

nemo_asr-pyctc-v1

June 19, 2022

```
[2]: import nemo
import nemo.collections.asr as nemo_asr
from pyctcdecode import build_ctcdecoder

from jiwer import wer ## libarry to compute WER
from statistics import mean ## Used to compute mean WER
import pandas as pd #for IO

## change this base_path pointing to location of task_Data
base_path = '/home/manju/Desktop/assign/task_data/'

def readDatasetsForDecoding(csv_file):
    data = pd.read_csv(csv_file)
    wav_loc = data['path']
    ground_truth = data['transcription']
    #action = data['action']
    #object_category = data['object']
    #location = data['location']
    return wav_loc, ground_truth

def getAbsoluteWavPath(wav_loc):
    ##Add base path to wav_loc list to get absolute path
    files = []
    for fname in wav_loc:
        files.append(base_path + fname)
    #### files=files[0:2]
    return files

def decodeAndComputeWER(inputFileName, NemoModelName, outputFileName):
    print("*** Decoding wav_data of ", inputFileName, " and writing results to_
↵", outputFileName, "*****")

    ## Reading the csv files
    wav_loc, ground_truth = readDatasetsForDecoding(inputFileName)
```

```

files = getAbsoluteWavPath(wav_loc)

index_cnt = 0
wer_lst = []
###files=files[0:10]

file_wr = open(outputFileName, 'w')
#print(files)
out = 'Wave-File-Name , ' + 'hypothesised-Transcription , ' +
↳'GroundTruth-Transcription' + 'WER' + '\n'
file_wr.write(out)
## Transcribe validation data using NEMO model
for fname, hypothesis in zip(files, NemoModelName.
↳transcribe(paths2audio_files=files)):
    #print(f"{fname},\t \t \"{decoded_transption}\",\t \t \t
↳\"{ground_truth[index_cnt]}\")
    error = wer(ground_truth[index_cnt], hypothesis) ### compute WER
    wer_lst.append(error)
    ## append to write output
    out = '\"' + fname + '\", \"' + hypothesis + '\", \"' +
↳ground_truth[index_cnt] + '\", \"' + str(error) + '\"\\n'
    ## Write to file
    file_wr.write(out)
    index_cnt += 1 ##increase the indent

print("Results of decoding written to ", outputFileName, " file.")
## Compute Overall WER
overall_WER = mean(wer_lst) * 100
print("Overall WER is: ", round(overall_WER, 3), "%")
print("Total number of files decoded is: ", index_cnt)

out = '\"Overall WER is: ' + str(round(overall_WER, 3)) + "%, " + '\"Total_
↳number of files : ' + str(index_cnt) + '\\n'
file_wr.write(out)
file_wr.close() ##close the file

def pyctc_decodeAndComputeWER(inputFileName, NemoModelName, outputFileName):
    print("*** Decoding wav_data of ", inputFileName, " and writing results to_
↳", outputFileName, "*****")

    ## Reading the csv files
    wav_loc, ground_truth = readDatasetsForDecoding(inputFileName)
    files = getAbsoluteWavPath(wav_loc)

```

```

index_cnt = 0
wer_lst = []
###files=files[0:10]

file_wr = open(outputFileName, 'w')
#print(files)
out = 'Wave-File-Name , ' + 'hypothesised-Transcription , ' +
↳ 'GroundTruth-Transcription' + 'WER' + '\n'
file_wr.write(out)
## Transcribe validation data using NEMO model

for fname in files:
    logits = NemoModelName.transcribe([fname], logprobs=True)[0]
    decoder = build_ctcdecoder(NemoModelName.decoder.vocabulary)
    hypothesis = decoder.decode(logits)
    error = wer(ground_truth[index_cnt], hypothesis) ### compute WER
    wer_lst.append(error)
    ## append to write output
    out = '\"' + fname + '\", \"' + hypothesis + '\", \"' +
↳ ground_truth[index_cnt] + '\", \"' + str(error) + '\"\\n'
    ## Write to file
    file_wr.write(out)
    index_cnt += 1 ##increase the indent

print("Results of decoding written to ", outputFileName, " file.")
## Compute Overall WER
overall_WER = mean(wer_lst) * 100
print("Overall WER is: ", round(overall_WER, 3), "%")
print("Total number of files decoded is: ", index_cnt)

out = '\"Overall WER is: ' + str(round(overall_WER, 3)) + "%, " + '\"Total_
↳ number of files : ' + str(index_cnt) + '\\n'
file_wr.write(out)
file_wr.close() ##close the file

## Read the data and put into different lists
def main(base_path):

    train_file = base_path + 'train_data.csv'
    valid_file = base_path + 'valid_data.csv'

    # Download various variants of Nemo models
    ## refer https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/
↳ asr/results.html#english

