Automatic Speech Recognition with Nvidia NeMo --- tutorial & project

joepareti54@gmail.com 27. Juli 2021 updated on 7. Aug. 2021 and on 14. Aug. 2021

Tutorial code on GPU, stage 0 **2**
Tutorial code on CPU **2**
Apply NeMo and the tutorial code to the Listen Attend Spell (LAS) Project, stage 1 **2**
Implementation 3
github repo 3
Implementation details on manifest file 3
Create input by segmenting out from global file 4
output checkpoint file 4
input checkpoint file 4
Results on August 7, 2021 4
Results on August 13, 2021 4
Continuing the training while using the checkpoint model 5
Conclusions **5**

Nvidia NeMo looks promising to unlock me on a speech recognition and translation project, currently placed on hold due to lack of support for critical software components.

Therefore I decided to run an experiment, a tutorial with this code.

Stage 0: make the code run on my PC/ubuntu
Stage 1: modify the code to train the NeMo model using my VCTK data that are described here

As of today, stage 0 runs to completion, but only when the cpu is used; when the gpu is used only the initial training is successful, while the cell *Model Improvement - Transfer Learning* ends abnormally with a 'gpu-out-of-memory ' exception.

Before investing more effort, I need to understand if this is just a hardware limitation. I could not identify any resources specifications: gpu is a critical component for any serious work in stage 1.

The Jupyter Notebook with the code changes that runs on a cpu, and the output is available here.

Here is a summary published on linkedin.

Tutorial code on GPU, stage 0

The hardware is the same as reported here. The only important code modification to make the tutorial run on a GPU is the NeMo upgrade :

```
!pip install --upgrade nemo_toolkit['all']
```

The other issue, *out-of-gpu-memory*, occurs on the cell *Model Improvement - Transfer Learning.*

CUDA out of memory. Tried to allocate 18.00 MiB (GPU 0; 5.80 GiB total capacity; 4.29 GiB already allocated; 8.44 MiB free; 4.45 GiB reserved in total by PyTorch)

Here is a link to the notebook and its output when run on the GPU.

Tutorial code on CPU

The Jupyter Notebook with the code changes that runs on a cpu, and the output is available here.

I also had a 'dead kernel' issue that automagically went away.

Apply NeMo and the tutorial code to the Listen Attend Spell (LAS) Project, *stage 1*

The VCTK corpus for this project is certainly a class above the small dataset used in the tutorial. Moreover, the data format is different, and it remains to be seen if and how NeMo can be applied. I understand that the yaml configuration file and the manifest have capabilities to adapt to different formats: the question is how, and hence I have asked in the NeMo forum, and in an Nvidia forum. Since there was no answer, I decided to go ahead with my implementation.

Another critical question is about resources; the LAS project can only be done on GPUs, and hence I must know if my PC is adequate, or what cloud configuration must be acquired. This will be a critical factor when aiming at a specific accuracy that would then expand the training effort.

The training even in a basic form and on a GPU platform takes a long time, and hence checkpointing is an important feature, and the good news is that NeMo supports it according to the information provided in the tutorial. The next paragraph explains checkpointing in greater detail.

Implementation

In order to use the NeMo notebook for this project, 2 modifications were necessary:

1. create a line in the manifest file in the required format out of the provided .wav file and corresponding .txt file, for each pair in the VCTK corpus
2. include code to read the model from an input checkpoint file and write the updated model to an output checkpoint file. The file names contain information on start/end training line, number of epochs and validation lines, the latter are taken from a variable line till EOF.
3. The initial implementation uses a coarse learning rate = 0.01 ?

github repo

This is the github repo for this project.

