<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/ForcedAlign/run.sh>

#!/bin/bash

set -e -o pipefail

dist\_path=/dist

tmp\_path=/tmp/work

data\_path=/data

model\_name=default

beam=20

retry\_beam=300

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1;

if [ $# -ne 3 ]; then

echo "Usage: ./run.sh <input-wav> <input-txt> <out-ctm>"

echo ""

echo "Options:"

echo " --model\_name"

echo " --beam"

echo " --retry\_beam"

exit 1

fi

#if file is an existing global path

if [ -f "$1" ] ; then

wav\_file=$(readlink -f $1)

txt\_file=$(readlink -f $2)

out=$(readlink -f $3)

else

#else it's within the $data\_path

wav\_file=$data\_path/$1

txt\_file=$data\_path/$2

out=$data\_path/$3

fi

for f in $wav\_file $txt\_file; do

[ ! -f "$f" ] && echo "no such file $f" && exit 1;

done

[ ! -d "${dist\_path}/model/${model\_name}" ] && echo "need to get the proper model: ${model\_name}" && exit 1;

if [ -e "$tmp\_path" ] ; then

rm -rf ${tmp\_path}

fi

mkdir -p ${tmp\_path}

mkdir ${tmp\_path}/data

echo input $wav\_file > ${tmp\_path}/data/wav.scp

echo input `cat $txt\_file` > ${tmp\_path}/data/text

echo input spk > ${tmp\_path}/data/utt2spk

echo spk input > ${tmp\_path}/data/spk2utt

cd ${tmp\_path}

ln -s ${dist\_path}/path.sh

ln -s ${dist\_path}/local\_utils

ln -s ${dist\_path}/model/${model\_name}/tri3b\_mmi

ln -s ${dist\_path}/model/${model\_name}/phonetisaurus

. path.sh

ln -s $KALDI\_ROOT/egs/wsj/s5/utils

ln -s $KALDI\_ROOT/egs/wsj/s5/steps

ln -s $KALDI\_ROOT/egs/wsj/s5/conf

ln -s $KALDI\_ROOT/egs/wsj/s5/local

./steps/make\_mfcc.sh --nj 1 data

./steps/compute\_cmvn\_stats.sh data

./local\_utils/prepare\_dict.sh data dict

./utils/prepare\_lang.sh dict "<unk>" tmp lang

./steps/align\_fmllr.sh --nj 1 --beam ${beam} --retry-beam ${retry\_beam} data lang tri3b\_mmi ali

./steps/get\_train\_ctm.sh data lang ali

./local\_utils/get\_phoneme\_ctm.sh data lang ali

awk '$0="@"$0' ali/phonectm | cat ali/ctm - | sort -r -k3n > $out

echo Finished generating alignment...

echo "Cleaning up..."

rm -rf ${tmp\_path}

<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/ForcedAlign/run_eaf.sh>

#!/bin/bash

set -e -o pipefail

dist\_path=/dist

tmp\_path=/tmp/work

data\_path=/data

nj=1

spk\_tier=true

skip\_tiers=

phones=true

model\_name=default

beam=20

retry\_beam=300

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1;

if [ $# -ne 3 ]; then

echo "Usage: ./run\_eaf.sh <input-wav> <input-eaf> <out-eaf>"

echo ""

echo "Options:"

echo " --model\_name"

echo " --beam"

echo " --retry\_beam"

echo " --nj"

echo " --spk-tier"

echo " --skip-tiers"

echo " --phones"

exit 1

fi

#if file is an existing global path

if [ -f "$1" ] ; then

wav\_file=$(readlink -f $1)

eaf\_in=$(readlink -f $2)

eaf\_out=$(readlink -f $3)

else

#else it's within the $data\_path

wav\_file=$data\_path/$1

eaf\_in=$data\_path/$2

eaf\_out=$data\_path/$3

fi

for f in $wav\_file $eaf\_in; do

[ ! -f "$f" ] && echo "no such file $f" && exit 1;

