Assignment Report

Objective of the problem is given an “audio” file, we need to do ASR & NLP and derive it’s transcription, and corresponding action, object, and location.

This assignment has 2 problems to be solved:
1. ASR to derive the text from the given audio
2. NLP to find intent of the transcribed text w.r.to action, object, and location fields

Part I : ASR:

Note: *I did not had personal GPU to train the ASR models. But, training any state-of-the-art pytorch based models would need GPU. I explored using Google colab for GPU training, but since Google colab environment is not persistent (everytime I will have setup env by installing required libraries etc.), and does not provide command terminal to execute shell commands, it was very painful to use it for training. I had even trained some espnet reciepes on Google colab, but frequent disruptions prompted me to recreate environment multiple times. After struggling for half-day, I dropped the idea of using Google colab and paper-space cloud for GPU. I had even requested Sarathi team to provide GPU on cloud (in case, if they can). So, finally decided work without GPU.*

Nvidia Nemo ASR:

I have used the pre-trained models of Nvidia Nemo and decoded the wave files of validation set using 3 Nvidia Nemo models, namely – Quartznet, Jasper, Conformer-CTC-large. These models are built on top of pytorch & pytorch lighting, but does not GPU for decoding.
The WER’s obtained are as below:
Quartznet – 50.412%
Jasper - 47.805%
Conformer-CTC-large - 40.456 %

Based on the info available at “https://github.com/kensho-technologies/pyctcdecode”, I further explored **pyctcdecode** approach for decoding, on top of Conformer-CTC-large, and obtained 41.769 % WER. Further, building an unigram or bigram language model (LM) using only the transcription data is expected to give much better improvements. This LM improves WER by restricting the vocabulary to only to the words in LM. The bi-gram LM will be helpful in capturing the context. *This is very novel idea, and this is very much expected to reduce the WERs significantly. I really wanted to try this, but due to lack of time, I could not do this now.*

The Nvidia Nemo models work well only with 16 KHz sampling rate audio files. Since, the given audio files were 8 Khz, they had to be upsampled to 16 KHz using sox utility. *It is mentioned in NVIDIA website that the performance of the manually up-sampled audio files is expected to degrade*. Instead of 8 KHz sampling rate, if the wave files were of 16 KHz sampling rate then WERs would have been even lower.

Further, I have also developed a template code using which one can train ASR models either from scratch using NVIDIA Nemo libraries, and also adapt pre-trained NVIDIA models by doing transfer learning on custom data. However, this needs a GPU. So, I was unable to execute it by myself. Similar to NVIDIA Nemo models, I have also explored the use of facebook’s Wav2Vec2 framework (using hugging face transformers and based on pytorch) for decoding.

Part II : NLP for intent classification

The SVM classifers are trained to classify the intent of action, object, & location fields. The word embeddings for the transcriptions of training data are generated using Spacy NLP model. Totally, three SVM classifiers are trained, one for each action, object, & location categories. The corresponding values in action, object, & location fields are used as labels for training. The validation tests are used for SVM validation and testing. It is found that all the three SVM’s have shown a accuracy of 100% on test sets. This is very good performace, and it might be because the datasets were very small & had only few number of classes. In addition, Named Entity Recognition is tried out on Validation set, to identify nouns, verbs, entities in the given transcription.

Assignment submission folder structure:

```
(base) manju@dinesh-pc:~/Desktop/assign/sarati_sub$ tree
.
├── asr_results
│   ├── conformer_ctc_validation_results.txt
│   ├── jasper_validation_results.txt
│   ├── pyctcdecoded_conformer_ctc_validation_results.txt
│   └── quartznet_validation_results.txt
├── ipynb
│   ├── action-intent-classify.ipynb
│   ├── hugging_face_pretrained_usage.ipynb
│   ├── location-intent-classify.ipynb
│   ├── nemo_asr-models-execution.ipynb
│   ├── nemo_asr-pyctcdecode-execution.ipynb
│   ├── nemo_asr-Transfer-learning-template.ipynb
│   ├── NLP-NER-validation.ipynb
│   └── object-intent-classify.ipynb
│   └── espnet_training_gpu_google_colab.ipynb
├── ipynb_executed_pdf
│   ├── action-intent-classify.pdf
│   ├── location-intent-classify.pdf
│   ├── nemo_asr-models-execution.pdf
│   ├── nemo_asr-pyctcdecode-execution.pdf
│   ├── NLP-NER-validation.pdf
│   └── object-intent-classify.pdf
├── py
│   ├── action-intent-classify.py
│   ├── location-intent-classify.py
│   ├── nemo_asr-models-execution.py
│   ├── nemo_asr-pyctcdecode-execution.py
│   ├── NLP-NER-validation.py
│   └── object-intent-classify.py
└── Sarathi-Report.pdf
```

Folder & File details:
py --> contains python executables
ipynb --> contains ipynb files
ipynb_executed_pdf --> Contains execution outputs of ipynb files
asr_results --> Contains the files with results obtained by various Nvidia nemo ASR models
Sarathi-Report.pdf --> current document

How to execute:
Replace the **“base_path”** variable present in the begining of each file, & point it to the “task_dataset” path, then execute
Any missing dependencies would be reported, and they can be installed via conda or pip

Which file corresponds to what:
NLP:
action-intent-classify.ipynb --> file contains code for intent categorization for action field
location-intent-classify.ipynb --> file contains code for intent categorization for location field

object-intent-classify.ipynb --> file contains code for intent categorization for object field
NLP-NER-validation.ipynb --> NLP named entity recognition code

<u>ASR:</u>
nemo_asr-models-execution.ipynb --> Contains Nemo ASR models code (Jasper & Quartznet)
nemo_asr-pyctcdecode-execution.ipynb --> Contains Nemo ASR models code (conformer-CTC) and pyCTCdecode code
hugging_face_pretrained_usage.ipynb --> Code template for using facebook's Wav2vec2 pre-trained models similar to Nvidia Nemo ASR models.
espnet_training_gpu_google_colab.ipynb --> espnet net setup code for CMU an4 dataset that I ran on Google colab using GPU. I wanted to replace an4 dataset with our task dataset and train using espnet framework.

<u>For Reference & future implementation using GPU:</u>
nemo_asr-Transfer-learning-template.ipynb --> Template code for training from scratch using NVIDIA Nemo, and to do transfer learning to custom dataset. I could not get chance to test & run this script as this needs GPU.