Data Preparation

1. audio data

kaldi/egs/doremi/doremi\_audio/train 에 aey\_0\_0\_0\_0 ~ bey\_6\_6\_6\_6

kaldi/egs/doremi/doremi\_audio/test 에 cey\_0\_0\_0\_0

1. acoustic data

kaldi/egs/doremi/data/train과 kaldi/egs/doremi/data/test에 각각 아래 파일들 작성

spk2gender

aey f

bey f

wav.scp

aey\_0\_1\_5\_2 /Users/user/ES/kaldi/egs/doremi/doremi\_audio/train/aey\_0\_1\_5\_2.wav

aey\_0\_2\_1\_3 /Users/user/ES/kaldi/egs/doremi/doremi\_audio/train/aey\_0\_2\_1\_3.wav

text

aey\_0\_1\_5\_2 do re la mi

aey\_0\_2\_1\_3 do mi re fa

utt2spk

aey\_0\_1\_5\_2 aey

aey\_0\_2\_1\_3 aey

kaldi/egs/doremi/data/local에 아래 파일 작성

corpus.txt

fa re mi si

fa si sol la

1. language data

kaldi/egs/doremi/data/local/dict

lexicon.txt

!SIL sil

<UNK> spn

do d o

fa f a

la l a

mi m i

re r e

si s i

sol s o l

nonsilence\_phones.txt

a

d

e

f

i

l

m

o

r

s

silence\_phones.txt

sil

spn

optional\_silence.txt

sil

1. necessary kaldi tools

kaldi/egs/wsj/s5/steps와 kaldi/egs/wsj/s5/utils를 아래 경로로 복사해옴

kaldi/egs/doremi/steps

kaldi/egs/doremi/utils

1. srilm 설치

kaldi/tools/install\_srilm.sh

http://www.speech.sri.com/projects/srilm/download.html에서 srilm.tgz를 다운받아 위 스크립트를 이용하여 설치

1. configuration file

kaldi/egs/doremi/conf에

decode.config

first\_beam=10.0

beam=13.0

lattice\_beam=6.0

mfcc.conf

--use-energy=false

--allow\_downsample=true

1. kaldi/egs/doremi/cmd.sh

# Setting local system jobs (local CPU - no external clusters)

export train\_cmd=run.pl

export decode\_cmd=run.pl

1. kaldi/egs/doremi/path.sh

# Defining Kaldi root directory

export KALDI\_ROOT=`pwd`/../..

# Setting paths to useful tools

export PATH=$PWD/utils/:$KALDI\_ROOT/src/bin:$KALDI\_ROOT/tools/openfst/bin:$KALDI\_ROOT/src/fstbin/:$KALDI\_ROOT/src/gmmbin/:$KALDI\_ROOT/src/featbin/:$KALDI\_ROOT/src/lmbin/:$KALDI\_ROOT/src/sgmm2bin/:$KALDI\_ROOT/src/fgmmbin/:$KALDI\_ROOT/src/latbin/:$PWD:$PATH

# Defining audio data directory (modify it for your installation directory!)

export DATA\_ROOT="/Users/user/ES/kaldi/egs/doremi/doremi\_audio"

# Enable SRILM

. $KALDI\_ROOT/tools/env.sh

# Variable needed for proper data sorting

export LC\_ALL=C

1. run.sh

#!/bin/bash

. ./path.sh || exit 1

. ./cmd.sh || exit 1

nj=1 # number of parallel jobs - 1 is perfect for such a small data set

lm\_order=1 # language model order (n-gram quantity) - 1 is enough for digits grammar

# Safety mechanism (possible running this script with modified arguments)

. 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

# Needs to be prepared by hand (or using self written scripts):

#

# spk2gender [<speaker-id> <gender>]

# wav.scp [<uterranceID> <full\_path\_to\_audio\_file>]

# text [<uterranceID> <text\_transcription>]

# utt2spk [<uterranceID> <speakerID>]

# corpus.txt [<text\_transcription>]

# 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 data sorting

# utils/validate\_data\_dir.sh data/train # script for checking prepared data - here: for data/train directory

# utils/fix\_data\_dir.sh data/train # tool for data proper sorting if needed - here: for data/train directory

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

# Needs to be prepared by hand (or using self written scripts):

#

# lexicon.txt [<word> <phone 1> <phone 2> ...]

# nonsilence\_phones.txt [<phone>]

# silence\_phones.txt [<phone>]

# optional\_silence.txt [<phone>]

# 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\_srilm.sh"

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 || exit 1

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

echo

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

echo

steps/align\_si.sh --nj $nj --cmd "$train\_cmd" data/train data/lang exp/mono exp/mono\_ali || exit 1

echo

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

echo

steps/train\_deltas.sh --cmd "$train\_cmd" 2000 11000 data/train data/lang exp/mono\_ali exp/tri1 || exit 1

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

echo

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

echo

GO Web Server

package main

import (

"fmt"

"log"

"net/http"

"strings"

)

func logging(f http.HandlerFunc) http.HandlerFunc {

return func(w http.ResponseWriter, r \*http.Request) {

log.Println("MiddlerWare Test(Log) :", r.URL.Path)

f(w, r)

}

}

func reqtest1(w http.ResponseWriter, r \*http.Request) {

fmt.Fprintln(w, "Requested : (/req-test1)")

}

func reqtest2(w http.ResponseWriter, r \*http.Request) {

fmt.Fprintln(w, "Requested : (/req-test2)")

}

func defaultHandler(w http.ResponseWriter, r \*http.Request) {

r.ParseForm()

fmt.Println("default : ", r.Form)

fmt.Println("path", r.URL.Path)

fmt.Println("param : ", r.Form["test\_param"])

for k, v := range r.Form {

fmt.Println("key : ", k)

fmt.Println("value : ", strings.Join(v, ""))

}

fmt.Fprintf(w, "Golang Webserver is Working!")

}

func main() {

http.HandleFunc("/req-test1", logging(reqtest1))

http.HandleFunc("/req-test2", logging(reqtest2))

http.HandleFunc("/", defaultHandler)

fmt.Println("Golang Webserver is Working!!!!!")

err := http.ListenAndServe(":9080", nil)

if err != nil {

log.Fatal("ListenAndServe: ", err)

} else {

fmt.Println("ListenAndServe Started! -> Port(9090)")

}

}

package main

import (

"fmt"

"log"

"os/exec"

)

func main() {

out, err := exec.Command("/bin/sh", "run.sh").Output()

if err != nil {

log.Fatal(err)

}

fmt.Printf("%s\n", out)

}

/Users/user/ES/kaldi/egs/doremi/exp/mono/decode/log/decode.1.log