done

[ ! -d "${dist\_path}/model/${model\_name}" ] && echo "need to get the proper model: ${model\_name}" && exit 1;

if [ -e "$tmp\_path" ] ; then

rm -rf ${tmp\_path}

fi

mkdir -p ${tmp\_path}

mkdir ${tmp\_path}/data

cd ${tmp\_path}

ln -s ${dist\_path}/path.sh

ln -s ${dist\_path}/local\_utils

ln -s ${dist\_path}/model/${model\_name}/tri3b\_mmi

ln -s ${dist\_path}/model/${model\_name}/phonetisaurus

. path.sh

ln -s $KALDI\_ROOT/egs/wsj/s5/utils

ln -s $KALDI\_ROOT/egs/wsj/s5/steps

ln -s $KALDI\_ROOT/egs/wsj/s5/conf

ln -s $KALDI\_ROOT/egs/wsj/s5/local

echo input $wav\_file > data/wav.scp

echo "input input A" > data/reco2file\_and\_channel

if $spk\_tier ; then

python3 local\_utils/eaf2data.py --spk-tier --skip-tiers "$skip\_tiers" $eaf\_in data

else

python3 local\_utils/eaf2data.py --skip-tiers "$skip\_tiers" $eaf\_in data

fi

for f in data/\* ; do

sort -o $f $f

done

./utils/utt2spk\_to\_spk2utt.pl data/utt2spk > data/spk2utt

./steps/make\_mfcc.sh --nj $nj data

./steps/compute\_cmvn\_stats.sh data

./local\_utils/prepare\_dict.sh data dict

./utils/prepare\_lang.sh dict "<unk>" tmp lang

./steps/align\_fmllr.sh --nj $nj --beam ${beam} --retry-beam ${retry\_beam} data lang tri3b\_mmi ali

./steps/get\_train\_ctm.sh --use-segments false data lang ali

if $phones ; then

./local\_utils/get\_phoneme\_ctm.sh --use-segments false data lang ali

python3 local\_utils/ctm2eaf.py --phones-ctm ali/phonectm ali/ctm data/seg2tier ${eaf\_in} ${eaf\_out}

else

python3 local\_utils/ctm2eaf.py ali/ctm data/seg2tier ${eaf\_in} ${eaf\_out}

fi

echo Finished generating alignment...

echo "Cleaning up..."

rm -rf ${tmp\_path}

<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/ForcedAlign/run_segments.sh>

#!/bin/bash

set -e -o pipefail

dist\_path=/dist

tmp\_path=/tmp/work

data\_path=/data

nj=1

model\_name=default

beam=20

retry\_beam=300

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1;

if [ $# -ne 4 ]; then

echo "Usage: ./run.sh <input-wav> <seg-txt> <spk\_ctm> <out-ctm>"

echo ""

echo "Options:"

echo " --model\_name"

echo " --beam"

echo " --retry\_beam"

exit 1

fi

#if file is an existing global path

if [ -f "$1" ] ; then

wav\_file=$(readlink -f $1)

txt\_file=$(readlink -f $2)

spk\_ctm=$(readlink -f $3)

out=$(readlink -f $4)

else

#else it's within the $data\_path

wav\_file=$data\_path/$1

txt\_file=$data\_path/$2

spk\_ctm=$data\_path/$3

out=$data\_path/$4

fi

for f in $wav\_file $txt\_file $spk\_ctm; do

[ ! -f "$f" ] && echo "no such file $f" && exit 1;

done

[ ! -d "${dist\_path}/model/${model\_name}" ] && echo "need to get the proper model: ${model\_name}" && exit 1;

