CTTS 可控语音合成 使用说明书

项目介绍

这是 Microsoft 的工作：**FastSpeech 2: Fast and High-Quality End-to-End Text to Speech**. 的 Pytorch 实现。

此项目基于 xcmyz's implementation 的 FastSpeech 实现. 代码实现基于 https://github.com/ming024/FastSpeech2. 更多细节可在原仓库中找到。

快速开始

Gradio 交互界面

通过以下指令启动 GUI 图形界面

```
python .\gui.py
```

安装依赖

通过以下指令安装项目所需的 Python 库。

```
pip3 install -r requirements.txt
```

注意：gradio 所需要安装的 pandas 库和项目模型所需的 Numpy 版本不同，请在安装所有依赖后，回滚 Numpy版本。

推理

如果您已经阅读过原仓库的代码实现，您可发现当前版本已经简化了推理所需的命令行参数。目前代码可通过指定一个数据集名，自动索引到所需的所有配置文件。为了使这项简化正常工作，您需要按以下的结构组织文件：

```
.
├── config
│   ├── MODEL_NAME
└── └── └── model.yaml
        └── preprocess.yaml
        └── train.yaml
```

以 ESD_en 数据集为例，对于英语说话人的语音合成推理，运行以下命令：

```
python .\synthesize.py -t "YOUR_CONTENT" -m ESD_en
```

命令支持其他的可选参数，请使用 help 功能查看详细信息：

```
python .\synthesize.py --help
```

Here lists some common used parameters:

`-s` or `--speaker_id`: specify the emotion id in multi emotion datasets.

`-e` or `--emotion_id`: specify the speaker id in multi speaker datasets.

`-r` or `--restore_step`: load the model of a particular checkpoint.

The generated utterances will be put in `output/result/`.

训练

数据集

支持的数据集如下：

- LJSpeech: a single-speaker English dataset consists of 13100 short audio clips of a female speaker reading passages from 7 non-fiction books, approximately 24 hours in total.
- AISHELL-3: a Mandarin TTS dataset with 218 male and female speakers, roughly 85 hours in total.
- LibriTTS: a multi-speaker English dataset containing 585 hours of speech by 2456 speakers.
- ESD: ESD is an Emotional Speech Database for voice conversion research. The ESD database consists of 350 parallel utterances spoken by 10 native English and 10 native Chinese speakers and covers 5 emotion categories (neutral, happy, angry, sad and surprise). More than 29 hours of speech data were recorded in a controlled acoustic environment. The database is suitable for multi-speaker and cross-lingual emotional voice conversion studies.

数据预处理

首先，运行：

```
python3 prepare_align.py config/AISHELL3/preprocess.yaml
```

以进行必要的预处理准备，随后使用MFA工具对齐文字与音频（详见下面MFA的更多细节）。之后，运行下面的命令执行数据预处理：

```
python3 preprocess.py config/LJSpeech/preprocess.yaml
```

MFA 的更多细节

~~Alternately, you can align the corpus by yourself.~~
~~Download the official MFA package and run~~

```
./montreal-forced-aligner/bin/mfa_align raw_data/LJSpeech/ lexicon/librispeech-
lexicon.txt english preprocessed_data/LJSpeech
```

~~or~~

```
./montreal-forced-aligner/bin/mfa_train_and_align raw_data/LJSpeech/
lexicon/librispeech-lexicon.txt preprocessed_data/LJSpeech
```

~~to align the corpus and then run the preprocessing script.~~

```
python3 preprocess.py config/LJSpeech/preprocess.yaml
```

Updated by CTTS: the commands above provided by origin repo cannot work in most cases, especially when the coder is not familiar with MFA tools, you can get more details in the Issues of the origin repo.

The following is the baselien of aligning your raw data with MFA, Attention: dictionarys and acoustic models provided my MFA can work well in languages like ENG (As far as I am concerned), but official dictionaries fail in aligning mandarins, especially in this repository, because the preprocess of this repository is based on pinyin in Mandarin. Therefore, used customized dictionary and dictionary to make sure it works well with Mandarin. Sources can be accessible in the Google Drive URL below:

Acoustic Model: Download In Google Drive

Dictionary: Download in Google Drive

> ### How to align Mandarin raw data (e.g. AISHELL3, ESD) with MFA?
>
> 1. Install MFA tools, use the command "mfa --help" to check if the environment is set correctly.
> 2. Install Mandarin pretrained acoustic model, the models can be found at: https://mfa-models.readthedocs.io/en/latest/acoustic/index.html
> 3. Get data prepared for alignment
> 4. Make sure a lexicon file is ready, like the "pinyin-lexicon-r.txt" in this project.
> 5. Use the command **"mfa align .\raw_data_path .\lexicon_path ACOUSTIC_MODEL .\target_path"** to start alignment.

> *In this project (ESD mandarin), the speaker "0005" is removed for unknown errors it would cause in MFA alignment"*

Attention: The following processes haven't been modified for ESD support yet!

Training

Train your model with

```
python3 train.py -m MODEL_NAME
```

Implementation Issues

- Following xcmyz's implementation, I use an additional Tacotron-2-styled Post-Net after the decoder, which is not used in the original FastSpeech 2.
- Gradient clipping is used in the training.
- In my experience, using phoneme-level pitch and energy prediction instead of frame-level prediction results in much better prosody, and normalizing the pitch and energy features also helps. Please refer to `config/README.md` for more details.

Please inform me if you find any mistakes in this repo, or any useful tips to train the FastSpeech 2 model.

References

- FastSpeech 2: Fast and High-Quality End-to-End Text to Speech, Y. Ren, *et al*.
- xcmyz's FastSpeech implementation
- TensorSpeech's FastSpeech 2 implementation
- rishikksh20's FastSpeech 2 implementation

Citation

```
@INPROCEEDINGS{chien2021investigating,
  author={Chien, Chung-Ming and Lin, Jheng-Hao and Huang, Chien-yu and Hsu, Po-
chun and Lee, Hung-yi},
  booktitle={ICASSP 2021 - 2021 IEEE International Conference on Acoustics,
Speech and Signal Processing (ICASSP)},
  title={Investigating on Incorporating Pretrained and Learnable Speaker
Representations for Multi-Speaker Multi-Style Text-to-Speech},
  year={2021},
  volume={},
  number={},
  pages={8588-8592},
  doi={10.1109/ICASSP39728.2021.9413880}}
```