Building a Mandarin ASR with Kaldi and NER-Trs-Vol1 Corpus

Compiled from:

- Sanjeev Khudanpur, Dan Povey and Jan Trmal, "Building Speech Recogni0on
  Eleanor Chodroff, "Corpus Phonetics Tutorial/Kaldi",
  https://www.eleanorchodroff.com/tutorial/kaldi/kaldi-familiarization.html
- 篠崎隆宏, Kaldiツールキットを用いた 音声認識システムの構築 - 東京工業大学, http://www.ts.ip.titech.ac.jp/demos/csjkaldisp2016oct.pdf

and many other sources.

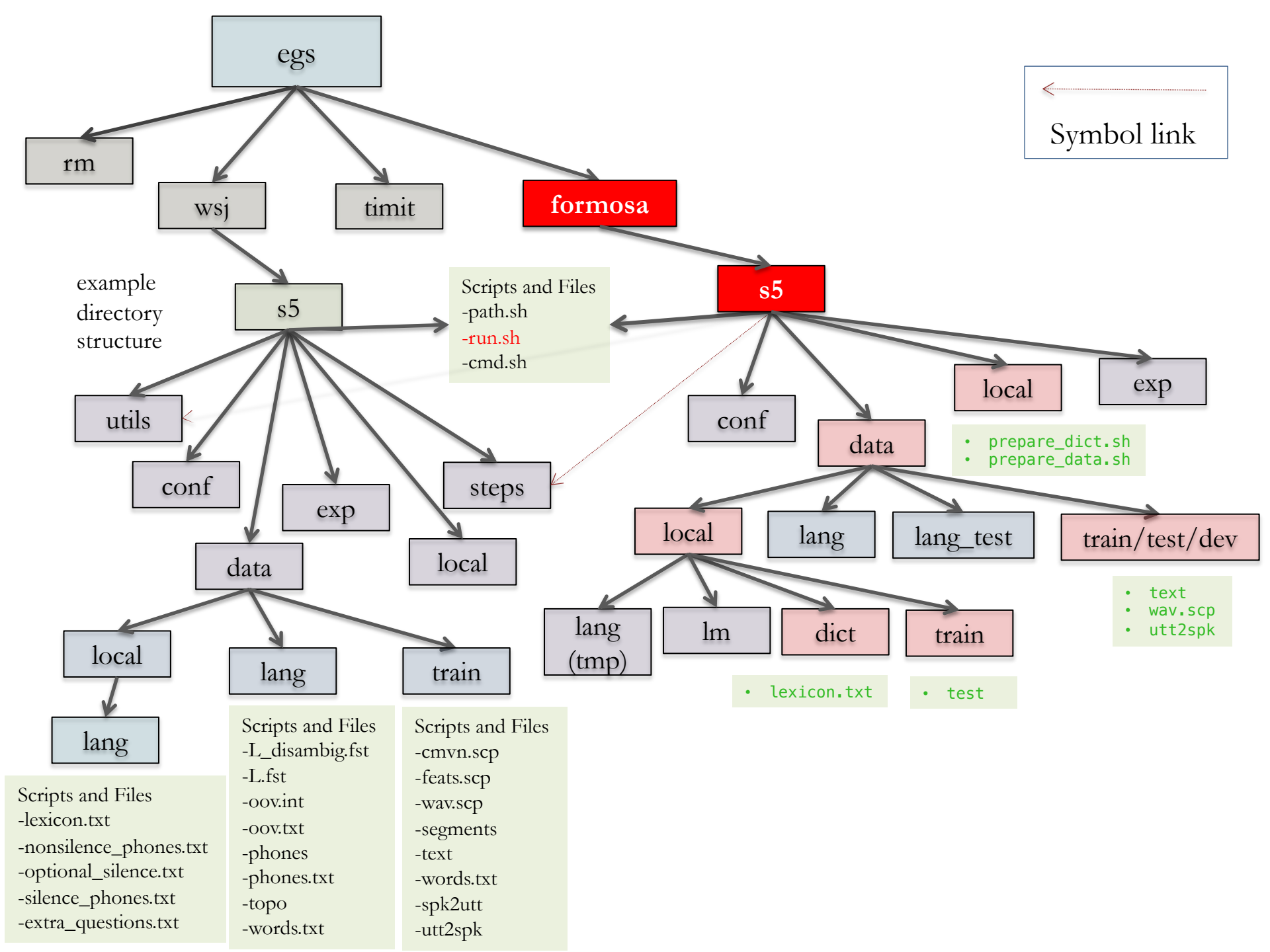
Yuan-Fu Liao
National Taipei University of Technology
yfliao@ntutedu.tw