if [ -e "$tmp\_path" ] ; then

rm -rf ${tmp\_path}

fi

mkdir -p ${tmp\_path}

cd ${tmp\_path}

ln -s ${dist\_path}/path.sh

ln -s ${dist\_path}/local\_utils

ln -s ${dist\_path}/model/${model\_name}/tri3b\_mmi

ln -s ${dist\_path}/model/${model\_name}/phonetisaurus

. path.sh

ln -s $KALDI\_ROOT/egs/wsj/s5/utils

ln -s $KALDI\_ROOT/egs/wsj/s5/steps

ln -s $KALDI\_ROOT/egs/wsj/s5/conf

ln -s $KALDI\_ROOT/egs/wsj/s5/local

mkdir data

echo input $wav\_file > ${tmp\_path}/data/wav.scp

echo "input input A" > ${tmp\_path}/data/reco2file\_and\_channel

awk '{printf "seg\_%s\_%03d input %0.2f %0.2f\n", $5, NR, $3, $3+$4}' < $spk\_ctm > ${tmp\_path}/data/segments

awk '{printf "seg\_%s\_%03d spk\_%s\n", $5, NR, $5}' < $spk\_ctm > ${tmp\_path}/data/utt2spk

./utils/utt2spk\_to\_spk2utt.pl < ${tmp\_path}/data/utt2spk > ${tmp\_path}/data/spk2utt

cut -f1 -d' ' ${tmp\_path}/data/utt2spk > ${tmp\_path}/data/utt.id

paste -d ' ' ${tmp\_path}/data/utt.id ${txt\_file} > ${tmp\_path}/data/text

for f in ${tmp\_path}/data/\* ; do

sort -o $f $f

done

./steps/make\_mfcc.sh --nj $nj data

./steps/compute\_cmvn\_stats.sh data

./local\_utils/prepare\_dict.sh data dict

./utils/prepare\_lang.sh dict "<unk>" tmp lang

./steps/align\_fmllr.sh --nj $nj --beam ${beam} --retry-beam ${retry\_beam} data lang tri3b\_mmi ali

./steps/get\_train\_ctm.sh data lang ali

./local\_utils/get\_phoneme\_ctm.sh data lang ali

awk '$0="@"$0' ali/phonectm | cat ali/ctm - | sort -r -k3n > $out

echo Finished generating alignment...

echo "Cleaning up..."

rm -rf ${tmp\_path}

<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/SegmentAlign/run.sh>

#!/bin/bash

set -e -o pipefail

dist\_path=/dist

tmp\_path=/tmp/work

data\_path=/data

model\_name=default

beam=20

retry\_beam=800

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1;

if [ $# -ne 3 ]; then

echo "Usage: ./run.sh <input-wav> <input-txt> <proj-name>"

echo "Creates a folder <proj-name> and aligns the WAV/TXT inside it."

echo "Result is saved in <proj-name>/output.ctm"

echo ""

echo "Options:"

echo " --model\_name"

echo " --beam"

echo " --retry\_beam"

exit 1

fi

#if file is an existing global path

if [ -f "$1" ] ; then

wav\_file=$(readlink -f $1)

txt\_file=$(readlink -f $2)

out=$(readlink -f $3)

else

#else it's within the $data\_path

wav\_file=$data\_path/$1

txt\_file=$data\_path/$2

out=$data\_path/$3

fi

for f in $wav\_file $txt\_file; do

[ ! -f "$f" ] && echo "no such file $f" && exit 1;

done

[ ! -d "${dist\_path}/model/${model\_name}" ] && echo "need to get the proper model: ${model\_name}" && exit 1;

