중노년층 방언데이터

[1] AI모델 환경 설치가이드

1. AI모델 환경용 서버 준비

권장 서버

파이토치를 지원하는 GPU가 장착된 서버

참고 : 아래 개발에 사용된 시스템 사양 이상 권고

```
----- System Info -----
===== CPU Info :
CPU Model : AMD Ryzen Threadripper PRO 5975WX 32-Cores
CPU MHz : 1800.000
CPU max MHz : 7006.6401
CPU min MHz : 1800.0000
Number of Sockets : 1
Core(s) per socket : 32
Thread(s) per core : 2
Total Number of cores(socket x core x thread) : 64
On-line CPU(s) list : 0-63

===== Memory Info :
Memory Total : 251Gi

===== GPU Info :
GPU 0 - model: NVIDIA RTX A6000, gpu memory: 47.54 GB
GPU 1 - model: NVIDIA RTX A6000, gpu memory: 47.54 GB
GPU 2 - model: NVIDIA RTX A6000, gpu memory: 47.54 GB
GPU 3 - model: NVIDIA RTX A6000, gpu memory: 47.54 GB
```

GPU 드라이버 및 CUDA Version:

NVIDIA-SMI 510.108.03 Driver Version: 510.108.03 CUDA Version: 11.6

운영체제 : Ubuntu 20.04

2. 도커 환경 구축

AI 모델 학습을 하기 위한 환경을 도커 컨테이너를 이용하여 구축한다. 이를 위해서는 우선 도커 소프트웨어를 설치하여야 한다. 그리고 구축하는 방법은 도커 컨테이너 이미지를 수동으로 생성하여 설치하는 방법과 제공된 도커 컨테이너 이미지 파일을 임포트하여 사용하는 방법이 있다. 도커 컨테이너 이미지 파일이 있는 경우 이미지 파일을 임포트하여 사용하는 것이 편리하며 이미지 파일이 없는 경우에는 수동으로 생성하여 사용하면 된다.

사용 도커 컨테이너 소프트웨어 버전 : Docker 20.10

도커 컨테이너는 아래 URL을 참고하여 설치할 수 있다.

https://docs.docker.com/engine/install/ubuntu/

도커 버전 확인 명령

```
$ docker --version
```

2-1. 도커 컨테이너 이미지 파일을 임포트하여 사용하기

도커 컨테이너 이미지 파일 임포트

다음과 같은 명령으로 도커 컨테이너 이미지를 임포트하여 컨테이너 이미지를 생성한다.

```
$ docker import kdialect-speech-container.tar kdialect-speech:202212
```

여기서 이미지 이름(kdialect-speech:202212)은 임의로 정할 수 있으며 다음 단계의 이미지 이름과 같아야 한다.

그리고 이 컨테이너 이미지를 이용하여 다음과 같이 컨테이너를 실행한다.

```
$ docker run -itd \
    --name kdialect-speech \
    --gpus all \
    --ipc=host \
    -p 6006:6006 \
    -p 8888:8888 \
    -v /etc/localtime:/etc/localtime:ro \
    -v <your-data-dir>:/data \
    kdialect-speech:202212 &
```

여기서 `-v <your-data-dir>:/data`은 데이터가 있는 디렉토리를 컨테이너에서 사용할 수 있게 볼륨을 마운트하는 것이므로 중노년층 방언데이터를 저장하고 있는 디렉토리가 되어야 한다. 그러므로 <your-data-dir>을 학습용 데이터가 저장된 디렉토리 위치와 같게 수정한다.

생성된 컨테이너 확인

```
$ docker ps
```

다음 명령으로 컨테이너에 로그인

```
$ docker exec -it kdialect-speech bash
```

이미지 파일을 임포트하여 사용하는 경우 필요한 speechbrain toolkit이 설치되어 있으므로 별도의 추가 설치 없이 사용하면 된다.

2-2. 도커 컨테이너 수동 생성하여 사용하기

도커 설치 및 컨테이너 생성

도커 컨테이너 생성

다음의 명령을 실행하여 도커 컨테이너를 생성

```
$ docker run -itd \
    --name kdialect-speech \
    --gpus all \
    --ipc=host \
    -p 6006:6006 \
    -p 8888:8888 \
    -v /etc/localtime:/etc/localtime:ro \
    -v <your-data-dir>:/data \
    nvcr.io/nvidia/pytorch:22.01-py3 &
```

여기서 `-v <your-data-dir>:/data`은 데이터가 있는 디렉토리를 컨테이너에서 사용할 수 있게 볼륨을 마운트하는 것이므로 중노년층 방언데이터를 저장하고 있는 디렉토리가 되어야 한다. 그러므로 <your-data-dir>을 학습용 데이터가 저장된 디렉토리 위치와 같게 수정한다.

