Digital Speech Processing
Homework #2-1

Automatic Speech Recognition of Mandarin Digits

張致強

October 28, 2020

Due Date: 23:59 November 20, 2020

Outline

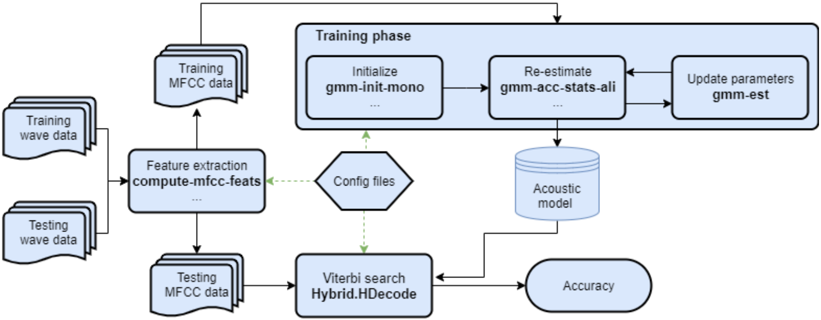
Introduction

Introduction

1. Construct a digit recognizer - monophone
    - lin | #i | #er | san | sy | #u | liou | qi | ba | jiou
2. Free tools of Kaldi ASR Toolkit:
    - https://kaldi-asr.org/
3. Training data, testing data, scripts, and other resources all are available on here[1]

[1] http://speech.ee.ntu.edu.tw/DSP2020Autumn/hw2/dsp_hw2-1.zip

Flowchart


Training phase
Training MFCC data
Training wave data
Testing wave data
Feature extraction
compute-mfcc-feats
...
Initialize
gmm-init-mono
...
Re-estimate
gmm-acc-stats-ali
...
Update parameters
gmm-est
Config files
Acoustic model
Testing MFCC data
Viterbi search
Hybrid.HDecode
Accuracy

Kaldi Speech Recognition Toolkit

What is Kaldi?

Kaldi is a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. Kaldi is intended for use by speech recognition researchers. For more detailed history and list of contributors see History of the Kaldi project.[2]

[2] https://kaldi-asr.org/doc/history.html

Setup Kaldi

Kaldi's code lives at *kaldi-asr/kaldi*.

Based on our experience, it's not easy to build the toolkit due to its dependencies. So we recommend you use the **pre-built Docker images**. And the following part will show you how to pull the image and run a container.

Use Pre-built Docker Image

Please follow these steps:

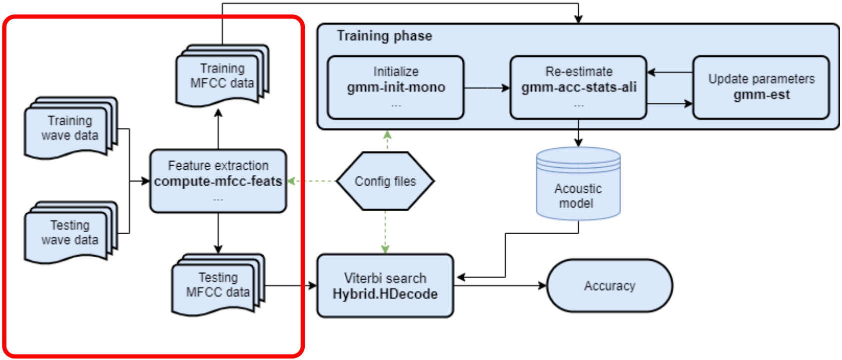
1. Install Docker on your system. Check *this* out for the installation of Docker.
2. Pull the image and run a container with,

```
docker run -it kaldiasr/kaldi:latest bash
```

For more details please refer to *base commands for the Docker CLI*.

Procedure

Feature Extraction 1/5



Feature Extraction:

```
compute-mfcc-feats scp:$1 ark,t,scp:$2,$3
```

Compute first 13 dimension of MFCC

Input:

- **$1** mapping from wav file to feature name

Output:

- **$2** 13 dimension MFCC of all files
- **$3** mapping from files to 13 dimension MFCC

Feature Extraction:

```
add-deltas ark:$2 ark:$4
```

Compute first and second derivative of MFCC

Input:

$2: 13 dimension MFCC of all files

Output:

$4: 39 dimension MFCC of all files

Feature Extraction:

```
compute-cmvn-stats ark:$4 ark:$5
```

Compute mean and variance of each dimension of MFCC

Input:

$4 39 dimension MFCC of all files

Output:

$5 mean and variance of each dimension of MFCC

Feature Extraction 5/5

Feature Extraction:

```
apply-cmvn ark:$5 ark:$4 ark,t,scp:$6, $7
```

Apply CMVN(Cepstral Mean and Variance Normalization)

