Text2Speech

This notebook provide a demonstration of the realtime E2E-TTS using ESPnet-TTS and ParallelWaveGAN (+ MelGAN)

- ESPnet: https://github.com/espnet/espnet
- ParallelWaveGAN: https://github.com/kan-bayashi/ParallelWaveGAN

Author: Hesam Tavakoli (1nj3ct0r)

Install

```
# Install minimal components

!pip install -q parallel_wavegan PyYaml unidecode ConfigArgparse g2p_en 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
```

Download Pretrained Feature Generation Model

You can select one from three models. Please only run the selected model cells.

(A) Tacotron 2

```
# Download pretrained model
import os
if not os.path.exists("downloads/en/tacotron2"):
    !./espnet/utils/download_from_google_drive.sh \
        https://drive.google.com/open?id=1lFfeyewyOsxaNO-DEWy9iSz6qB9ZS1UR downloads/en/tacot

# Set path
trans_type = "phn"
dict_path = "downloads/en/tacotron2/data/lang_1phn/phn_train_no_dev_units.txt"
model_path = "downloads/en/tacotron2/exp/phn_train_no_dev_pytorch_train_pytorch_tacotron2.v3/

print("Sucessfully Finished Download...")
```

(B) Transformer

```
# Download pretrained model
import os
if not os.path.exists("downloads/en/transformer"):
    !./espnet/utils/download_from_google_drive.sh \
        https://drive.google.com/open?id=1z8KSOWVBjK-_Ws4RxVN4NTx-Buy03-7c downloads/en/trans

# Set path
trans_type = "phn"
dict_path = "downloads/en/transformer/data/lang_1phn/phn_train_no_dev_units.txt"
model_path = "downloads/en/transformer/exp/phn_train_no_dev_pytorch_train_pytorch_transformer

print("Sucessfully Finished Download...")
```

(C) Fast Speech

```
# Download pretrained model
import os
if not os.path.exists("downloads/en/fastspeech"):
    !./espnet/utils/download_from_google_drive.sh \
        https://drive.google.com/open?id=1P9I4qag8wAcJiTCPawt6WCKBqUfJFtFp downloads/en/fasts

# Set path
trans_type = "phn"
dict_path = "downloads/en/fastspeech/data/lang_1phn/phn_train_no_dev_units.txt"
model_path = "downloads/en/fastspeech/exp/phn_train_no_dev_pytorch_train_tacotron2.v3_fastspe

print("Sucessfully Finished Download...")
```

Download Prentrained Vocoder Model

You can select one from two models. Please only run the selected model cells.

(A) Parallel WaveGAN

```
# Download pretrained model
import os
if not os.path.exists("downloads/en/parallel_wavegan"):
    !./espnet/utils/download_from_google_drive.sh \
        https://drive.google.com/open?id=1Grn7X9wD35UcDJ5F7chwdTqTa4U7DeVB downloads/en/paral

# Set path
vocoder_path = "downloads/en/parallel_wavegan/ljspeech.parallel_wavegan.v2/checkpoint-400000s

print("Sucessfully Finished Download...")
```

(B) MelGAN

```
# Download pretrained model
import os
if not os.path.exists("downloads/en/melgan"):
    !./espnet/utils/download_from_google_drive.sh \
        https://drive.google.com/open?id=1_a8faVA5OGCzIcJNw4blQYjfG4oA9VEt downloads/en/melga

# Set path
vocoder_path = "downloads/en/melgan/train_nodev_ljspeech_melgan.v3.long/checkpoint-4000000ste

print("Sucessfully Finished Download...")
```

(C) Multi-Band MelGAN

This is an **Experimental** model.

```
# Download pretrained model
import os
if not os.path.exists("downloads/en/mb-melgan"):
    !./espnet/utils/download_from_google_drive.sh \
        https://drive.google.com/open?id=1rGG5y15uy4WZ-lJy8NPVTkmB_6VhC20V downloads/en/mb-me

# Set path
vocoder_path = "downloads/en/mb-melgan/train_nodev_ljspeech_multi_band_melgan.v1/checkpoint-1

print("Sucessfully Finished Download...")
```

Setup

```
# Add path
import sys
sys.path.append("espnet")

# Define device
import torch
device = torch.device("cuda")

# 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_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,
    # Only for Tacotron 2
    "use_attention_constraint": True, "backward_window": 1,"forward_window":3,
    # Only for fastspeech (lower than 1.0 is faster speech, higher than 1.0 is slower speech)
    "fastspeech_alpha": 1.0,
})

# Define neural vocoder
from parallel_wavegan.utils import load_model
fs = 22050
vocoder = load_model(vocoder_path)
vocoder.remove_weight_norm()
vocoder = vocoder.eval().to(device)

# Define text frontend
from tacotron_cleaner.cleaners import custom_english_cleaners
from g2p_en import G2p
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}
g2p = G2p()
def frontend(text):
    """Clean text and then convert to id sequence."""
    text = custom_english_cleaners(text)

    if trans_type == "phn":
        text = filter(lambda s: s != " ", g2p(text))
        text = " ".join(text)
        print(f"Cleaned Text: {text}")
        charseq = text.split(" ")
    else:
        print(f"Cleaned Text: {text}")
        charseq = list(text)
    idseq = []
    for c in charseq:
        if c.isspace():
            idseq += [char_to_id["<space>"]]
        elif c not in char_to_id.keys():
            idseq += [char_to_id["<unk>"]]
        else:
            idseq += [char_to_id[c]]
    idseq += [idim - 1]  # <eos>
    return torch.LongTensor(idseq).view(-1).to(device)

import nltk
nltk.download('punkt')
```

```
print("Now Ready to Synthesize!")
```

Synthesis

```
import time
print("Input your favorite sentence in English:")
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}")

from IPython.display import display, Audio
display(Audio(y.view(-1).cpu().numpy(), rate=fs))
```