: nj=4 align prob=-100.08 over 25.49h [retry=0.2%, fail=0.0%] states=656
gauss=989
steps/train_mono.sh: Done training monophone system in exp/mono
```

```
3969 warnings in exp/mono/log/align.*.*.log
1051 warnings in exp/mono/log/update.*.log
56 warnings in exp/mono/log/acc.*.*.log
exp/mono: nj=4 align prob=-100.08 over 25.49h [retry=0.2%, fail=0.0%] states=656 gauss=989
steps/train_mono.sh: Done training monophone system in exp/mono
```

The log indicates that a monophone system training, part of a speech recognition experiment using Kaldi software, has been completed. The training utilized four parallel jobs and took 25.49 hours, with a very low retry rate (0.2%) and no failures. It achieved a log likelihood alignment probability of -100.08, and the model was configured with 656 states and 989 Gaussians. The results are stored in the directory 'exp/mono'.

triphone

The result is

```
exp/tri1: nj=4 align prob=-96.76 over 25.48h [retry=0.3%, fail=0.0%] states=1672
gauss=10026 tree-impr=4.80
steps/train_deltas.sh: Done training system with delta+delta-delta features in
exp/tri1
```

```
69 warnings in exp/tri1/log/acc.*.*.log
9 warnings in exp/tri1/log/init_model.log
9 warnings in exp/tri1/log/questions.log
1 warnings in exp/tri1/log/build_tree.log
281 warnings in exp/tri1/log/update.*.log
1 warnings in exp/tri1/log/compile_questions.log
84 warnings in exp/tri1/log/align.*.*.log
exp/tri1: nj=4 align prob=-96.76 over 25.48h [retry=0.3%, fail=0.0%] states=1672 gauss=10026 tree-impr=4.80
steps/train_deltas.sh: Done training system with delta+delta-delta features in exp/tri1
```

The log entry suggests the completion of a more complex triphone-based speech recognition system using delta and delta-delta features, conducted with the Kaldi toolkit. The process ran with four jobs in parallel, taking 25.48 hours, showing a slight improvement in alignment probability to -96.76, with a retry rate of 0.3% and no failures. This system is larger, with 1672 states and 10026 Gaussians, and it shows a tree improvement score of 4.80. The trained model's data is stored under the 'exp/tri1' directory.

lda_mllt

The result is

```
exp/tri2b: nj=4 align prob=-48.20 over 25.48h [retry=0.5%, fail=0.0%]
states=2072 gauss=15030 tree-impr=4.32 lda-sum=24.01 mllt:impr,logdet=1.16,1.67
steps/train_lda_mllt.sh: Done training system with LDA+MLLT features in
exp/tri2b
```

```
137 warnings in exp/tri2b/log/align.*.*.log
8 warnings in exp/tri2b/log/questions.log
272 warnings in exp/tri2b/log/update.*.log
9 warnings in exp/tri2b/log/init_model.log
1 warnings in exp/tri2b/log/compile_questions.log
131 warnings in exp/tri2b/log/acc.*.*.log
4 warnings in exp/tri2b/log/lda_acc.*.log
1 warnings in exp/tri2b/log/build_tree.log
exp/tri2b: nj=4 align prob=-48.20 over 25.48h [retry=0.5%, fail=0.0%] states=2072 gauss=15030 tree-impr=4.32
steps/train_lda_mllt.sh: Done training system with LDA+MLLT features in exp/tri2b
```

The log reflects that the 'exp/tri2b' directory holds the results of training a triphone speech recognition system with Linear Discriminant Analysis (LDA) and Maximum Likelihood Linear Transform (MLLT) optimizations. This training ran with four parallel jobs for 25.48 hours, substantially improving the alignment probability to -48.20. The system shows a modest retry rate of 0.5% with no failures, consisting of 2072 states and 15030 Gaussians, and it records a tree improvement of 4.32. Additional LDA and MLLT improvements are noted with scores of 24.01 and 1.16 for improvement and 1.67 for log determinant, respectively.

sat

The result is

```
exp/tri3b: nj=4 align prob=-47.93 over 25.48h [retry=0.7%, fail=0.0%]
states=2072 gauss=15018 fmllr-impr=2.41 over 18.97h tree-impr=6.36
steps/train_sat.sh: done training SAT system in exp/tri3b
```

