Kaldi构建一个简单的英文数字串识别系统

FEBRUARY 22, 2017

本文主要参考的是 kaldi-asr.org，主要讲述的是用自己的录音来构建一个数字串识别系统。

本文将主要分为以下几个部分：

- 录制语音
- 数据准备
  - 声学数据
    - 声学训练数据准备
      - spk2gender
      - wav.scp
      - text
      - utt2spk
    - 声学测试数据准备
    - 语料库
  - 语言数据
    - lexicon.txt
    - nonsilence_phones.txt
    - silence_phones.txt
    - optional_silence.txt
- 环境准备
- 编写运行脚本
- 运行脚本

录制语音

这里是英文数字串识别，因此需要一些用英语朗读数字的语音。我录制了 128 个语音文件，分别是两个人朗读，其中每个文件只包含三个数字。这 128 文件

中 80 个用于训练， 48 个用于测试。并且训练数据和测试数据都被分成了 8 部分（可以假装成 8 个人），每部分分别 10 个 和 6 个。读者可以到我的 GitHub 上下载这些语音数据。训练集和测试集的前五个目录是我(男)朗读，后面三个是女士朗读的。

我的录音是 16 kHz 采样，16 位量化的。如果是 8 kHz 采样，需要在提特征时指明采样频率。即将 `steps/make_mfcc.sh` 里面 `compute-mfcc-feats` 命令加上 `--sample-frequency=8000` 的选项。

目录结构如下：

```
├── test
│   ├── 1
│   ├── 2
│   ├── 3
│   ├── 4
│   ├── 5
│   ├── 6
│   ├── 7
│   └── 8
└── train
    ├── 1
    ├── 2
    ├── 3
    ├── 4
    ├── 5
    ├── 6
    ├── 7
    └── 8
```

当然您也可以选择自己录音，也欢迎你把你的录音共享。

从数据准备到编写脚本一直都是在为最后做准备，如果都完成了的话，你的目录应该长这样：

```
.
├── cmd.sh
```

```
├── conf
│   ├── decode.config
│   └── mfcc.conf
├── data
│   ├── local
│   │   ├── corpus.txt
│   │   └── dict
│   │       ├── lexicon.txt
│   │       ├── nonsilence_phones.txt
│   │       ├── optional_silence.txt
│   │       └── silence_phones.txt
│   ├── test
│   │   ├── spk2gender
│   │   ├── text
│   │   ├── utt2spk
│   │   └── wav.scp
│   └── train
│       ├── spk2gender
│       ├── text
│       ├── utt2spk
│       └── wav.scp
├── local
│   ├── make_mfcc.sh
│   └── score.sh
├── path.sh
├── run.sh
├── steps -> ../../timit/s5/steps
└── utils -> ../../timit/s5/utils
```

数据准备

数据的准备包括声学数据和语言数据。首先建立一个目录 `data/`。

声学数据

声学数据主要包括训练集和测试集的数据，以及一个语料库，在 `data/` 目录下新建两个目录：`train/` 、`test/` 和 `local`。

声学训练数据准备

这些数据主要是四个文件：spk2gender、wav.scp、text 和 utt2spk，都保存在 data/train/ 目录下。

spk2gender

这个文件主要描述说话人编号和性别的对应关系，具有这样的形式：<speakerID> <gender>，比如我这里训练集是“8”个人，前 5 个男(Male)，后 3 个女(Female)。比如

```
1 m
2 m
3 m
4 m
5 m
6 f
7 f
8 f
```

wav.scp

这个文件主要是保存了语音编号和语音文件的对应关系，具有这样的形式：<uterranceID> <full_path_to_audio_file>。比如

```
1_040 /mnt/b/huanglu/speech/numbers-en/train/1/1-040.wav
1_089 /mnt/b/huanglu/speech/numbers-en/train/1/1-089.wav
1_104 /mnt/b/huanglu/speech/numbers-en/train/1/1-104.wav
1_231 /mnt/b/huanglu/speech/numbers-en/train/1/1-231.wav
1_327 /mnt/b/huanglu/speech/numbers-en/train/1/1-327.wav
1_413 /mnt/b/huanglu/speech/numbers-en/train/1/1-413.wav
1_462 /mnt/b/huanglu/speech/numbers-en/train/1/1-462.wav
1_468 /mnt/b/huanglu/speech/numbers-en/train/1/1-468.wav
1_470 /mnt/b/huanglu/speech/numbers-en/train/1/1-470.wav
1_560 /mnt/b/huanglu/speech/numbers-en/train/1/1-560.wav
..............
```