Formosa Recipe

- git clone https://github.com/yfliao/kaldi.git

kaldi
- egs/formosa/s5
- misc
- src
- tools
- windows

Files	Content
cmd.sh	Environment
path.sh	Path
run.sh	Scripts
data/	Corpus
@steps/, @utils/	Universal scripts (symbol link)
local/	Local scripts


egs
rm
wsj
timit
formosa
Symbol link
example
directory
structure
s5
Scripts and Files
-path.sh
-run.sh
-cmd.sh
s5
utils
conf
exp
steps
local
data
local
lang
train
lang
Scripts and Files
-lexicon.txt
-nonsilence_phones.txt
-optional_silence.txt
-silence_phones.txt
-extra_questions.txt
Scripts and Files
-L_disambig.fst
-L.fst
-oov.int
-oov.txt
-phones
-phones.txt
-topo
-words.txt
Scripts and Files
-cmvn.scp
-feats.scp
-wav.scp
-segments
-text
-words.txt
-spk2utt
-utt2spk
conf
data
local
exp
• prepare_dict.sh
• prepare_data.sh
local
lang
lang_test
train/test/dev
lang (tmp)
lm
dict
train
• lexicon.txt
• test
• text
• wav.scp
• utt2spk

NER-Trs-Vol1 Corpus

- Copy/symbol link NER-Trs-Vol1 to formosa/s5/

```
liao@gchead:~/GitHub/kaldi/egs/formosa/s5$ ls -l
total 68
-rwxr-xr-x  1 liao ntut 1017 Jun  5 13:37 cmd.sh
drwxr-xr-x  2 liao ntut  133 Jun  5 13:45 conf
drwxr-xr-x 16 liao ntut 4096 Jun 29 09:35 data
drwxr-xr-x 20 liao ntut 4096 Jun 29 07:15 exp
drwxr-xr-x  4 liao ntut 4096 Jun 27 16:19 local
drwxr-xr-x  2 liao ntut 4096 Jun 27 14:15 mfcc
drwxr-xr-x  2 liao ntut 8192 Jun 27 20:26 mfcc_perturbed
drwxr-xr-x  2 liao ntut 4096 Jun 27 22:05 mfcc_perturbed_hires
lrwxrwxrwx  1 liao ntut   56 Jun  8 03:11 NER-Trs-Vol1 -> /home/liao/Corpora/TaiwaneseSpeechInTheWild/NER-Trs-Vol1
-rwxr-xr-x  1 liao ntut  373 Jun  4 12:00 path.sh
-rw-r--r--  1 liao ntut 2273 Jun  6 15:08 RESULTS
-rwxr-xr-x  1 liao ntut  454 Jun  4 12:00 result.sh
-rwxr-xr-x  1 liao ntut 6409 Jun 29 04:14 run.sh
-rwxr-xr-x  1 liao ntut 6345 Jun 14 08:05 run.sh.bak
lrwxrwxrwx  1 liao ntut   18 Jun  4 13:54 steps -> ../../wsj/s5/steps
lrwxrwxrwx  1 liao ntut   18 Jun  4 13:54 utils -> ../../wsj/s5/utils
liao@gchead:~/GitHub/kaldi/egs/formosa/s5$
```

NER-Trs-Vol1 Corpus

- Lexicon

```
liao@gchead:~/GitHub/kaldi/egs/formosa/s5$ ls NER-Trs-Vol1/Language
lexicon.txt
liao@gchead:~/GitHub/kaldi/egs/formosa/s5$
```