```
272 warnings in exp/tri3b/log/update.*.log
8 warnings in exp/tri3b/log/questions.log
16 warnings in exp/tri3b/log/fmllr.*.*.log
210 warnings in exp/tri3b/log/align.*.*.log
9 warnings in exp/tri3b/log/init_model.log
127 warnings in exp/tri3b/log/acc.*.*.log
8 warnings in exp/tri3b/log/est_alimdl.log
1 warnings in exp/tri3b/log/build_tree.log
1 warnings in exp/tri3b/log/compile_questions.log
steps/train_sat.sh: Likelihood evolution:
-49.7806 -49.5859 -49.492 -49.3179 -48.7262 -48.1611 -47.7847 -47.5561 -47.3914 -46.9374 -46.749 -46.5491 -46
exp/tri3b: nj=4 align prob=-47.93 over 25.48h [retry=0.7%, fail=0.0%] states=2072 gauss=15018 fmllr-impr=2.41
steps/train_sat.sh: done training SAT system in exp/tri3b
```

The training session logged under 'exp/tri3b' indicates successful training of an advanced triphone-based speech recognition system, incorporating feature-space Maximum Likelihood Linear Regression (fMLLR). This session also utilized four parallel jobs, ran for 25.48 hours, and achieved a very good alignment probability of -47.93. Despite a slightly higher retry rate of 0.7%, there were no failures. The system maintained 2072 states and had a nearly consistent number of Gaussians at 15018. Notably, the fMLLR improvement was recorded at 2.41 over 18.97 hours, and the tree-based clustering improvement was at 6.36, indicating enhancements in the model's accuracy and robustness.

3. Record a segment of your own speech and recognize its contents using the trained (THCHS-30) models.

Installation

1. Install PortAudio

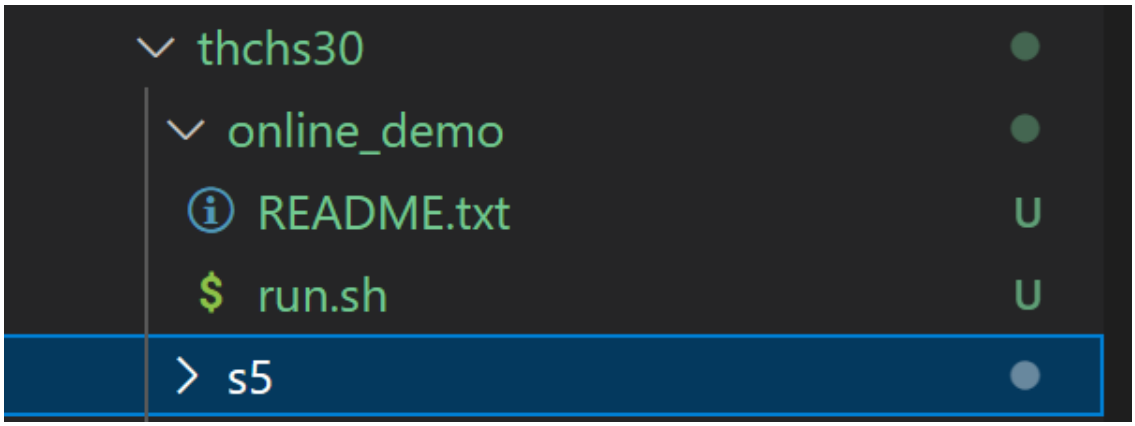
```
cd tools
./install_portaudio.sh
```

```
----------------------------------------------------------------
PortAudio was successfully installed.

On some systems (e.g. Linux) you should run 'ldconfig' now
to make the shared object available.  You may also need to
modify your LD_LIBRARY_PATH environment variable to include
the directory /mnt/workspace/syf/kaldi/tools/portaudio/install/lib
----------------------------------------------------------------