text

这个文件主要是标注，描述了语音编号和标注之间的对应关系，具有这样的形式：<utterranceID> <text_transcription>。比如

```
1_040 zero four zero
1_089 zero eight nine
1_104 one zero four
1_231 two three one
1_327 three two seven
1_413 four one three
1_462 four six two
1_468 four six eight
1_470 four seven zero
1_560 five six zero
```

utt2spk

这个文件是联系说话人编号和语音编号，具有这样的形式：<uterranceID> <speakerID>。比如

```
1_040 1
1_089 1
1_104 1
1_231 1
1_327 1
1_413 1
1_462 1
1_468 1
1_470 1
1_560 1
```

这里我们建议把说话人的编号放在语音编号的前缀。至此，训练集的声学数据准备好了。

声学测试数据准备

和训练集数据一样，这些数据也主要是四个文件：spk2gender、wav.scp、text 和 utt2spk，都保存在 data/test/ 目录下。这些目录的形式和之前一样，只是 wav.scp 需要映射到测试集的数据，重新修改语音编号。

语料库

我们还需要一个语料库，其包含了所有训练集数据的标注，请命名为 corpus.txt，并保存在 data/local/ 目录下。比如

```
zero four zero
zero eight nine
one zero four
two three one
three two seven
four one three
four six two
four six eight
four seven zero
five six zero
```

语言数据

语言数据主要包括 lexicon.txt 、optional_silence.txt、nonsilence_phones.txt、silence_phones.txt，并在 data/local/ 目录下新建文件夹 dict，将这四个文件保存到那里。

lexicon.txt

这个文件应该包括你标注里所有出现的词的发音，即音素表达，由于这里只有十个单词，再加上静音，因此不管是词还是音素，数量都比较少。

```
!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
```

nonsilence_phones.txt

这个文件列出了上面出现的所有的非静音音素。

```
ah
ao
ay
eh
ey
f
hh
ih
iy
k
n
ow
r
s
t
th
uw
w
v
z
```

silence_phones.txt

这里面包含了静音音素。

```
sil
spn
```

optional_silence.txt

只有可选的静音音素。

```
sil
```

环境准备

回到项目根目录。首先我们需要定义文件 cmd.sh 和 path.sh，其中前者主要包括运行的形式，而后者主要包括 kaldi 依赖的路径。

cmd.sh：

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

path.sh：

```
export train_cmd="run.pl"
export decode_cmd="run.pl"
export cuda_cmd="run.pl"
export mkgraph_cmd="run.pl"
Huang-Lus-MacBook-Air:en huanglu$ cat path.sh
export KALDI_ROOT=`pwd`/../../..
[ -f $KALDI_ROOT/tools/env.sh ] && . $KALDI_ROOT/tools/env.sh
export PATH=$PWD/utils/:$KALDI_ROOT/tools/openfst/bin:$KALDI_ROOT/tools/irstlm/bin/
[ ! -f $KALDI_ROOT/tools/config/common_path.sh ] && echo >&2 "The standard file $KA
. $KALDI_ROOT/tools/config/common_path.sh
export LC_ALL=C
Huang-Lus-MacBoo
```

此外我们还需要一些 kaldi 已经写好的脚本，比如 steps/decode.sh 和 utils/mkgraph.sh，简单起见，我们直接这两个文件夹链接过来。

```
$ ln -s ../../timit/s5/steps steps
$ ln -s ../../timit/s5/utils utils
```

然后就是为了打分以及处理提特征时采样率非 16 kHz 时，需要在根目录下建立 local 文件夹，然后从别的那里拷贝来 make_mfcc.sh 和 score.sh。