```

```

jasper_model = nemo_asr.models.EncDecCTCModel.
↳from_pretrained(model_name="stt_en_jasper10x5dr")
quartznet_model = nemo_asr.models.EncDecCTCModel.
↳from_pretrained(model_name="QuartzNet15x5Base-En")
conformer_ctc_md1 = nemo_asr.models.EncDecCTCModelBPE.
↳from_pretrained(model_name="stt_en_conformer_ctc_large")

## conf_trandcr_model = nemo_asr.models.EncDecRNNTBPEModel.
↳from_pretrained(model_name="stt_en_conformer_transducer_xlarge")
## asr_model = nemo_asr.models.EncDecCTCModelBPE.
↳from_pretrained(model_name="stt_en_conformer_ctc_xlarge")
## print(nemo_asr.models.EncDecCTCModelBPE.list_available_models())

## Derive hypothesis and compute WER
##for NemoModelName in jasper_model, quartznet_model:

outputFileName = "pyctc_conformer_ctc_validation_results.txt"
pyctc_decodeAndComputeWER(valid_file, conformer_ctc_md1, outputFileName)

outputFileName = "conformer_ctc_validation_results.txt"
decodeAndComputeWER(valid_file, conformer_ctc_md1, outputFileName)

outputFileName = "jasper_validation_results.txt"
decodeAndComputeWER(valid_file, jasper_model, outputFileName)

outputFileName = "quartznet_validation_results.txt"
decodeAndComputeWER(valid_file, quartznet_model, outputFileName)

print("\n***** For reference : architecture of Jasper nemo model_
↳***** \n")
print(jasper_model)

print("\n***** For reference : architecture of quartznet nemo_
↳model ***** \n")
print(quartznet_model)

if __name__ == "__main__":
    main(base_path)

```

[NeMo I 2022-06-19 18:48:09 cloud:56] Found existing object /home/manju/.cache/torch/NeMo/NeMo_1.0.0/stt_en_jasper10x5dr/856ae08d5c4bd78b5e27f696e96f7aab/stt_en_jasper10x5dr.nemo.

[NeMo I 2022-06-19 18:48:09 cloud:62] Re-using file from: /home/manju/.cache/tor

ch/NeMo/NeMo_1.0.0/stt_en_jasper10x5dr/856ae08d5c4bd78b5e27f696e96f7aab/stt_en_jasper10x5dr.nemo

[NeMo I 2022-06-19 18:48:09 common:675] Instantiating model from pre-trained checkpoint

[NeMo W 2022-06-19 18:48:17 modelPT:137] If you intend to do training or fine-tuning, please call the ModelPT.setup_training_data() method and provide a valid configuration file to setup the train data loader.

Train config :

manifest_filepath: /data2/voices/train_1k.json

sample_rate: 16000

labels:

- ' '

- a

- b

- c

- d

- e

- f

- g

- h

- i

- j

- k

- l

- m

- 'n'

- o

- p

- q

- r

- s

- t

- u

- v

- w

- x

- 'y'

- z

- ' '' '' ''

batch_size: 32

trim_silence: true

max_duration: 16.7

shuffle: true

is_tarred: false

tarred_audio_filepaths: null

[NeMo W 2022-06-19 18:48:17 modelPT:144] If you intend to do validation, please

call the `ModelPT.setup_validation_data()` or `ModelPT.setup_multiple_validation_data()` method and provide a valid configuration file to setup the validation data loader(s).

```
Validation config :
manifest_filepath: /data2/voices/train_1k_samp.json
sample_rate: 16000
labels:
- ' '
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- 'n'
- o
- p
- q
- r
- s
- t
- u
- v
- w
- x
- 'y'
- z
- ''''
batch_size: 32
shuffle: false
```

[NeMo I 2022-06-19 18:48:17 features:252] PADDING: 16

[NeMo I 2022-06-19 18:48:17 features:269] STFT using torch

[NeMo W 2022-06-19 18:48:17 nemo_logging:349]