- Database

```
liao@gchead:~/GitHub/kaldi/egs/formosa/s5$ tree -L 2 NER-Trs-Vol1/Train/
NER-Trs-Vol1/Train/
├── Clean
│   ├── Text
│   └── Wav
└── Other
    ├── Text
    └── Wav

6 directories, 0 files
liao@gchead:~/GitHub/kaldi/egs/formosa/s5$
```

Lexicon/X-SAMPA

一 i:1
一一 i:1 i:1
一一九 i:1 i:1 ts6 j oU3
一一二 i:1 i:1 axr4
一一二年 i:1 i:1 axr4 n j A: n2
一一分 i:1 i:1 f ax n1
一一列舉 i:1 i:1 l j E4 ts6 y3
一一對 i:1 i:1 t w eI4
一一對應 i:1 i:1 t w eI4 j ax N4
一一年 i:1 i:1 n j A: n2
一丁 i:1 t j ax N1
一丁 i:4 t j ax N1
一丁點 i:4 t j ax N1 t j A: n3
一丁點兒 i:4 t j ax N1 t j A: n3 axr2
一七 i:1 ts6_h i:1
一七一 i:1 ts6_h i:1 i:1
一七一三 i:1 ts6_h i:1 i:1 s A: n1
一七七 i:1 ts6_h i:1 ts6_h i:1
一七三 i:1 ts6_h i:1 s A: n1
一七二 i:1 ts6_h i:1 axr4
一七二五 i:1 ts6_h i:1 axr4 u:3
一七五 i:1 ts6_h i:1 u:3
一七八 i:1 ts6_h i:1 p A:1
一七六 i:1 ts6_h i:1 l j oU4
一七四 i:1 ts6_h i:1 s4

Consonants			
BoPoMo	IPA	X-SAMPA	Phone
ㄅ	p	p	p
ㄆ	pʰ	p_h	p_h
ㄇ	m	m	m
ㄈ	f	f	f
ㄉ	t	t	t
ㄊ	tʰ	t_h	t_h
ㄋ	n	n	n
ㄌ	l	l	l
ㄍ	k	k	k
ㄎ	kʰ	k_h	k_h
ㄏ	x	x	x
ㄐ	tɕ	ts\	ts6
ㄑ	tɕʰ	ts_h	ts6_h
ㄒ	ɕ	s\	s6
ㄓ	ʈʂ	t`s`	ttss
ㄔ	ʈʂʰ	t`s`_h	ttss_h
ㄕ	ʂ	s`	ss
ㄖ	ʐ	z`	zz
ㄗ	ts	ts	ts
ㄘ	tsʰ	ts_h	ts_h
ㄙ	s	s	s
ㄧ	j	j	j
ㄨ	w	w	w
ㄩ	ɥ	H	H
ㄥ	ŋ	N	N

Vowels			
BoPoMo	IPA	X-SAMPA	Phone
ㄚ	a	A:	A:
ㄛ	ɔ	O:	O:
ㄜ	ə	@	ax
ㄝ	ɛ	E	E
ㄞ	aɪ	aI	aI
ㄟ	eɪ	eI	eI
ㄠ	ɑʊ	aU	aU
ㄡ	oʊ	oU	oU
ㄢ	a+n	A:+n	A:+n
ㄣ	ə+n	ax+n	ax+n
ㄤ	a+ŋ	A:+N	A:+N
ㄥ	ə+ŋ	ax+N	ax+N
ㄦ	ɚ	@`	axr
ㄧ	i	i:	i:
ㄨ	u	u:	u:
ㄩ	y	y	y

