**Tutorial**

❑ 본 튜토리얼은 리눅스 환경에서 Docker를 사용한 방식으로 설명되어 있습니다.

- Docker download : https://www.docker.com/
(Docker image는 AI-Hub에서 다운받을 수 있습니다.)

❑ 시스템 사양

- CPU 32 core 이상
- Mem 244GB 이상
- Ubuntu 18.04
- CUDA 11.1
- Docker 20.10.2
- GPU Tesla-A100 x 1

❑ Docker 사용 방법

* 독일어(de), 영어(en), 스페인어(es), 프랑스어(fr), 일어(jp), 러시아어(ru), 중국어(zh) 에 동일하게 적용가능합니다.

* 발음 평가 학습과 추론은 각 언어 동일하게 모두 /data/project/nia/pron 폴더에서 수행합니다.

* 미리 학습한 모델이 docker에 포함되어 있어 학습 준비 및 학습 과정인 3-4 과정을 생략하고 5-6번 과정을 통해 음성 파일에 대해 추론을 수행할 수 있습니다.

0. 다운받은 데이터는 /data1/nia_db/ 아래에 설치되었다고 가정합니다.

1. 다운받은 Docker image(nia_pron.tar.gz)를 load 합니다.
   - 첨부된 nia_pron.tar.gz 파일에서 아래와 같이 이미지를 로딩합니다.

```
$ docker load -i nia_pron.tar.gz
```

2. Container를 실행합니다.

-v/data1/nia_db:/data/project/rw/nia_db/nia_2022_012 : 다운받은 데이터가 포함된 "/data1/nia_db" 폴더를 docker container 안의 "/data/project/rw/nia_db/nia_2022_012" 폴더와 공유합니다. (학습에 필요한 data는 AI-Hub 사이트에서 다운로드 가능합니다. )

```
$ docker run --rm -it -v /data1/nia_db:/data/project/nia_db/nia_2022_012 --gpus all --shm-size 10gb nia_pron
```

3. 데이터 세트를 훈련, 검증, 실험 세트로 8:1:1 분리합니다.

훈련을 위해서 data_prepare 폴더 안의 lang_{de|en|es|fr|jp|ru|zh} 폴더 아래에 아래와 같이 trn/dev/test list파일을 생성합니다.

```
* fluency_trn.list
* fluency_val.list
* fluency_test.list
* pron_trn.list
* pron_val.list
* pron_test.list
```

생성된 list파일은 다음과 같은 구조를 갖습니다.

```
*.list : [wav file] [articulation score] [prosody score] [script]
```

- 미리 학습된 모델에 사용된 데이터 리스트가 data_prepare 폴더에 포함되어 있습니다.

* 참고
    * 미리 만들어져 있는 리스트가 아니라 AI_HUB에서 DB를 다운로드 받아 학습 리스트를 다시 생성하려고 할 때는 첨부한 data_prepare_sample를 참조하여 step1 ~ step5를 실행해 data 파일을 다시 만들어 줍니다.

    step1 : 데이터 다운로드
        - 작업서버의 /data1/nia_db 라는 디렉토리에 jsonFor12.tar.gz 라는 압축파일을 다운로드 받았다고 가정

    step2 : 압축해제
        - tar xvfz jsonFor12.tar.gz

    step3 : 첨부한 data_prepare_sample 을 /data1/nia_db/ 아래에 copy

    step4 : json file list 생성
        - cd /data1/nia_db/data_prepare_sample
        - find /data1/nia_db/jsonFor12 -name "*.json" > nia_012_json.list

  step5 : 언어별로 train:val:test = 8:1:1 로 리스트 생성
        - sh mk_data.sh nia_012_json.list

4. 발음 평가 모델 훈련

- 발음 평가 모델 훈련은 각 언어에 대해 미리 학습된 wav2vec2 모델을 통과하는 특징 벡터 추출 과정과 추출된 특징 벡터를 바탕으로 발음 평가 모델을 학습하는 과정으로 구성되어 있습니다.

- --lang 옵션을 통하여 언어를 선택하면 hugging face에서 각 언어에서 미리 학습된 wav2vec2 모델을 내려 받습니다.

- 학습에 사용할 wav 파일들을 내려받은 wav2vec2 모델에 통과하여 특징 벡터를 모두 추출한 후 발음 평가 학습이 시작됩니다.

- 훈련은 validation 데이터를 기준으로 pcc(pearson correlation coeifficient)를 계산한 후 pcc가 patience 갯수의 epoch만큼 증가하지 않으면 종료합니다.