if [ -e "$tmp\_path" ] ; then

rm -rf ${tmp\_path}

fi

mkdir -p ${tmp\_path}

mkdir ${tmp\_path}/data

echo input $wav\_file > ${tmp\_path}/data/wav.scp

echo input `cat $txt\_file` > ${tmp\_path}/data/text

echo input spk > ${tmp\_path}/data/utt2spk

echo spk input > ${tmp\_path}/data/spk2utt

cd ${tmp\_path}

ln -s ${dist\_path}/path.sh

ln -s ${dist\_path}/local\_utils

ln -s ${dist\_path}/model/${model\_name}/tri3b\_mmi

ln -s ${dist\_path}/model/${model\_name}/phonetisaurus

. path.sh

ln -s $KALDI\_ROOT/egs/wsj/s5/utils

ln -s $KALDI\_ROOT/egs/wsj/s5/steps

ln -s $KALDI\_ROOT/egs/wsj/s5/conf

ln -s $KALDI\_ROOT/egs/wsj/s5/local

#feature extraction

./steps/make\_mfcc.sh --nj 1 data

./steps/compute\_cmvn\_stats.sh data

#data preparation

./local\_utils/prepare\_dict.sh data dict

./utils/prepare\_lang.sh dict "<unk>" tmp lang

#segmentation and cleaning

./steps/cleanup/clean\_and\_segment\_data.sh --nj 1 data lang tri3b\_mmi cleanup cleaned

echo "input input 1" > cleaned/reco2file\_and\_channel

#alignemnt of clean data

./steps/align\_fmllr.sh --nj 1 --beam ${beam} --retry-beam ${retry\_beam} cleaned lang tri3b\_mmi ali\_clean

#adaptation to clean data

./steps/train\_map.sh cleaned lang ali\_clean adapted

#resegmentation using adapted model

./steps/cleanup/clean\_and\_segment\_data.sh --nj 1 data lang adapted cleanup\_ad cleaned\_ad

echo "input input 1" > cleaned\_ad/reco2file\_and\_channel

#alignemnt of adapted clean data

./steps/align\_fmllr.sh --nj 1 --beam ${beam} --retry-beam ${retry\_beam} cleaned\_ad lang adapted ali\_ad

#make CTM in the ali folder

./steps/get\_train\_ctm.sh cleaned\_ad lang ali\_ad

python3 local\_utils/fix\_ctm.py ali\_ad/ctm ali\_ad/ctm.fixed

./local\_utils/get\_phoneme\_ctm.sh cleaned\_ad lang ali\_ad

python3 local\_utils/fix\_ctm.py ali\_ad/phonectm ali\_ad/phonectm.fixed

#get missing segments

sort -k3n ali\_ad/ctm -o ali\_ad/ctm.sorted

./local\_utils/get\_deleted\_seg.sh cleanup\_ad lang data ali\_ad/ctm.sorted deleted || true

if [ -s deleted/segments ] ; then #if there are any missing segments

#force realign missing segments

./steps/align\_fmllr.sh --nj 1 --beam ${beam} --retry-beam ${retry\_beam} deleted lang adapted ali\_deleted

echo "input input 1" > deleted/reco2file\_and\_channel

./steps/get\_train\_ctm.sh deleted lang ali\_deleted

python3 local\_utils/fix\_ctm.py ali\_deleted/ctm ali\_deleted/ctm.fixed

./local\_utils/get\_phoneme\_ctm.sh deleted lang ali\_deleted

python local\_utils/fix\_ctm.py ali\_deleted/phonectm ali\_deleted/phonectm.fixed

#merge CTMs

cat ali\_ad/ctm.fixed ali\_deleted/ctm.fixed > ctm.combined

python3 local\_utils/fix\_ctm.py ctm.combined words.ctm

cat ali\_ad/phonectm.fixed ali\_deleted/phonectm.fixed > phonectm.combined

python3 local\_utils/fix\_ctm.py phonectm.combined phonemes.ctm

else

cp ali\_ad/ctm.fixed words.ctm

cp ali\_ad/phonectm.fixed phonemes.ctm

fi

awk '$0="@"$0' phonemes.ctm | cat words.ctm - | sort -r -k3n > $out

echo Finished generating alignment...