make install-recursive
make[1]: Entering directory '/mnt/workspace/syf/kaldi/tools/portaudio'
if test -n "" ; then for dir in ""; do make -C $dir install; done ; fi
make[1]: Leaving directory '/mnt/workspace/syf/kaldi/tools/portaudio'
(base) /mnt/workspace/syf/kaldi/tools>
```

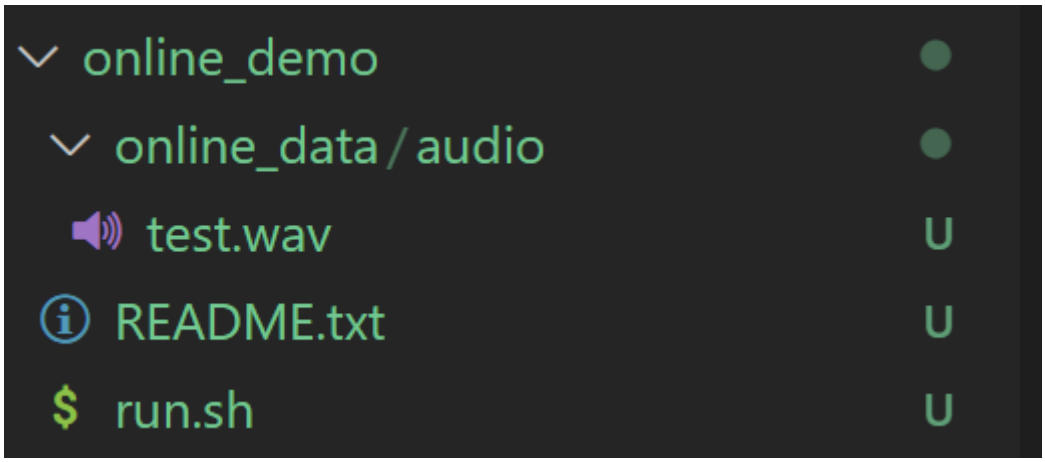
2. Compile Related Tools:

```
cd ../src
make ext
```

3. Copy the dir `kaldi/egs/voxforge/online_demo` to `kaldi/egs/thchs30`



4. Record your audio. Create `online-data` directory. Put your audio, as the following image.



5. Change `kaldi/egs/thchs30/online_demo/run.sh`.

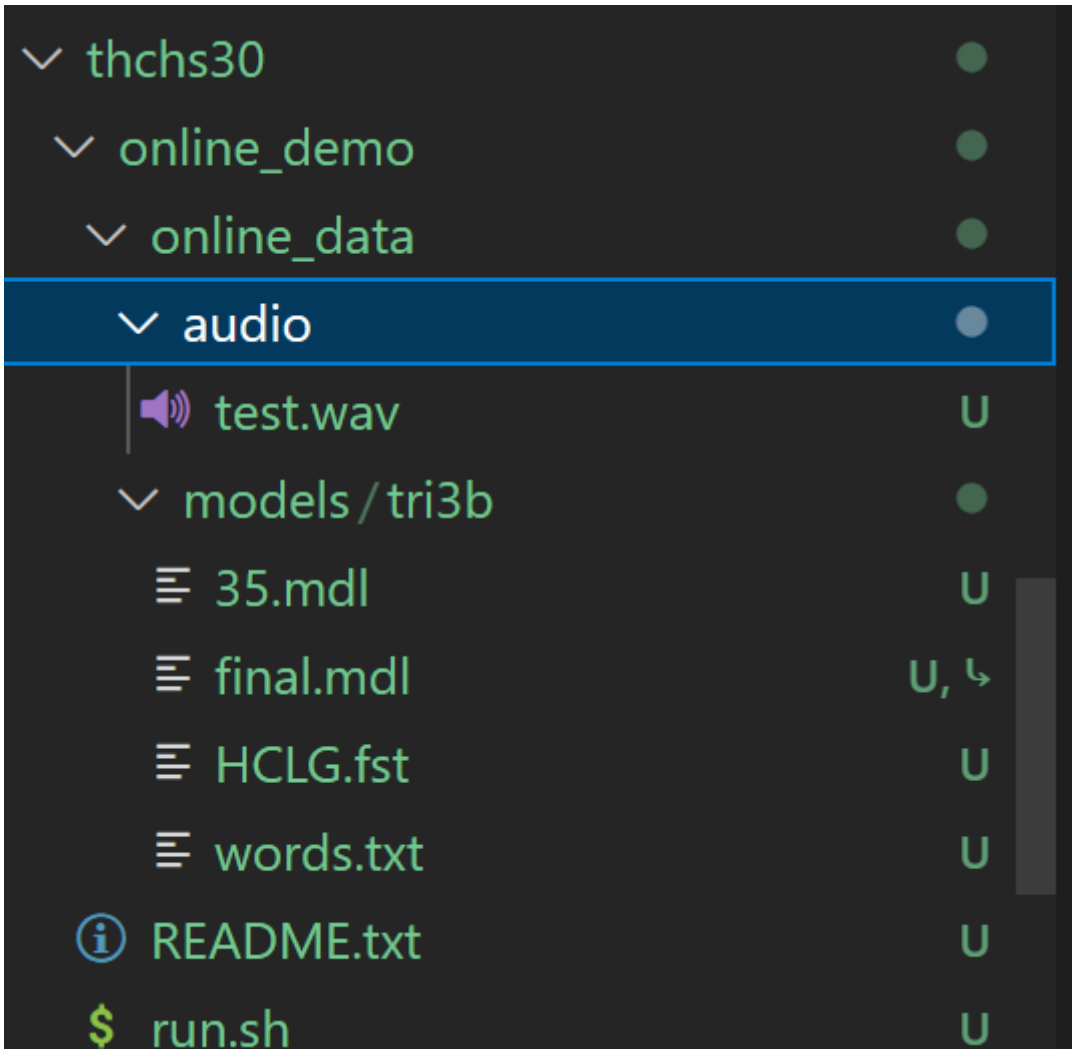
   Comment out the file downloading section .

