Text-to-speech experiment on ESPnet
Lab 8

[Objective]

1. Learn how to do speech synthesis with the use of ESPnet

[Task description]

ESPnet (https://github.com/espnet/espnet) is an end-to-end speech processing toolkit, mainly focuses on end-to-end speech recognition and end-to-end text-to-speech (TTS). If you want to run a full experiment, including the training, please follow the instructions at the original web to have a full installation.
In this lab, you only need to do the speech synthesis (testing) using the pre-trained models. There are two models for you to examine. One of them is from ESPnet1 and the other one is from ESPnet2. Please examine both models and determine the better one. These models are trained with the CSMSC Mandarin dataset.

[ESPnet1]

1) Installation

Install the necessary package with the following commands:

```
pip install parallel_wavegan PyYaml unidecode ConfigArgparse g2p_en
pip install espnet_tts_frontend
git clone -q https://github.com/espnet/espnet.git
cd espnet && git fetch && git checkout -b v.0.9.1 refs/tags/v.0.9.1
pip install pypinyin
pip install chainer
```

Here, please note that Pytorch is automatically installed when you install "parallel_wavegan".

2) Copy the pre-trained models

Copy the folder "downloads" from /home/cseadmin/TTSdemo/downloads to your local folder. The Mandarin models are in the folder.

3) Script preparation

Copy the following codes to a python file.

zhuym@sustc.edu.cn

```
# set path
dict_path = "downloads/zh/fastspeech/data/lang_phn/train_no_dev_units.txt"
model_path =
"downloads/zh/fastspeech/exp/train_no_dev_pytorch_train_fastspeech.v3.single
/results/model.last1.avg.best"
vocoder_path =
"downloads/zh/fastspeech/csmsc.parallel_wavegan.v1/checkpoint-
400000steps.pkl"

# add path
import sys
sys.path.append("espnet")

# define device
import torch
device = torch.device("cpu")

# define E2E-TTS model
from argparse import Namespace
from espnet.asr.asr_utils import get_model_conf
from espnet.asr.asr_utils import torch_load
from espnet.utils.dynamic_import import dynamic_import
idim, odim, train_args = get_model_conf(model_path)
model_class = dynamic_import(train_args.model_module)
model = model_class(idim, odim, train_args)
torch_load(model_path, model)
model = model.eval().to(device)
inference_args = Namespace(**{"threshold": 0.5, "minlenratio": 0.0,
"maxlenratio": 10.0})

# define neural vocoder
from parallel_wavegan.utils import load_model
fs = 24000
vocoder = load_model(vocoder_path)
vocoder.remove_weight_norm()
vocoder = vocoder.eval().to(device)

# define text frontend
```

```
from pypinyin import pinyin, Style
from pypinyin.style._utils import get_initials, get_finals
with open(dict_path) as f:
    lines = f.readlines()
lines = [line.replace("\n", "").split(" ") for line in lines]
char_to_id = {c: int(i) for c, i in lines}
def frontend(text):
    """Clean text and then convert to id sequence."""
    text = pinyin(text, style=Style.TONE3)
    text = [c[0] for c in text]
    print(f"Cleaned text: {text}")
    idseq = []
    for x in text:
        c_init = get_initials(x, strict=True)
        c_final = get_finals(x, strict=True)
        for c in [c_init, c_final]:
            if len(c) == 0:
                continue
            c = c.replace("ü", "v")
            c = c.replace("ui", "uei")
            c = c.replace("un", "uen")
            c = c.replace("iu", "iou")
            # Special rule: "e5n" -> "en5"
            if "5" in c:
                c = c.replace("5", "") + "5"
            if c not in char_to_id.keys():
                print(f"WARN: {c} is not included in dict.")
                idseq += [char_to_id["<unk>"]]
            else:
                idseq += [char_to_id[c]]
    idseq += [idim - 1]  # <eos>
    return torch.LongTensor(idseq).view(-1).to(device)

print("now ready to synthesize!")

import time
print("请用中文输入您喜欢的句子!")
input_text = input()
```

```
with torch.no_grad():
    start = time.time()
    x = frontend(input_text)
    c, _, _ = model.inference(x, inference_args)
    y = vocoder.inference(c)
rtf = (time.time() - start) / (len(y) / fs)
print(f"RTF = {rtf:5f}")

data = y.view(-1).cpu().numpy()

# write the generated audio stream to a .wav file

import numpy as np
from scipy.io.wavfile import write

scaled = np.int16(data/np.max(np.abs(data)) * 32767)
write('test.wav', fs, scaled)
```

4) Running the script

Run the above python code and generate the speech signal. The speech should be saved to the “test.wav” file.

5) Check the performance

How do you think about the performance of the TTS models? You are suggested to examine this TTS demonstration and find some words which it pronounce incorrectly.

[ESPnet2]

1) Installation

Install the necessary package with the following commands:

```
pip install parallel_wavegan==0.4.8 espnet_model_zoo
git clone -q https://github.com/espnet/espnet.git
cd espnet && git fetch && git checkout -b v.0.9.7
```

2) Copy the pre-trained models

There are 3 .npz data files required by the models. The paths of those files are specified in "downloads/tts_train_fastspeech2_raw_phn_pypinyin_g2p_phone/config.yaml". Currently, they point to the files located in the admin's folder. Since the files are accessible, you can skip this modification for convenience.

3) Script preparation

Copy the following codes to a python file.

```
# Set paths of pre-trained models
# "downloads" should consist with the path storing the pre-trained models
am_path = "downloads/tts_train_fastspeech2_raw_phn_pypinyin_g2p_phone"
vocoder_path = "downloads/csmsc.parallel_wavegan.v1/checkpoint-
400000steps.pkl"

import sys
sys.path.append("espnet") # "espnet" is the path of the repository you just
pulled

import os
import time
import torch
from espnet2.bin.tts_inference import Text2Speech
from parallel_wavegan.utils import load_model

# Load the acoustic model, FastSpeech2
device = "cpu"
text2speech = Text2Speech(
    train_config=os.path.join(am_path, "config.yaml"),
    model_file=os.path.join(am_path, "train.loss.ave_5best.pth"),
    device=device,
    speed_control_alpha=1.0,
)
text2speech.spc2wav = None

# Load the vocoder
```

```
fs = 24000
vocoder = load_model(vocoder_path).to(device).eval()
vocoder.remove_weight_norm()

# Decide the input sentence by yourself
print(f"请用中文输入你喜欢的句子：")
x = input()

# Synthesis
with torch.no_grad():
  start = time.time()
  wav, c, *_ = text2speech(x)
  wav = vocoder.inference(c)
rtf = (time.time() - start) / (len(wav) / fs)
print(f"RTF = {rtf:5f}")

# Save the synthesized speech
import numpy as np
from scipy.io.wavfile import write

wav = wav.view(-1).cpu().numpy()
pad_len = int(0.15 * fs)
wav = np.pad(wav, (pad_len, pad_len), 'constant', constant_values=(0, 0))
scaled = np.int16(wav/np.max(np.abs(wav)) * 32767)
write('test.wav', fs, scaled)
```

4) Running the script

Run the above python code and generate speech signals. The speech should be saved to the “test.wav” file.

5) Compare the performance

Compare the speech quality of the two models and determine the better one.