Implementation details on manifest file

```
import os
import librosa
path  =  '/media/joepareti54/Elements/x/finance-2020/AI/Listen_attend_spell/VCTK-Corpus/wav48'
path_t=  '/media/joepareti54/Elements/x/finance-2020/AI/Listen_attend_spell/VCTK-Corpus/txt'
dirct= os.listdir(path)
outDict = {"audio_filepath": ' ', 'duration': 0, 'text': 'abc'}
#
fout =
open("/media/joepareti54/Elements/x/finance-2020/AI/Listen_attend_spell/VCTK-Corpus/out_file.txt", "w")
#
#
for SUBDIR in dirct:
        PATH = path +'/'+ SUBDIR
        FILES = os.listdir(PATH)
        for FILE in FILES:
                x = FILE.split(".")
                FILE_NAME = path_t+'/'+SUBDIR+'/'+x[0]+'.txt'
                FILE_NAMEw= path+'/'+SUBDIR+'/'+x[0]+'.wav'
                f = open(FILE_NAME, 'r')
                St= f.read()
                duration = librosa.core.get_duration(filename=FILE_NAMEw)
                f.close()
                outDict={"audio_filepath": FILE_NAMEw, "duration": duration, "text" : St}
                json.dump(outDict, fout)
                fout.write('\n')
fout.close()
```

Create input by segmenting out from global file

```
START_LINE = 20000
END_LINE = 40000
NUMB_OF_LINES = END_LINE - START_LINE
NUMB_OF_VALID_LINES = 1000
MAX_EPOCHS = 100
OUT_DIR = '/media/joepareti54/Elements/x/finance-2020/AI/Listen_attend_spell/VCTK-Corpus/'
OUT_FILE=OUT_DIR+'out_file.txt'
TRAIN_MANIFEST=OUT_DIR+'manifest.json'
TEST_MANIFEST=OUT_DIR+'t-manifest.json'
!head -$END_LINE $OUT_FILE | tail -$NUMB_OF_LINES  > $TRAIN_MANIFEST
!tail -1000  $OUT_FILE > $TEST_MANIFEST
```

output checkpoint file

```
CHECKPOINT_DIR = OUT_DIR + 'CHECKPOINT/'
#
CHECKPOINT_FILE_OUT = CHECKPOINT_DIR + 'train-' + str(START_LINE) + '-' +
str(END_LINE)+'-valid-End-'+str(NUMB_OF_VALID_LINES)+'-'+str(MAX_EPOCHS)+'e.chk'
#
#print('CHECKPOINT_FILE_IN', CHECKPOINT_FILE_IN)
print('CHECKPOINT_FILE_OUT', CHECKPOINT_FILE_OUT)
```

input checkpoint file

```
import glob
list_of_files = glob.glob(CHECKPOINT_DIR +'*') # * means all if need specific format then *.csv
CHECKPOINT_FILE_IN = max(list_of_files, key=os.path.getctime)
#
print('input checkpoint file ',CHECKPOINT_FILE_IN)
```

Results on August 7, 2021

I have used up to 40000 lines out of the 44000 of the VCTK corpus, and trained the NeMo model on 100 epochs from line 8000 to 40000, and on 50 epochs for the initial 8000 lines.

The results show some accuracy improvement, by comparison of the 20k and 40k lines checkpoints, but are still far from production quality.

Reducing the learning rate could help.

The latest run on 20000 lines and 100 epochs took 4 solid days on my GPU laptop.

Results on August 13, 2021

Best accuracy using Quartznet as is. See this report.

Continuing the training while using the checkpoint model

The checkpointing code has been changed from the beginning of the project and subject to debugging a new approach has been determined.

Conclusions

1. **I achieved the best results using QuartzNet pre-trained model even though it never saw my VCTK corpus dataset. To understand why it is the way it is, some expert advice was necessary.** Here it is:
   The QuartzNet model was trained on nearly 5000 hours of speech, roughly 2.5 Million audio files, from various public corpora. Fine Tuning on a specific domain will usually help that domain only and degrade generalization performance on other domains unless the corpus is as large as or even larger than the original dataset
2. doing inferences on quartznet is a simple task: only a few lines of code do the job: so anybody can do it, I am not sure where is the competitive advantage of integrating a VCTK corpus with a public domain model, if the latter is already capable of transcribing almost any spoken sentence.
3. because of the results explained at point #2, the VCTK corpus does not add any value to the project besides testing and inferencing
4. The next step is machine translation from English to Spanish, that remains to be done. NeMo does support machine translation, yet it is not clear if there is any value in doing so.
5. A project proposal has been submitted to kickstarter: the budget figures are at this stage guesswork.