```
ac_model=${data_file}/models/$ac_model_type
trans_matrix=""
audio=${data_file}/audio

# if [ ! -s ${data_file}.tar.bz2 ]; then
#     echo "Downloading test models and data ..."
#     wget -T 10 -t 3 $data_url;

#     if [ ! -s ${data_file}.tar.bz2 ]; then
#         echo "Download of $data_file has failed!"
#         exit 1
#     fi
# fi
```

Change the acoustic model type ac_model_type to the trained model before. For example, I choose `tri3b`

```
# Change this to "tri2a" if you like to test using a ML-trained model
#ac_model_type=tri2b_mmi
ac_model_type=tri3b
```

6. In dir `online-data`, create dir `models`.
    - Inside the `models` directory, create a `tri3b` directory.
    - Copy the following files from `s5/exp/tri3b` to your `tri3b` directory: `final.mdl` and `35.mdl` (because symbolic link).
    - Copy the following files from `s5/exp/tri3b/graph_word` to your `tri3b` directory: `words.txt` and `HCLG.fst`.



7. Modify the recognition code to use your recorded audio and use your model.

```
87      done
88      # online-wav-gmm-decode-faster --verbose=1 --rt-min=0.8 --rt-max=0.85\
89      #     --max-active=4000 --beam=12.0 --acoustic-scale=0.0769 \
90      #     scp:$decode_dir/input.scp $ac_model/model $ac_model/HCLG.fst \
91      #     $ac_model/words.txt '1:2:3:4:5' ark,t:$decode_dir/trans.txt \
92      #     ark,t:$decode_dir/ali.txt $trans_matrix;;
93      online-wav-gmm-decode-faster --verbose=1 --rt-min=0.8 --rt-max=0.85\
94          --max-active=4000 --beam=12.0 --acoustic-scale=0.0769 \
95          scp:$decode_dir/input.scp  $ac_model/final.mdl $ac_model/HCLG.fst \
96          $ac_model/words.txt '1:2:3:4:5' ark,t:$decode_dir/trans.txt \
97          ark,t:$decode_dir/ali.txt $trans_matrix;;
98
```

and run the cmd `online_demo\run.sh`.(Exactly,this type of `relative address` often elicits errors, so I use the `absolute address` in fact.)

Result

And my input is `语音识别作业 这里是上海市同济大学`。

And my output is

```
online-wav-gmm-decode-faster --verbose=1 --rt-min=0.8 --rt-max=0.85 --max-active=4000 --beam=12.0 --acoustic-scale=0.
0769 --left-context=3 --right-context=3 scp:./work/input.scp /mnt/workspace/syf/kaldi/egs/thchs30/online_demo/online_
data/models/tri1/final.mdl /mnt/workspace/syf/kaldi/egs/thchs30/online_demo/online_data/models/tri1/HCLG.fst /mnt/wor
kspace/syf/kaldi/egs/thchs30/online_demo/online_data/models/tri1/words.txt 1:2:3:4:5 ark,t:./work/trans.txt ark,t:./w
ork/ali.txt
File: new_hw3
平日 语音 识别 作业

实施 上海 市容

况且 大型
```

Reference

1. https://kaldi-asr.org/doc/
2. Kaldi系列--Ubuntu中TIMIT在线识别（三） error opening input stream online-data/models/tri1-CSDN博客