Corpus

Show	CN	Hrs	Utt.
創設市集 (Maker Market On-Air)	CS	14.4	4,028
技職最前線 (Frontier in Technological Education)	JZ	1.8	438
國際教育心動線 (International Education Outlook)	GJ	3.2	640
多愛自己一點點 (Love Yourself More)	DA	13.6	2,347
科學SoEasy (Science So Easy)	KX	1.8	208
青年故事館 (Young Creators)	QG	17.3	3,202
不太乖學堂 (Experimental Education)	BG	9.5	1,568
星期講座 (Weekly Lecture)	WK	8.4	1,102
遇見幸福幼兒園 (Non-Profit Preschools, NP)	YK	5.6	826
收藏人生 (Story of Collectors)	SR	16.5	2,670
雙語新聞 (Bilingual News)	SY	34.5	4,015
Total		126.6	21,044

```
# data preparation
if [ $stage -le -2 ]; then


  # Lexicon Preparation,
  echo "$0: Lexicon Preparation"
  local/prepare_dict.sh || exit 1;


  # Data Preparation
  echo "$0: Data Preparation"
  local/prepare_data.sh || exit 1;


  # Phone Sets, questions, L compilation
  echo "$0: Phone Sets, questions, L compilation Preparation"
  rm -rf data/lang
  utils/prepare_lang.sh --position-dependent-phones false data/local/dict \
      "<SIL>" data/local/lang data/lang || exit 1;


  # LM training
  echo "$0: LM training"
  rm -rf data/local/lm/3gram-mincount
  local/train_lms.sh || exit 1;


  # G compilation, check LG composition
  echo "$0: 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;


fi
```

Data Folder

- data/

File	Content
train/	Training set
test/	Test Set
dev/	Development set
dict/	Dictionary
graph/	Language model

  - train/, test/, dev/

File	Content
wav.scp	Waveform paths
text	Transcriptions
utt2spk	Utterances vs. Speakers
spk2utt	Speakers vs. Utterances

train/, test/, dev/

wav.scp

Speaker0001-0	~/kaldi-data/OC16-CE80/Training_Set/train/Speaker0001/0.wav
Speaker0001-1	~/kaldi-data/OC16-CE80/Training_Set/train/Speaker0001/1.wav
Speaker0001-10	~/kaldi-data/OC16-CE80/Training_Set/train/Speaker0001/10.wav

text

Speaker0001-0	打开 Notepad 编辑 文件
Speaker0001-1	结束 teleconference
Speaker0001-10	Capital Hotel 你 觉 的 怎么样

utt2spk

Speaker0001-0	Speaker0001
Speaker0001-1	Speaker0001
Speaker0001-10	Speaker0001

spk2utt

Speaker0001	Speaker0001-0 Speaker0001-1 Speaker0001-10 Speaker0001-11 … … …
Speaker0003	Speaker0003-1 Speaker0003-10 Speaker0003-11 Speaker0003-115 … … …
Speaker0004	Speaker0004-0 Speaker0004-1 Speaker0004-10 Speaker0004-11 … … …

Dictionary

- data/dict/

File	Content
lexicon.txt	dictionary
nonsilence_phones.txt	Phoneme set
silence_phones.txt	Silence
extra_questions.txt	Question set
optional_silence.txt	optional Silence

lexicon.txt

```
# sil
<SPOKEN_NOISE>      sil
<UNK>               sil
SIL                 sil
X光                 EH1 K S g uang1
X光线               EH1 K S g uang1 x
ian4
T恤                 T x v4
```