생성된 컨테이너 확인

```
$ docker ps
```

다음 명령으로 컨테이너에 로그인

```
$ docker exec -it kdialect-speech bash
```

음성인식 학습용 소프트웨어 준비

도커 컨테이너를 수동 생성하여 사용하는 경우 컨테이너에 로그인 한 후 다음과 같은 작업으로 모델 환경 설치한다.

소스 다운로드

```
# git clone https://github.com/starcell/KdialectSpeech.git
```

speechbrain 설치

중노년층 방언 데이터 음성인식 모델은 speechbrain AI toolkit을 사용하므로 speechbrain을 설치해야 한다.

KdialectSpeech 소스에는 중노년층 방언 데이터 음성인식 모델링을 위해 일부 수정된 내용들을 포함하는 speechbrain 소스가 있으므로 이를 이용하여 speechbrain을 설치한다.

(speechbrain에 대한 자세한 내용은 공식 홈페이지 참고 : https://speechbrain.github.io)

```
#cd KdialectSpeech
#apt update
#apt upgrade
#apt install ffmpeg
#pip install -r requirements.txt
#pip install editable .
```

설치 확인

다음과 같이 python REPL에서 speechbrain import 테스트를 해본다.

오류 메시지 없이 넘어가면 성공적으로 speechbrain이 설치된 것이다.

```
# python
Python 3.8.12 | packaged by conda-forge | (default, Oct 12 2021, 21:59:51)
```

```
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import speechbrain
```

[2] AI모델 실행 매뉴얼

음성인식 모델 학습

Speechbrain toolkit이 정상적으로 설치되고 난 후, 중노년층 방언 데이터 음성인식 모델을 실행하기 위해서는 파이프라인 실행 파일을 다음과 같이 사용한다.

기본 베이스 데렉토리는 /workspace/KdialectSpeech 이며 사용자의 환경에 맞게 수정하여 사용할 수 있다.

```
# cd recipes/KdialectSpeech/Pipeline
# python run-pipe.py run-pipe.yaml
```

위의 명령은 run-pipe.yaml에 지정된 값에 따라 모델 학습을 실행한다.

run-pipe.yaml에 지정된 값의 주요 내용은 아래와 같다.

```
run_modules : 실행할 모듈 선택
   - tokenizer
   - lm
   - asr
```

run_modules에서 주석처리된 모듈은 실행되지 않는다. tokenizer를 먼저 실행하여 데이터 전처리를 수행하고 토큰화 모델을 만들어야 음성인식 모델을 학습할 수 있다. lm(language modle)은 언어모델 학습을 실행할 것인지 설정하며, 학습된 언어모델은 음성인식의 전사 추론에서 선택적으로 사용할 수 있다. 기본(default)적으로 lm은 사용하지 않는다. 이 두 작업은 한 번에 연속하여 수행되어도 되며, 하나씩 별개로 수행되어도 된다.

```
run_provinces: 모델 학습을 수행할 지역 선택
   - gw
   - gs
   - jl
   - jj
   - cc
   - total : 전체 지역, 베이스 모델 학습에 사용할 수 있음
```

방언의 코드는 각각 다음과 같다.

```
gw : 강원
gs : 경상
jl : 전라
jj : 제주
cc : 충청
```

run_provinces에서 주석처리된 지역 방언은 처리되지 않는다. total은 모든 지역을 통합하여 처리하므로 베이스라인 모델 학습에 사용할 수 있다.

run_modules와 run_provinces는 서로 조합하여 선택 가능하다. 즉 아래와 같이 주석처리하여 설정하면 모든 지역의 데이터들을 통합하여 토큰화 작업이 실행된다.

```
run_modules:
    - tokenizer
    # - asr

run_provinces:
    # - gw
    # - gs
    # - jl
    # - jj
    # - cc
    - total
```

다음과 같이 설정하면 강원도(gw) 지역에 대하여 토큰화 작업과 음성인식 모델 학습이 연속하여 수행된다.

```
run_modules:
    - tokenizer
    - asr

run_provinces:
    - gw
    # - gs
    # - jl
    # - jj
    # - cc
    # - total
```

학습 결과 확인

학습 모델 결과 디렉토리

/workspace/KdialectSpeech/recipes/KdialectSpeech/ASR/Conformer/results/Conformer/7774

여기서 7774는 seed값으로 설정에 따라 달라질 수 있다.

이 디렉토리에는 방언별로5개의 하위 디렉토리(gw, gs, jl, jj, cc)가 있으며, 각 디렉토리에서 로그 파일을 통하여 모델 학습 결과를 확인할 수 있다.

train_log.txt에서 학습로그를 확인할 수 있다.