最后就是建立目录 conf，存放一些配置文件。包括：

decode.config：

```
first_beam=10.0
beam=13.0
lattice_beam=6.0
```

mfcc.conf：

```
--use-energy=false
```

编写运行脚本

在根目录下建立一个 run.sh 脚本，输入以下内容（有注释，我不解释了）：

```
#!/bin/bash

. ./path.sh || exit 1
. ./cmd.sh || exit 1

nj=4
lm_order=1

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


# Removing previously created data (from last run.sh execution)
rm -rf exp mfcc data/train/spk2utt data/train/cmvn.scp data/train/feats.scp \
      data/train/split1 data/test/spk2utt data/test/cmvn.scp data/test/feats.scp \
      data/test/split1 data/local/lang data/lang data/local/tmp \
      data/local/dict/lexiconp.txt
```

```
echo
echo "===== PREPARING ACOUSTIC DATA ====="
echo

# Making spk2utt files
utils/utt2spk_to_spk2utt.pl data/train/utt2spk > data/train/spk2utt
utils/utt2spk_to_spk2utt.pl data/test/utt2spk > data/test/spk2utt

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

# Making feats.scp files
mfccdir=mfcc
# Uncomment and modify arguments in scripts below if you have any problems with dat
# utils/validate_data_dir.sh data/train     # script for checking prepared data - h
# utils/validate_data_dir.sh data/test
# utils/fix_data_dir.sh data/train          # tool for data proper sorting if neede
# utils/fix_data_dir.sh data/test

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

# Making cmvn.scp files
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test $mfccdir

echo
echo "===== PREPARING LANGUAGE DATA ====="
echo
# Preparing language data
utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang data/lang

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_
        exit 1
    fi
fi

local=data/local
mkdir $local/tmp
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt \
            -wbdiscount -text $local/corpus.txt -lm $local/tmp/lm.arpa

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

lang=data/lang
arpa2fst --disambig-symbol=#0 --read-symbol-table=$lang/words.txt \
                         $local/tmp/lm.arpa $lang/G.fst

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

steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train data/lang exp/mono  || e

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_cmd" \
                exp/mono/graph data/test exp/mono/decode
local/score.sh data/test data/lang exp/mono/decode/

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

steps/align_si.sh --nj $nj --cmd "$train_cmd" data/train data/lang exp/mono exp/mon

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

steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train data/lang exp/mono_a

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_cmd" \
                exp/tri1/graph data/test exp/tri1/decode
local/score.sh data/test data/lang exp/tri1/decode/

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

运行脚本

最后在运行 `run.sh` 之前先进行下面的命令：

```
$ chmod u+x local/*.sh *.sh
```

运行结果（省去了部分重复的）：

```
$ ./run.sh

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

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

steps/make_mfcc.sh --nj 4 --cmd run.pl data/train exp/make_mfcc/train mfcc
utils/validate_data_dir.sh: Successfully validated data-directory data/train
steps/make_mfcc.sh: [info]: no segments file exists: assuming wav.scp indexed by ut
Succeeded creating MFCC features for train
steps/make_mfcc.sh --nj 4 --cmd run.pl data/test exp/make_mfcc/test mfcc
utils/validate_data_dir.sh: Successfully validated data-directory data/test
steps/make_mfcc.sh: [info]: no segments file exists: assuming wav.scp indexed by ut
Succeeded creating MFCC features for test
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train mfcc
Succeeded creating CMVN stats for train
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test mfcc
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
--> data/local/dict/silence_phones.txt is OK

Checking data/local/dict/optional_silence.txt ...
--> reading data/local/dict/optional_silence.txt
--> data/local/dict/optional_silence.txt is OK

Checking data/local/dict/nonsilence_phones.txt ...
--> reading data/local/dict/nonsilence_phones.txt
--> 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
--> 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
fstaddselfloops data/lang/phones/wdisambig_phones.int data/lang/phones/wdisambig_w
prepare_lang.sh: validating output directory
utils/validate_lang.pl data/lang
Checking data/lang/phones.txt ...
--> data/lang/phones.txt is OK

Checking words.txt: #0 ...
--> 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 ...
--> summation property is OK

Checking data/lang/phones/context_indep.{txt, int, csl} ...
--> 10 entry/entries in data/lang/phones/context_indep.txt
--> data/lang/phones/context_indep.int corresponds to data/lang/phones/context_inde
--> data/lang/phones/context_indep.csl corresponds to data/lang/phones/context_inde
--> data/lang/phones/context_indep.{txt, int, csl} are OK