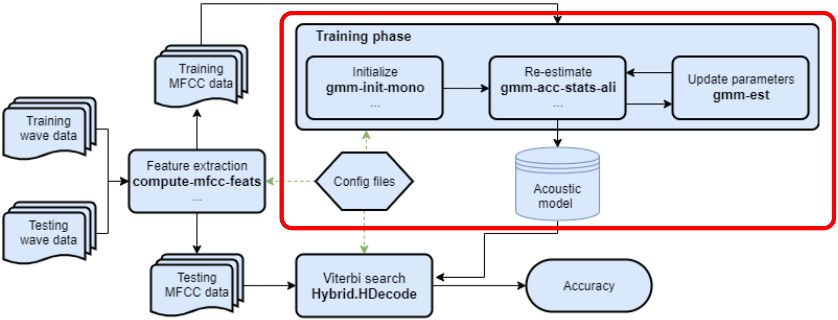
Input:

- **$5** mean and variance of each dimension of MFCC
- **$4** 39 dimension MFCC of all files

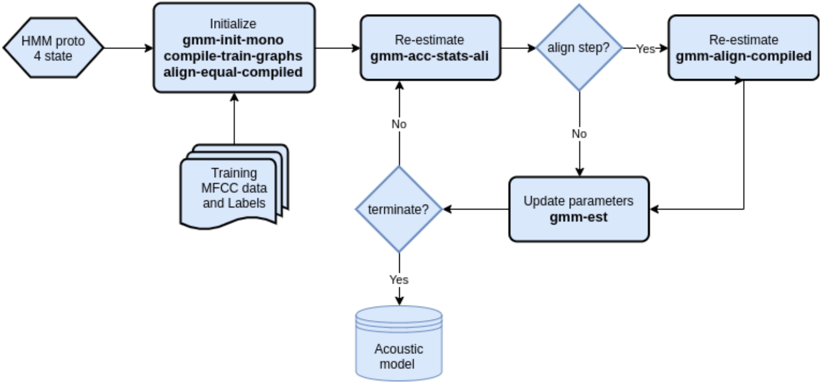
Output:

- **$6** 39 dimension CMVN MFCC of all files
- **$7** mapping from files to 39 dimension CMVN MFCC

Training Flowchart



Training Phase $^1/_7$



Initialize:

```
gmm-init-mono ...
```

Initialize monophone GMM model

Input:

- \$6 39 dimension CMVN MFCC of all files
- \$8 mapping each phone to corresponding index, ex:lin->0

Output:

- \$9 initialized GMM model
- \$10 training tree structure

Training Phase $^3/_7$

Initialize:

```
compile-train-graphs $10 $9 ark:$11 ark:$12
```

Compile the training FST(finite state transducer) graph

Input:

- $10 training tree structure
- $9 initialized GMM model
- $11 mapping from feature name to label

Output:

- $12 a FST(finite state transducer) of this task

Initialize:

```
align-equal-compiled ark:$12 ark,s,cs:$6 ark:$13
```

For each file, according to FST, generate align sequence.

Input:

- $12 a fst(finite state transducer) of this task
- $6 39 dimension CMVN MFCC of all files

Output:

- $13 align sequence for each file

Training Phase 5/7

Re-estimate:

```
gmm-acc-stats-ali $9 ark,s,cs:$6 ark:$13 $14
```

Accumulating GMM statistics

Input:

- \$9 initialized gmm model
- \$6 39 dimension CMVN MFCC of all files
- \$13 align sequence for each file

Output:

- \$14 accumulate of each file

Training Phase 6/7

Re-estimate:

```
gmm-align-compiled $9 ark:$12 ark,s,cs:$6 ark:$13
```

Aligning training graphs by GMM model

Input:

- **$9** GMM model of last step
- **$12** a FST(finite state transducer) of this task
- **$6** 39 dimension CMVN MFCC of all files

Output:

- **$13** new align sequence of each file

Training Phase 7/7

Update parameters:

```
gmm-est $9 $14 $15
```

Update GMM parameters and split to several gaussians

Input:

- **$9** GMM model of last step
- **$14** accumulate of each file

Output:

- **$15** updated GMM model

Requirements

Provided Files

`clean.sh`
- Clear all files produced by scripts

`0-activate.sh`
- Activate kaldi environment

`1-preprocess.sh`
- Preprocess files

`2-extract-feat.sh`
- Extract 39 dim MFCC of training and testing files

`3-train.sh`
- Train HMM model

`4-test.sh`
- Use Viterbi algorithm to get accuracy on testing data

`speechdata`
- Training and testing wav files

`material`
- Config and label data

`utility`
- Some utility scripts

```
dsp-hw2
├── clean.sh
├── 0-activate.sh
├── 1-preprocess.sh
├── 2-extract-feat.sh
├── 3-train.sh
├── 4-test.sh
├── speechdata
│   └── training
│   └── testing
├── material
│   └── ...
└── utility
    └── ...
```