nonsilence_phones.txt

```
a1
a2
a3
a4
a5
aa
```

silence_phones.txt

```
sil
```

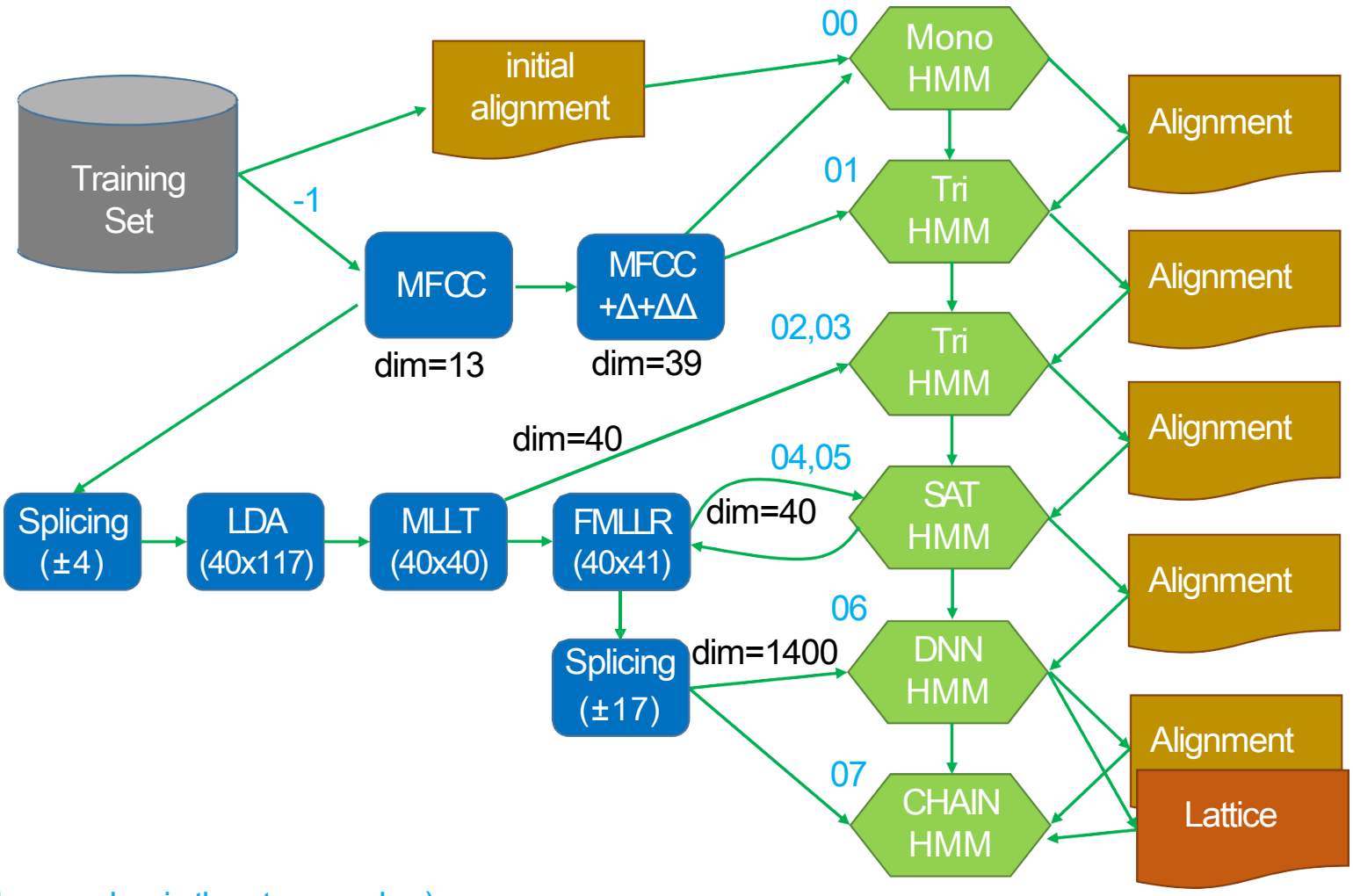
extra_questions.txt

```
sil
```

optional_silence.txt

```
sil
```

Training Procedure



(Blue number is the step number)

MFCC Extraction

```
mfccdir=mfcc


# mfcc
if [ $stage -le -1 ]; then


  echo "$0: making mfccs"
  for x in train test; do
    steps/make_mfcc_pitch.sh --cmd "$train_cmd" --nj $num_jobs data/$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


fi
```