<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/misc/transcribe_word_list.sh>

#!/bin/bash

set -e -o pipefail

data\_path=/data

dist\_path=/dist

model\_path=/dist/model/default/phonetisaurus

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1

if [ $# -ne 2 ]; then

echo "Usage: ./local/transcribe\_word\_list.sh <word\_list> <lexicon>"

exit 1

fi

if [ -f "$1" ]; then

word\_list=$(readlink -f $1)

lexicon=$(readlink -f $2)

else

#else it's within the $data\_path

word\_list=$data\_path/$1

lexicon=$data\_path/$2

fi

ln -s $dist\_path/path.sh

. path.sh

export LD\_LIBRARY\_PATH=$KALDI\_ROOT/tools/openfst/lib

python2.7 $KALDI\_ROOT/tools/phonetisaurus-g2p/src/scripts/phonetisaurus-apply --model $model\_path/model.fst --lexicon $model\_path/lexicon.txt --word\_list $word\_list -p 0.8 >$lexicon

<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/misc/train_g2p.sh>

#!/bin/bash

set -e -o pipefail

data\_path=/data

dist\_path=/dist

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1

if [ $# -ne 2 ]; then

echo "Usage: ./local/train\_g2p.sh <lexicon> <model\_out>"

exit 1

fi

if [ -f "$1" ]; then

lexicon=$(readlink -f $1)

model=$(readlink -f $2)

else

#else it's within the $data\_path

lexicon=$data\_path/$1

model=$data\_path/$2

fi

ln -s $dist\_path/path.sh path.sh

. path.sh

export LD\_LIBRARY\_PATH=$KALDI\_ROOT/tools/openfst/lib:/usr/local/lib

python2.7 $KALDI\_ROOT/tools/phonetisaurus-g2p/src/scripts/phonetisaurus-train --lexicon $lexicon --seq2\_del

mv train/model.fst "$model"

<https://github.com/speech-clarin-pl/SpeechToolsWorkers/blob/master/speech_tools/tools/SpeechActivityDetection/run.sh>

#!/bin/bash

set -e -o pipefail

dist\_path=/dist

tmp\_path=/tmp/work

data\_path=/data

nj=1

echo "$0 $@" # Print the command line for logging

. parse\_options.sh || exit 1;

if [ $# -ne 2 ]; then

echo "Usage: ./run.sh <input-wav> <out-ctm>"

echo ""

echo "Options:"

exit 1

fi

#if file is an existing global path

if [ -f "$1" ] ; then

wav\_file=$(readlink -f $1)

out=$(readlink -f $2)

else

#else it's within the $data\_path

wav\_file=$data\_path/$1

out=$data\_path/$2

fi

for f in $wav\_file; do

[ ! -f "$f" ] && echo "no such file $f" && exit 1;

done

if [ -e "$tmp\_path" ] ; then

rm -rf ${tmp\_path}

fi

mkdir -p ${tmp\_path}

mkdir ${tmp\_path}/data

tmp\_wav\_file=${tmp\_path}/$(basename $wav\_file).wav

sox $wav\_file -r8k $tmp\_wav\_file

echo input $tmp\_wav\_file > ${tmp\_path}/data/wav.scp

echo "input n/a" > ${tmp\_path}/data/text

echo input spk > ${tmp\_path}/data/utt2spk

echo spk input > ${tmp\_path}/data/spk2utt

cd ${tmp\_path}

ln -s ${dist\_path}/path.sh

ln -s ${dist\_path}/sad/conf

ln -s ${dist\_path}/sad/segmentation\_1a/tdnn\_stats\_asr\_sad\_1a model

. path.sh

ln -s $KALDI\_ROOT/egs/wsj/s5/utils

ln -s $KALDI\_ROOT/egs/wsj/s5/steps

./steps/segmentation/detect\_speech\_activity.sh --nj $nj --cmd run.pl --extra-left-context 79 --extra-right-context 21 \

--extra-left-context-initial 0 --extra-right-context-final 0 --frames-per-chunk 150 \

--mfcc-config conf/mfcc\_hires.conf \

data model mfcc work sad

awk '{printf "input 1 %0.3f %0.3f speech\n",$3,$4-$3}' < sad\_seg/segments > $out

echo Finished generating speech activity segmentation...