Speech Emotion Recognition using Semantic Information

Navya (210657)

1 Implementation Details

All the codes given in the original github link by the author were ran other than evaluation. All the training codes(4 files) contained code which had attributes of tensorflow 1.x version and not compatible with tensorflow 2.x version which had to be converted and to work with tensorflow 2.x version and python 3.9

Folders segmentations and tfrecords present in the datasent link were generated by the code by running the first two codes.

For training, all the other codes also have to be copy pasted in the same jupyter notebook before running training as the training files uses those functions.

As aeneas is a python c/c++ library, if you are not able to download manually using pip as it is problematic, you can use the following link to install all its dependencies and library in one quick installer.

aeneas library download link

2 Dataset

Dataset provided only had the audio file and transcript file of the participants and did not contain the labels/annotations for emotional(arousal, valence, and liking) data.

The data was generated by me by studying the required format and the necessary fields. The folders(turns and labels) are both created by me, and the labels data contains all zeroes as this was to check whether the code would run.

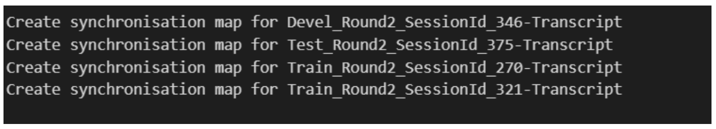
The data for which labeled data was present had no transcript for as there was no conversation happening in those files and the transcript of conversations was necessary for generating segmentation and training records.

3 Results

Running the two codes which generate the word segmentation using the audio and transcript file should give an output of this:

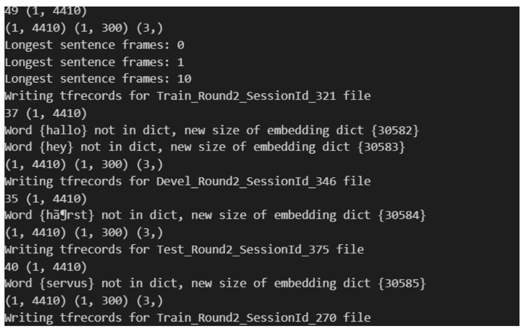
I was only able to train the model for 5 epochs using a small dataset containing only 4 files as otherwise my laptop was showing resource exhausted error/out of memory error.

As all the labels data was zero and not the actual data, the model did not show any improvement during training but rather the same loss was encountered in all five epochs.



```
Create synchronisation map for Devel_Round2_SessionId_346-Transcript
Create synchronisation map for Test_Round2_SessionId_375-Transcript
Create synchronisation map for Train_Round2_SessionId_270-Transcript
Create synchronisation map for Train_Round2_SessionId_321-Transcript
```

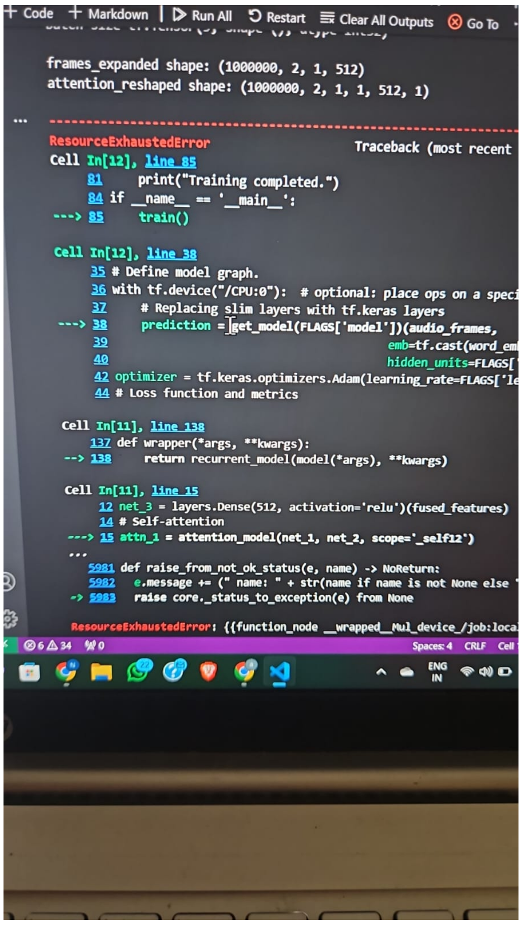
(a) speech2word mappings



```
49 (1, 4410)
(1, 4410) (1, 300) (3,)
Longest sentence frames: 0
Longest sentence frames: 1
Longest sentence frames: 10
Writing tfrecords for Train_Round2_SessionId_321 file
37 (1, 4410)
Word {hallo} not in dict, new size of embedding dict {30582}
Word {hey} not in dict, new size of embedding dict {30583}
(1, 4410) (1, 300) (3,)
Writing tfrecords for Devel_Round2_SessionId_346 file
35 (1, 4410)
Word {hã¶rst} not in dict, new size of embedding dict {30584}
(1, 4410) (1, 300) (3,)
Writing tfrecords for Test_Round2_SessionId_375 file
40 (1, 4410)
Word {servus} not in dict, new size of embedding dict {30585}
(1, 4410) (1, 300) (3,)
Writing tfrecords for Train_Round2_SessionId_270 file
```

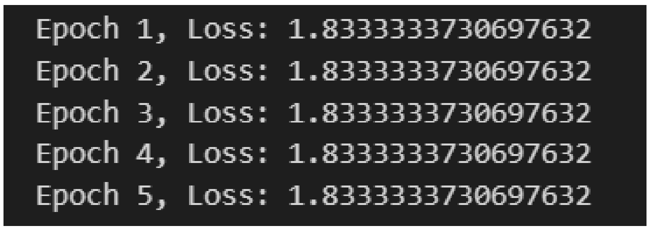
(b) generating tfrecords of the data

Figure 1: output



```
frames_expanded shape: (1000000, 2, 1, 512)
attention_reshaped shape: (1000000, 2, 1, 1, 512, 1)

---------------------------------------------------------------------------
ResourceExhaustedError                    Traceback (most recent
Cell In[12], line 85
     81     print("Training completed.")
     84 if __name__ == '__main__':
---> 85     train()

Cell In[12], line 38
     35 # Define model graph.
     36 with tf.device("/CPU:0"):  # optional: place ops on a speci
     37     # Replacing slim layers with tf.keras layers
---> 38     prediction = get_model(FLAGS['model'])(audio_frames,
     39                                            emb=tf.cast(word_em
     40                                            hidden_units=FLAGS[
     42 optimizer = tf.keras.optimizers.Adam(learning_rate=FLAGS['le
     44 # Loss function and metrics

Cell In[11], line 138
    137 def wrapper(*args, **kwargs):
--> 138     return recurrent_model(model(*args), **kwargs)

Cell In[11], line 15
     12 net_3 = layers.Dense(512, activation='relu')(fused_features)
     14 # Self-attention
---> 15 attn_1 = attention_model(net_1, net_2, scope='_self12')
...
   5981 def raise_from_not_ok_status(e, name) -> NoReturn:
   5982   e.message += (" name: " + str(name if name is not None else
-> 5983   raise core._status_to_exception(e) from None

ResourceExhaustedError: {{function_node __wrapped__Mul_device_/job:loca
```

(a) Resource Exhausted error



```
Epoch 1, Loss: 1.8333333730697632
Epoch 2, Loss: 1.8333333730697632
Epoch 3, Loss: 1.8333333730697632
Epoch 4, Loss: 1.8333333730697632
Epoch 5, Loss: 1.8333333730697632
```

(b) Training output

Figure 2: output