- 훈련이 완료되면 validation 데이터 기준 가장 높은 pcc 결과의 모델이 dir_model/lang_{de|en|es|fr|jp|ru|zh} 폴더에 {label_type1}_{label_type2}_checkpoint.pt 라는 이름으로 저장됩니다.

run_tr.sh 참조..

```
#!/bin/bash
DATA_DIR=/data/project/rw/nia_db/nia_2022_012/
python train_mlp.py --lang='de' --label_type1='pron' --label_type2='articulation' --dir_data=$DATA_DIR/jsonFor12/39/audio --dir_list=$DATA_DIR/data_prepare_sample --mlp_hidden=64 --epochs=200 --patience=20 --batch_size=256 --dir_model='model' --audio_len_max=300000
```

--lang : 언어 선택 (de-독일어, en-영어, es-스페인어, fr-프랑스어, jp-일어, ru-러시아어, zh-중국어)
--label_type1 : 학습할 발음평가 라벨 선택(1) ('pron' / 'fluency')
--label_type2 : 학습할 발음평가 라벨 선택(2) ('articulation' / 'prosody')
--dir_data : 학습할 발음평가 데이터가 위치한 폴더 경로
--dir_list : 과정 3에서 준비한 trn/val/test list가 있는 폴더 경로
--dir_model : inference를 수행할 모델 경로
--mlp_hidden : 모델의 hidden units 개수
--epochs : finetune 훈련 횟수
--patience : 성능이 더이상 오르지 않으면 학습을 조기 종료하기 위한 파라미터
--batch_size : 훈련 batch 크기

5. wave 파일 이용한 추론

- model/lang_{언어}의 모델을 사용하여 단일 wave 파일에 대한 추론을 다음과 같이 수행합니다.

- 단일 파일에 대한 발음 평가 점수를 프린트 합니다.

```
$ python inference_wav.py --wav=dir_wav --lang='de' --label_type1='pron' --label_type2='articulation' --device='cpu' --dir_model='model'
```

--wav : wav 파일 경로
--lang : 언어 선택 (de-독일어, en-영어, es-스페인어, fr-프랑스어, jp-일어, ru-러시아어, zh-중국어)
--label_type1 : 평가할 발음평가 라벨 선택(1) ('pron' / 'fluency')
--label_type2 : 평가할 발음평가 라벨 선택(2) ('articulation' / 'prosody')
--device : inference를 수행할 device 선택 ('cuda' / 'cpu')
--dir_model : inference를 수행할 모델 경로

6. 리스트를 이용한 추론

- model/lang_{언어}의 모델을 사용하여 과정 3에서 생성한 파일 리스트에 대한 추론을 다음과 같이 수행합니다.

- 결과는 result_{de|en|es|fr|jp|ru|zh}_{label_type1}_{}abel_type2l_{data_type}.txt 에 저장됩니다.

- 결과 파일에는 각 파일에 대한 발음 평가 점수와 해당 리스트에 대한 pcc 점수가 포함되어 있습니다.

run_test.sh 참조

```
#!/bin/bash
DATA_DIR=/data/project/rw/nia_db/nia_2022_012/
python inference.py --lang='de' --label_type1='pron' --label_type2='articulation' --dir_list=$DATA_DIR/data_prepare_sample --device='cuda' --dir_data=$DATA_DIR/jsonFor12/39/audio
~
```

--lang : 언어 선택 (de-독일어, en-영어, es-스페인어, fr-프랑스어, jp-일어, ru-러시아어, zh-중국어)
--label_type1 : 평가할 발음평가 라벨 선택(1) ('pron' / 'fluency')
--label_type2 : 평가할 발음평가 라벨 선택(2) ('articulation' / 'prosody')
--dir_list : 과정 3에서 준비한 리스트가 위치한 폴더 경로
--device : inference를 수행할 device 선택 ('cuda' / 'cpu')
--dir_data : 학습할 발음평가 데이터가 위치한 폴더 경로
--data_type : 평가할 데이터 타입 (trn / val / test)
--dir_model : inference를 수행할 모델 경로

7. 기존 학습된 모델로부터 발음 평가 모델 훈련

- 4번 학습 과정에서 --dir_resume 옵션을 통해 이미 학습된 모델로 초기화 한 후 새로운 데이터로 모델을 fine-tuning 할 수 있습니다.

```
$ export DATA_DIR=/data/project/rw/nia_db/nia_2022_012
$ python train_mlp.py --lang='de' --label_type1='pron' --label_type2='articulation' --dir_data=$DATA_DIR/jsonFor12/39/audio --dir_list='data/data_prepare' --mlp_hidden=64 --epochs=200 --patience=20 --batch_size=256 --dir_model='model' --dir_resume='model_1cycle'
```

--lang : 언어 선택 (de-독일어, en-영어, es-스페인어, fr-프랑스어, jp-일어, ru-러시아어, zh-중국어)
--label_type1 : 학습할 발음평가 라벨 선택(1) ('pron' / 'fluency')
--label_type2 : 학습할 발음평가 라벨 선택(2) ('articulation' / 'prosody')
--dir_data : 학습할 발음평가 데이터가 위치한 폴더 경로
--dir_list : 과정 3에서 준비한 trn/val/test list가 있는 폴더 경로
--dir_model : inference를 수행할 모델 경로
--mlp_hidden : 모델의 hidden units 개수
--epochs : finetune 훈련 횟수
--patience : 성능이 더이상 오르지 않으면 학습을 조기 종료하기 위한 파라미터
--batch_size : 훈련 batch 크기