mono

```
# mono
if [ $stage -le 0 ]; then


  echo "$0: train mono model"
  # Make some small data subsets for early system-build stages.
  echo "$0: make training subsets"
  utils/subset_data_dir.sh --shortest data/train 3000 data/train_mono


  # train mono
  steps/train_mono.sh --boost-silence 1.25 --cmd "$train_cmd" --nj $num_jobs \
    data/train_mono data/lang exp/mono || exit 1;


  # Get alignments from monophone system.
  steps/align_si.sh --boost-silence 1.25 --cmd "$train_cmd" --nj $num_jobs \
    data/train data/lang exp/mono exp/mono_ali || exit 1;


  # Monophone decoding
  (
  utils/mkgraph.sh data/lang_test exp/mono exp/mono/graph || exit 1;
  steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj $num_jobs \
    exp/mono/graph data/test exp/mono/decode_test
  )&


fi
```

tri1, tri2

```
# tri1
if [ $stage -le 1 ]; then


  echo "$0: train tri1 model"
  # train tri1 [first triphone pass]
  steps/train_deltas.sh --boost-silence 1.25 --cmd "$train_cmd" \
   2500 20000 data/train data/lang exp/mono_ali exp/tri1 || exit 1;


  # align tri1
  steps/align_si.sh --cmd "$train_cmd" --nj $num_jobs \
    data/train data/lang exp/tri1 exp/tri1_ali || exit 1;


  # decode tri1
  (
  utils/mkgraph.sh data/lang_test exp/tri1 exp/tri1/graph || exit 1;
  steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj $num_jobs \
    exp/tri1/graph data/test exp/tri1/decode_test
  )&


fi
```

tri3

```
# tri3a
if [ $stage -le 3 ]; then


  echo "$-: train tri3 model"
  # Train tri3a, which is LDA+MLLT,
  steps/train_lda_mllt.sh --cmd "$train_cmd" \
   2500 20000 data/train data/lang exp/tri2_ali exp/tri3a || exit 1;


  # decode tri3a
  (
  utils/mkgraph.sh data/lang_test exp/tri3a exp/tri3a/graph || exit 1;
  steps/decode.sh --cmd "$decode_cmd" --nj $num_jobs --config conf/decode.config \
    exp/tri3a/graph data/test exp/tri3a/decode_test
  )&


fi
```

tri4

```
# tri4
if [ $stage -le 4 ]; then

  echo "$0: train tri4 model"
  # From now, we start building a more serious system (with SAT), and we'll
  # do the alignment with fMLLR.

  steps/align_fmllr.sh --cmd "$train_cmd" --nj $num_jobs \
    data/train data/lang exp/tri3a exp/tri3a_ali || exit 1;


  steps/train_sat.sh --cmd "$train_cmd" \
    2500 20000 data/train data/lang exp/tri3a_ali exp/tri4a || exit 1;


  # align tri4a
  steps/align_fmllr.sh  --cmd "$train_cmd" --nj $num_jobs \
    data/train data/lang exp/tri4a exp/tri4a_ali


  # decode tri4a
  (
  utils/mkgraph.sh data/lang_test exp/tri4a exp/tri4a/graph
  steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $num_jobs --config conf/decode.config \
    exp/tri4a/graph data/test exp/tri4a/decode_test
  )&

fi
```

nnet3/tdnn

```
# nnet3 tdnn models
if [ $stage -le 6 ]; then

  echo "$0: train nnet3 model"
  local/nnet3/run_tdnn.sh --stage $train_stage

fi
```

```
if [ $stage -le 7 ]; then
  echo "$0: creating neural net configs";


  num_targets=$(tree-info $ali_dir/tree |grep num-pdfs|awk '{print $2}')


  mkdir -p $dir/configs
  cat <<EOF > $dir/configs/network.xconfig
  input dim=100 name=ivector
  input dim=43 name=input

  # please note that it is important to have input layer with the name=input
  # as the layer immediately preceding the fixed-affine-layer to enable
  # the use of short notation for the descriptor
  fixed-affine-layer name=lda input=Append(-2,-1,0,1,2,ReplaceIndex(ivector, t, 0)) affine-transform-file=$dir/configs/l

  # the first splicing is moved before the lda layer, so no splicing here
  relu-batchnorm-layer name=tdnn1 dim=850
  relu-batchnorm-layer name=tdnn2 dim=850 input=Append(-1,0,2)
  relu-batchnorm-layer name=tdnn3 dim=850 input=Append(-3,0,3)
  relu-batchnorm-layer name=tdnn4 dim=850 input=Append(-7,0,2)
  relu-batchnorm-layer name=tdnn5 dim=850 input=Append(-3,0,3)
  relu-batchnorm-layer name=tdnn6 dim=850
  output-layer name=output input=tdnn6 dim=$num_targets max-change=1.5
EOF
  steps/nnet3/xconfig_to_configs.py --xconfig-file $dir/configs/network.xconfig --config-dir $dir/configs/
fi
```