Workflow

After setting up docker environment successfully

```
apt install libc6-dev-i386
wget http://speech.ee.ntu.edu.tw/DSP2020Autumn/hw2/dsp_hw2-1.zip
unzip dsp_hw2-1.zip
cd dsp-hw2-1
source 0-activate.sh
bash 1-preprocess.sh
bash 2-extract-feat.sh
bash 3-train.sh
bash 4-test.sh
```

And the output of `4-test.sh` will look like:

```
Converting acoustic models to HTK format
    output -> viterbi/mono/final.mmf viterbi/mono/tiedlist
    log -> viterbi/mono/log/am.to.htk.log
Generating results for test set with acoustic weight = [ 0.87 ]
    output -> viterbi/mono/test.mlf
    log -> viterbi/mono/log/latgen.test.log
    result -> viterbi/mono/test.rec
    accuracy -> [ 75.30 ] %

Execution time for whole script = 00 hours 00 mins 04 secs
```

Requirements

File Format

Report Format

Please write a `one-page` report in PDF format, name it `report.pdf` and submit with your source code.

State your name, student ID and any challenges you encounter or attempts you try. A good report may grant you bonus of extra 5%.

File Structure

Let's say you only have five scripts, your material directory, and report.pdf.

```
.
├── 0-activate.sh
├── 1-preprocess.sh
├── 2-extract-feat.sh
├── 3-train.sh
├── 4-test.sh
├── report.pdf
└── material
    └── [config and label files]
```

Requirements

Submission Requirement

Submission Requirement 1/2

1. Create a directory named `hw2-1_[STUDENT_ID]`.
2. Put
   - `0-activate.sh`
   - `1-preprocess.sh`
   - `2-extract-feat.sh`
   - `3-train.sh`
   - `4-test.sh`
   - `report.pdf`
   - `material/`

   into the directory.
3. Compress the directory into a **ZIP** file named `hw2_1_[STUDENT_ID].zip`.
4. Upload this ZIP file to CEIBA.

Submission Requirement $^2/_2$

Let's say your student ID is `r01234567`.

```
hw2_1_r01234567.zip
└── hw2-1_r01234567
    ├── 0-activate.sh
    ├── 1-preprocess.sh
    ├── 2-extract-feat.sh
    ├── 3-train.sh
    ├── 4-test.sh
    ├── report.pdf
    └── material
        └── [config and label files]
```

Grading

Grading Method

TA will use the docker image mentioned in page 5 to run your scripts.

For each of you, your scripts are allowed to run for 5 mins in total.

Grading Policy

Accuracy[3] 80%

- 40% for simple Baseline 75.40%
- 40% for strong Baseline 95.00%
- 5% bonus of extra points for outperforming 99.00%

Report 20%

And bonus of extra 5% for the impressive ones

[3] Get full credit for outperforming strong baseline, otherwise get partial credit.

Late Submission

You are still allowed to submit after the due date. The penalty for late submission is an exponential decay with decay rate 1.5%[4] of the maximum grade applicable for the assignment, for each hour that the assignment is late.

An assignment submitted more than 3 days after the deadline will have a grade of zero recorded for that assignment.

$$\text{SCORE}_{final}(hr) = \begin{cases} \text{SCORE}_{original} \times 0.985^{hr} & , \; hr \leq 72 \\ 0 & , \; hr > 72 \end{cases}$$

[4] less than 70% after 24 hrs, 48% for 48 hrs and 33% for 72 hrs

DO NOT CHEAT

Any form of cheating, lying, or plagiarism will not be tolerated.

Contact TAs

Should you have any question or need help,

- please read the FAQ[5] first.
- send email to *ntu-dsp-2020-ta@googlegroups.com*
- and use "**`[HW2-1]`**" as the subject line prefix

Or come to EE2 R531, and don't forget to inform us by email, thanks!

張致強	Mon.	13:30 - 17:30
	Fri.	9:00 - 12:00

Office hours

[5] http://speech.ee.ntu.edu.tw/DSP2020Autumn/hw2/FAQ.html

Appendix

Submission Suggestion

You may find it difficult to get files from your docker container.

For this, TA suggest that you can use git.
And here[6] is a tutorial for git.

Then you can download your files from GitHub and upload to CEIBA.

[6] Start a new git repository

Useful Command

After building docker container by `docker run ...` successfully.

If you want to attach container you built.
You can use `docker ps --all` to show containers on your device.

Then use `docker start -a [CONTAINER ID]` to attach it again.

Useful Tips

You can take a look into the following files

- `3-train.sh`
- `4-test.sh`
- `material/topo.proto`

There are some tips that may help you with the assignment.