/home/manju/anaconda3/lib/python3.9/site-packages/nemo/collections/asr/parts/features.py:302: FutureWarning: Pass sr=16000, n_fft=512 as keyword args. From version 0.10 passing these as positional arguments will result in an error

```
librosa.filters.mel(sample_rate, self.n_fft, n_mels=nfilt, fmin=lowfreq,
fmax=highfreq), dtype=torch.float
```

```

[NeMo I 2022-06-19 18:48:23 modelPT:434] Model EncDecCTCModel was successfully
restored from /home/manju/.cache/torch/NeMo/NeMo_1.0.0/stt_en_jasper10x5dr/856ae
08d5c4bd78b5e27f696e96f7aab/stt_en_jasper10x5dr.nemo.
[NeMo I 2022-06-19 18:48:23 cloud:56] Found existing object
/home/manju/.cache/torch/NeMo/NeMo_1.0.0/QuartzNet15x5Base-
En/2b066be39e9294d7100fb176ec817722/QuartzNet15x5Base-En.nemo.
[NeMo I 2022-06-19 18:48:23 cloud:62] Re-using file from:
/home/manju/.cache/torch/NeMo/NeMo_1.0.0/QuartzNet15x5Base-
En/2b066be39e9294d7100fb176ec817722/QuartzNet15x5Base-En.nemo
[NeMo I 2022-06-19 18:48:23 common:675] Instantiating model from pre-trained
checkpoint
[NeMo I 2022-06-19 18:48:24 features:252] PADDING: 16
[NeMo I 2022-06-19 18:48:24 features:269] STFT using torch
[NeMo I 2022-06-19 18:48:24 modelPT:434] Model EncDecCTCModel was successfully
restored from /home/manju/.cache/torch/NeMo/NeMo_1.0.0/QuartzNet15x5Base-
En/2b066be39e9294d7100fb176ec817722/QuartzNet15x5Base-En.nemo.
[NeMo I 2022-06-19 18:48:24 cloud:56] Found existing object /home/manju/.cache/t
orch/NeMo/NeMo_1.0.0/stt_en_conformer_ctc_large/b2da9ca277cfb30bb0c6b2abd44eef02
/stt_en_conformer_ctc_large.nemo.
[NeMo I 2022-06-19 18:48:24 cloud:62] Re-using file from: /home/manju/.cache/tor
ch/NeMo/NeMo_1.0.0/stt_en_conformer_ctc_large/b2da9ca277cfb30bb0c6b2abd44eef02/s
tt_en_conformer_ctc_large.nemo
[NeMo I 2022-06-19 18:48:24 common:675] Instantiating model from pre-trained
checkpoint
[NeMo I 2022-06-19 18:48:27 mixins:147] Tokenizer SentencePieceTokenizer
initialized with 128 tokens

[NeMo W 2022-06-19 18:48:27 modelPT:137] If you intend to do training or fine-
tuning, please call the ModelPT.setup_training_data() method and provide a valid
configuration file to setup the train data loader.
    Train config :
    manifest_filepath:
/data/nemo_asr_set/asr_set_1.4/train_no_appen/tarred_audio_manifest.json
    sample_rate: 16000
    batch_size: 8
    shuffle: true
    num_workers: 8
    pin_memory: true
    use_start_end_token: false
    trim_silence: false
    max_duration: 20.0
    min_duration: 0.1
    shuffle_n: 2048
    is_tarred: true
    tarred_audio_filepaths:
/data/nemo_asr_set/asr_set_1.4/train_no_appen/audio__OP_0..2047_CL_.tar

```

[NeMo W 2022-06-19 18:48:27 modelPT:144] If you intend to do validation, please call the ModelPT.setup_validation_data() or ModelPT.setup_multiple_validation_data() method and provide a valid configuration file to setup the validation data loader(s).

```
Validation config :
manifest_filepath:
- /manifests/librispeech/librivox-dev-other.json
- /manifests/librispeech/librivox-dev-clean.json
- /manifests/librispeech/librivox-test-other.json
- /manifests/librispeech/librivox-test-clean.json
sample_rate: 16000
batch_size: 8
shuffle: false
num_workers: 8
pin_memory: true
use_start_end_token: false
is_tarred: false
tarred_audio_filepaths: ''
```

[NeMo W 2022-06-19 18:48:27 modelPT:151] Please call the ModelPT.setup_test_data() or ModelPT.setup_multiple_test_data() method and provide a valid configuration file to setup the test data loader(s).

```
Test config :
manifest_filepath:
- /manifests/librispeech/librivox-dev-other.json
- /manifests/librispeech/librivox-dev-clean.json
- /manifests/librispeech/librivox-test-other.json
- /manifests/librispeech/librivox-test-clean.json
sample_rate: 16000
batch_size: 8
shuffle: false
num_workers: 8
pin_memory: true
use_start_end_token: false
is_tarred: false
tarred_audio_filepaths: ''
```

[NeMo I 2022-06-19 18:48:27 features:252] PADDING: 0

[NeMo I 2022-06-19 18:48:27 features:269] STFT using torch

[NeMo I 2022-06-19 18:48:29 modelPT:434] Model EncDecCTCModelBPE was successfully restored from /home/manju/.cache/torch/NeMo/NeMo_1.0.0/stt_en_conformer_ctc_large/b2da9ca277cfb30bb0c6b2abd44eef02/stt_en_conformer_ctc_large.nemo.
*** Decoding wav_data of /home/manju/Desktop/assign/task_data/valid_data.csv and writing results to pyctc_conformer_ctc_validation_results.txt *****

Transcribing: 0%| | 0/1 [00:00<?, ?it/s]

[NeMo W 2022-06-19 18:48:29 nemo_logging:349]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]
Transcribing: 0%|          | 0/1 [00:00<?, ?it/s]

```

Results of decoding written to pyctc_conformer_ctc_validation_results.txt file.

Overall WER is: 41.769 %

Total number of files decoded is: 3118

```
[3]: print()
```

```
[ ]:
```