Aishell experiment on Kaldi
Lab 6

[Objective]

1. Run aishell experiment on Kaldi and understand the GMM-HMM speech recognition.

[Task description]

Aishell is an open Chinese Mandarin speech database designed for Chinese ASR experiment.

For the details of the dataset, please read kaldi/egs/aishell/README.txt

[Running aishell script]

1) Script modification

The aishell experiment is located at kaldi/egs/aishell/s5 and run.sh is the entry point of the experiment.

First of all, you have to modify the file **cmd.sh** before running the experiment.

Change the 3 lines of cmd.sh from

```
export train_cmd="queue.pl --mem 2G"
export decode_cmd="queue.pl --mem 4G"
export mkgraph_cmd="queue.pl --mem 8G"
```

to

```
export train_cmd="run.pl"
export decode_cmd=run.pl
export mkgraph_cmd="run.pl"
```

This change is needed as our server is a standalone machine and it is not in a grid system.

2) Install language model tool

In this experiment, we need to use the kaldi language model tool, install it by running kaldi/tools/extras/install_kaldi_lm.sh.

And then source kaldi/tools/env.sh before you run the aishell experiment.

3) Script understanding

Open run.sh and understand the flow in the script.

It first downloads the data and unzip it with

```
local/download_and_untar.sh $data $data_url data_aishell || exit 1;
local/download_and_untar.sh $data $data_url resource_aishell || exit 1;
```

Then it prepare the necessary kaldi files with

```
# Lexicon Preparation,
local/aishell_prepare_dict.sh $data/resource_aishell || exit 1;

# Data Preparation,
local/aishell_data_prep.sh $data/data_aishell/wav $data/data_aishell/transcript || exit 1;

# Phone Sets, questions, L compilation
utils/prepare_lang.sh --position-dependent-phones false data/local/dict \
"<SPOKEN_NOISE>" data/local/lang data/lang || exit 1;
```

Then it does the LM training by

```
# LM training
local/aishell_train_lms.sh || exit 1;

# G compilation, check LG composition
utils/format_lm.sh data/lang data/local/lm/3gram-mincount/lm_unpruned.gz \
data/local/dict/lexicon.txt data/lang_test || exit 1;
```

Then it does the MFCC extraction by

```
mfccdir=mfcc
for x in train dev test; do
    steps/make_mfcc_pitch.sh --cmd "$train_cmd" --nj 10 data/$x
exp/make_mfcc/$x $mfccdir || exit 1;
    steps/compute_cmvn_stats.sh data/$x exp/make_mfcc/$x $mfccdir
|| exit 1;
    utils/fix_data_dir.sh data/$x || exit 1;
done
```

The rest are the training, starting from monophone training to triphone training.

4) Running the script

To avoid the network and storage burden, the data is already downloaded and every one can share it. In the run.sh, please modify the location of data by

data=/home/cseadmin/source/aishell1_small

and remove/comment the following two lines:

local/download_and_untar.sh $data $data_url data_aishell || exit 1;

local/download_and_untar.sh $data $data_url resource_aishell || exit 1;

Run the rest of the script till the training and decoding of "tri2". The decoding of the test set with the tri2 model is like

decode tri2

utils/mkgraph.sh data/lang_test exp/tri2 exp/tri2/graph

steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj 10 \

exp/tri2/graph data/test exp/tri2/decode_test

5) Check the result CER (character error rate)

Use the last two lines in run.sh to check the result.

for x in exp/*/decode_test; do [-d $x] && grep WER $x/cer_* | utils/best_wer.sh; done 2>/dev/null

The output text from the decoding can be viewed at

exp/tri2/decode_test/scoring_kaldi/penalty_*/*.txt

[Testing with your voice]

Now, please test the system with your own voice.

Speak several Chinese sentences and record them with your device.

Upload them to the server and decode them with your aishell model.

Remember to convert them to .wav format with the correct sampling rate.

You need to create a data folder of your own voice similar to data/train and data/test.

There are 4 basic files need to be prepared: wav.scp, utt2spk, spk2utt and text.

The file "text" contains the transcriptions of each utterance.
00308777_001 今天天气很好
00308777_002 我要去学校
00308777_003 考试结果出来了

The first element on each line is the utterance-id, which is an arbitrary text string, but if you have speaker information in your setup, you should make the speaker-id a prefix of the utterance id.
In this lab, you should make your student id to be the speaker id.
For example, if your student id is 00308777, the utterance id of your first sentences should be
00308777_001

The file "utt2spk" maps the utterances to their speaker. The format is
`<utterance-id> <speaker-id>`

In your utt2spk file, it should look like
00308777_001 00308777
00308777_002 00308777
00308777_003 00308777

The utt2spk and spk2utt files contain exactly the same information.
After creating your utt2spk file, the spk2utt file can be generated by

The file "wav.scp" maps the utterances to the location of the audio files.
It should look like
00308777_001 /home/tom/recording/001.wav
00308777_002 /home/tom/recording/002.wav
00308777_003 /home/tom/recording/003.wav

Change the path of the files accordingly.

Replace the “test” folder with your created folder in run.sh and do the decoding with the trained model.