(로그 파일 내용 예시)

```
epoch: 98, lr: 2.50e-05, steps: -1 - train loss: 18.57 - valid loss: 21.07, valid ACC: 8.21e-01
epoch: 99, lr: 2.50e-05, steps: -1 - train loss: 18.36 - valid loss: 20.37, valid ACC: 8.21e-01
epoch: 100, lr: 2.50e-05, steps: -1 - train loss: 17.86 - valid loss: 19.89, valid ACC: 8.23e-01
Epoch loaded: 100 - test loss: 25.80, test ACC: 7.87e-01, test sWER: 30.15, test CER: 14.26
```

swer_test.txt에서 테스트 데이터에 대하여 전사한 결과를 볼 수 있다.

(결과 파일 내용 예시)

```
================================================================================
gw-say_set1_collectorgw133_speakergw758_58_0_32-0, %sWER 4.88 [ 2 / 41, 0 ins, 0 del, 2 sub ]
감기가 ; 걸리면은 ; 옛날에는 ; 약이 ; 없어서 ; 귀했을 ; 때는 ; 어머니들이 ; 다들 ; 생강을 ; 썰어서 ; 팔팔 ; 끓인 ; 다음에 ; 꿀을 ; 넣어서 ; 주시기도 ; 했고 ;
 =  ;  =   ;  =   ; =  ;  =  ;  =  ; =  ;  =   ; =  ;  =  ;  =  ;  =  ; =  ;  =  ;  =  ;  =  ;  =  ;  =  ;  =  ;  =  ;  =  ;  =  ; =  ; =  ;  S  ; =  ;  =  ;  =  ; =  ;
감기가 ; 걸리면은 ; 옛날에는 ; 약이 ; 없어서 ; 귀했을 ; 때는 ; 어머니들이 ; 다들 ; 생강을 ; 썰어서 ; 팔팔 ; 끓인 ; 다음에 ; 꿀을 ; 넣어서 ; 주시기도 ; 했고 ;
================================================================================
```

학습된 모델 파일을 이용한 추론

다음과 같이 학습된 모델을 이용하여 음성 파일을 문자 파일(text)로 변환하는 추론을 수행할 수 있다.

학습모델 추론 디렉토리로 이동

```
# cd /workspace/KdialectSpeech/recipes/KdialectSpeech/ASR/Conformer/Inference
```

음성 파일을 입력으로 모델 추론 프로그램 실행

```
# python inference-EncoderDecoderASR.py <음성파일>
```

별첨 : KdialectSpeech 설정 파일

<run_pipe.yaml>

run_pipe.py 실행 시 전달

```yaml
### 0. common
# pipe_log
log_config: log-config.yaml # 로그 파일 설정 파일
log_file: log.txt # run_pipe.py 실행로그를 기록할 파일 지정
error_file_log: error_files.txt # wrong audio files
wrong_samplerate_file: wrong_samplerate.txt # samplerate 이 16000 이 아닌 파일 목록

kdialect_base_dir: ..
tokenizer_dir: !ref <kdialect_base_dir>/Tokenizer
lm_dir: !ref <kdialect_base_dir>/LM
asr_dir: !ref <kdialect_base_dir>/ASR/Conformer

# run modules, 아래 지정된 모듈 들을 순차적으로 실행
run_modules:
    - tokenizer
    # - lm
    - asr

run_provinces: # 아래 지정된 지역 방언에 대하여 순차적으로 실행(gw:강원, gs:경상, jl:전라, jj:제주, cc:충청)
    # - gw
    # - gs
    # - jl
    - jj
    # - cc
    # - total

gpu_num: 4 # 사용할 gpu 수, 시스템의 가용 GPU 수 보다 크면 안됨.

# 결과 모델 파일을 trained model dir 로 복사 여부
copy_trained_model: True
pretrained_model_base: !ref <asr_dir>/Inference/pretrained-model-src # 학습된 모델을 사용하기 위해 복사할 디렉토리

smaplerate: 16000

### 1. Tokenizer, 지역 별 어휘의 수 설정, 제주만 어휘 수가 적어서 1000 으로 설정
```

```
tokenizer_dict: # provice_code : token_output: 5000
   gw: 5000
   gs: 5000
   jl: 5000
   jj: 1000
   cc: 5000

### 2. LM(언어모델)

### 3. ASR(음성인식)
```

음성인식(ASR)모델 학습에 사용한 초매개변수 값

음성인식 모델의 초매개변수들(hyperparameters)은

KdialectSpeech/recipes/KdialectSpeech/ASR/Conformer/hparams/conformer_medium.yaml 파일에 설정된다. KdialectSpeech에서 사용된 주요 초매개변수 설정값은 다음과 같다.

초매개변수	기본 설정값	설명
number_of_epochs	100	전체 데이터에 대한 학습 반복 횟수,
batch_size	32	
ctc_weight	0.3	
sorting	ascending	
lr_adam	0.001	
lr_sgd	0.000025	
sample_rate	16000	
n_fft	400	
n_mels	80	