Checking data/lang/phones/nonsilence.{txt, int, csl} ...
--> 80 entry/entries in data/lang/phones/nonsilence.txt
--> data/lang/phones/nonsilence.int corresponds to data/lang/phones/nonsilence.txt
--> data/lang/phones/nonsilence.csl corresponds to data/lang/phones/nonsilence.txt
--> data/lang/phones/nonsilence.{txt, int, csl} are OK
```

```
Checking data/lang/phones/silence.{txt, int, csl} ...
--> 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} ...
--> 1 entry/entries in data/lang/phones/optional_silence.txt
--> data/lang/phones/optional_silence.int corresponds to data/lang/phones/optional_
--> data/lang/phones/optional_silence.csl corresponds to data/lang/phones/optional_
--> data/lang/phones/optional_silence.{txt, int, csl} are OK

Checking data/lang/phones/disambig.{txt, int, csl} ...
--> 2 entry/entries in data/lang/phones/disambig.txt
--> data/lang/phones/disambig.int corresponds to data/lang/phones/disambig.txt
--> data/lang/phones/disambig.csl corresponds to data/lang/phones/disambig.txt
--> data/lang/phones/disambig.{txt, int, csl} are OK

Checking data/lang/phones/roots.{txt, int} ...
--> 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} ...
--> 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} ...
--> 9 entry/entries in data/lang/phones/extra_questions.txt
--> data/lang/phones/extra_questions.int corresponds to data/lang/phones/extra_ques
--> data/lang/phones/extra_questions.{txt, int} are OK

Checking data/lang/phones/word_boundary.{txt, int} ...
--> 90 entry/entries in data/lang/phones/word_boundary.txt
--> data/lang/phones/word_boundary.int corresponds to data/lang/phones/word_boundar
--> 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 symbols
--> data/lang/phones/word_boundary.txt is the union of nonsilence.txt and silence.t
--> 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 67 word sequence
--> resulting phone sequence from L.fst corresponds to the word sequence
--> L.fst is OK
--> generating a 16 word sequence
--> resulting phone sequence from L_disambig.fst corresponds to the word sequence
--> L_disambig.fst is OK

Checking data/lang/oov.{txt, int} ...
--> 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 =====