chain/tdnn

```
# chain models
if [ $stage -le 7 ]; then

  echo "$0: train nnet3 model"
  local/chain/run_tdnn.sh --stage $train_stage

fi
```

```
if [ $stage -le 10 ]; then
  echo "$0: creating neural net configs using the xconfig parser";

  num_targets=$(tree-info $treedir/tree |grep num-pdfs|awk '{print $2}')
  learning_rate_factor=$(echo "print 0.5/$xent_regularize" | python)

  mkdir -p $dir/configs
  cat <<EOF > $dir/configs/network.xconfig
  input dim=100 name=ivector
  input dim=43 name=input

  # please note that it is important to have input layer with the name=input
  # as the layer immediately preceding the fixed-affine-layer to enable
  # the use of short notation for the descriptor
  fixed-affine-layer name=lda input=Append(-1,0,1,ReplaceIndex(ivector, t, 0)) affine-transform-file=$dir/configs/lda.ma

  # the first splicing is moved before the lda layer, so no splicing here
  relu-batchnorm-layer name=tdnn1 dim=625
  relu-batchnorm-layer name=tdnn2 input=Append(-1,0,1) dim=625
  relu-batchnorm-layer name=tdnn3 input=Append(-1,0,1) dim=625
  relu-batchnorm-layer name=tdnn4 input=Append(-3,0,3) dim=625
  relu-batchnorm-layer name=tdnn5 input=Append(-3,0,3) dim=625
  relu-batchnorm-layer name=tdnn6 input=Append(-3,0,3) dim=625

  ## adding the layers for chain branch
  relu-batchnorm-layer name=prefinal-chain input=tdnn6 dim=625 target-rms=0.5
  output-layer name=output include-log-softmax=false dim=$num_targets max-change=1.5

  # adding the layers for xent branch
    relu-batchnorm-layer name=prefinal-xent input=tdnn6 dim=625 target-rms=0.5
  output-layer name=output-xent dim=$num_targets learning-rate-factor=$learning_rate_factor max-change=1.5

EOF
  steps/nnet3/xconfig_to_configs.py --xconfig-file $dir/configs/network.xconfig --config-dir $dir/configs/
fi
```

result.sh

```
echo "WER: test"
for x in exp/*/decode_test*; do [ -d $x ] && grep WER $x/wer_* | utils/best_wer.sh; done 2>/dev/null
for x in exp/*/*/decode_test*; do [ -d $x ] && grep WER $x/wer_* | utils/best_wer.sh; done 2>/dev/null
echo

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

Results

- Baseline

Model	WER (%)	CER (%)
Mono	61.32	54.09
Tri1	41.00	32.61
Tri2	40.41	32.10
Tri3	38.67	30.40
Tri4	35.70	27.53
Tri5	32.11	24.21
Nnet3/TDNN	24.43	17.07
Chain/TDNN	23.97	16.86

- FSR 2018

ID	CER	CRR	SER
A	17.31	83.59	98.61
B	24.28	75.99	99.53
C	17.07	83.55	95.12
D	89.65	10.37	100.00
E	11.93	88.98	94.70
F	13.20	87.78	96.10
G	10.53	90.58	91.12
H	13.24	88.27	91.40
I	17.31	83.59	98.61
J	100.00	0.00	100.00
K	21.06	81.25	97.07
L	16.22	86.14	96.47
M	21.32	80.29	98.70
baseline	16.64	84.48	98.14