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

```
arpa2fst --disambig-symbol=#0 --read-symbol-table=data/lang/words.txt data/local/tr
LOG (arpa2fst:Read():arpa-file-parser.cc:96) Reading \data\ section.
LOG (arpa2fst:Read():arpa-file-parser.cc:151) Reading \1-grams: section.
LOG (arpa2fst:RemoveRedundantStates():arpa-lm-compiler.cc:355) Reduced num-states -

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

steps/train_mono.sh --nj 4 --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: 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
steps/diagnostic/analyze_alignments.sh: see stats in exp/mono/log/analyze_alignment
311 warnings in exp/mono/log/update.*.log
84 warnings in exp/mono/log/align.*.*.log
exp/mono: nj=4 align prob=-79.54 over 0.06h [retry=0.0%, fail=0.0%] states=70 gauss
steps/train_mono.sh: Done training monophone system in exp/mono

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

WARNING: the --mono, --left-biphone and --quinphone options are now deprecated and
tree-info exp/mono/tree
tree-info exp/mono/tree
fstdeterminizestar --use-log=true
fsttablecompose data/lang/L_disambig.fst data/lang/G.fst
fstminimizeencoded
fstpushspecial
fstisstochastic data/lang/tmp/LG.fst
-0.0421835 -0.0423222
[info]: LG not stochastic.
```

```
fstcomposecontext --context-size=1 --central-position=0 --read-disambig-syms=data/
fstisstochastic data/lang/tmp/CLG_1_0.fst
-0.0421835 -0.0423222
[info]: CLG not stochastic.
make-h-transducer --disambig-syms-out=exp/mono/graph/disambig_tid.int --transition-
fsttablecompose exp/mono/graph/Ha.fst data/lang/tmp/CLG_1_0.fst
fstminimizeencoded
fstrmepslocal
fstdeterminizestar --use-log=true
fstrmsymbols exp/mono/graph/disambig_tid.int
fstisstochastic exp/mono/graph/HCLGa.fst
0.000319138 -0.0427566
HCLGa is not stochastic
add-self-loops --self-loop-scale=0.1 --reorder=true exp/mono/final.mdl
steps/decode.sh --config conf/decode.config --nj 4 --cmd run.pl exp/mono/graph data
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/mono/graph exp/mono/decode
steps/diagnostic/analyze_lats.sh: see stats in exp/mono/decode/log/analyze_alignmen
Overall, lattice depth (10,50,90-percentile)=(1,1,2) and mean=1.2
steps/diagnostic/analyze_lats.sh: see stats in exp/mono/decode/log/analyze_lattice_
exp/mono/decode/wer_10
%WER 1.39 [ 2 / 144, 1 ins, 0 del, 1 sub ]
%SER 4.17 [ 2 / 48 ]
exp/mono/decode/wer_11
%WER 1.39 [ 2 / 144, 1 ins, 0 del, 1 sub ]
%SER 4.17 [ 2 / 48 ]
exp/mono/decode/wer_12
.....................
%SER 4.17 [ 2 / 48 ]
exp/mono/decode//wer_8
%WER 1.39 [ 2 / 144, 1 ins, 0 del, 1 sub ]
%SER 4.17 [ 2 / 48 ]
exp/mono/decode//wer_9
%WER 1.39 [ 2 / 144, 1 ins, 0 del, 1 sub ]
%SER 4.17 [ 2 / 48 ]

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

steps/align_si.sh --nj 4 --cmd run.pl data/train data/lang exp/mono exp/mono_ali
```

```
steps/align_si.sh: feature type is delta
steps/align_si.sh: aligning data in data/train using model from exp/mono, putting a
steps/diagnostic/analyze_alignments.sh --cmd run.pl data/lang exp/mono_ali
steps/diagnostic/analyze_alignments.sh: see stats in exp/mono_ali/log/analyze_align
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/mono_ali exp
steps/train_deltas.sh: accumulating tree stats
steps/train_deltas.sh: getting questions for tree-building, via clustering
steps/train_deltas.sh: building the tree
WARNING (gmm-init-model:InitAmGmm():gmm-init-model.cc:55) Tree has pdf-id 1 with no
** The warnings above about 'no stats' generally mean you have phones **
** (or groups of phones) in your phone set that had no corresponding data. **
** 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 current 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: 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
steps/diagnostic/analyze_alignments.sh: see stats in exp/tri1/log/analyze_alignment
382 warnings in exp/tri1/log/update.*.log
1 warnings in exp/tri1/log/questions.log
1 warnings in exp/tri1/log/build_tree.log
12 warnings in exp/tri1/log/align.*.*.log
28 warnings in exp/tri1/log/init_model.log
1 warnings in exp/tri1/log/mixup.log
exp/tri1: nj=4 align prob=-79.41 over 0.06h [retry=0.0%, fail=0.0%] states=88 gauss
steps/train_deltas.sh: Done training system with delta+delta-delta features in exp/
```

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

tree-info exp/tri1/tree
tree-info exp/tri1/tree
fstcomposecontext --context-size=3 --central-position=1 --read-disambig-syms=data/
fstisstochastic data/lang/tmp/CLG_3_1.fst
0 -0.0423222
[info]: CLG not stochastic.
make-h-transducer --disambig-syms-out=exp/tri1/graph/disambig_tid.int --transition
fsttablecompose exp/tri1/graph/Ha.fst data/lang/tmp/CLG_3_1.fst
fstdeterminizestar --use-log=true
fstminimizeencoded
fstrmsymbols exp/tri1/graph/disambig_tid.int
fstrmepslocal
fstisstochastic exp/tri1/graph/HCLGa.fst
0.000384301 -0.0999308
HCLGa is not stochastic
add-self-loops --self-loop-scale=0.1 --reorder=true exp/tri1/final.mdl
steps/decode.sh --config conf/decode.config --nj 4 --cmd run.pl exp/tri1/graph data
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/tri1/graph exp/tri1/decode
steps/diagnostic/analyze_lats.sh: see stats in exp/tri1/decode/log/analyze_alignmen
Overall, lattice depth (10,50,90-percentile)=(1,1,1) and mean=1.1
steps/diagnostic/analyze_lats.sh: see stats in exp/tri1/decode/log/analyze_lattice_
exp/tri1/decode/wer_10
...................


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

TAGS ASR kaldi

PREVIOUS POST NEXT POST

kaldi 笔记：数据准备 kaldi 笔记：部分术语词汇表