Action Unit Sequence Prediction based on Fundamental Frequencies Sequence

Please note that the following is just one example of how you can proceed with this lab. So, in case you have other ways of solving it, that's very good. What is important is to get results.

Perform Quantization/Discretization on the original continuous data (all your features)

A. Action units

Action Units intensities are continuous values that range between 0 and 5. You need to perform quantization to have specific discrete AU intensities between 0 and 5.

Apply ***linear quantization***: depending on your quantization step that you choose, you will end up with specific number of discrete AU values.

After performing the quantization, you need to build a dictionary **"ID_to_AU"** containing the discrete AU values that you got, and you need to map each value to an "ID", which will be the key in your dictionary.

So, for example, the following could be your ID_to_AU dictionary of AUs:

ID_to_AU: {0: SOS, 1: EOS, 2: 0.0, 3: 0.1, 4: 0.2, 5: 0.3, 6: 0.4, 7: 0.5, 8: 0.6, 9: 0.7, 10: 0.8, 11: 0.9, 12: 1.0, 13: 1.1, 14: 1.2, 15: 1.3, 16: 1.4, 17: 1.5, 18: 1.6, 29:1.7, 20: 1.8, 21: 1.9, 22: 2.0, 23: 2.1, 24: 2.2, 25: 2.3, 26: 2.4, 27: 2.5, 28: 2.6, 29: 2.7, 30: 2.8, 31: 2.9, 32: 3.0, 33: 3.1, 34: 3.2, 35: 3.3, 36: 3.4, 37: 3.5, 38: 3.6, 39: 3.7, 40: 3.8, 41: 3.9, 42: 4.0, 43: 4.1, 44: 4.2, 45: 4.3, 46: 4.4, 47: 4.5, 48: 4.6, 49: 4.7, 50: 4.8, 51: 4.9, 52: 5.0}

Ignore the SOS and EOS tokens for a minute, I'll explain them in another section.

To get the one hot vectors of all the values in this dictionary, you can use the function 'to_categorical' in Keras.

You also need to build another dictionary **"AU_to_ID"** that maps each AU to its ID, this way you can invert a one hot vector embedding back to its original value of AU.

```
AU_to_ID = {v:k for k, v in ID_to_AU.items()}
```

Check out this link for a more detailed explanation of the usage of dictionaries in one hot encoding: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/

A. F0

For the Fundamental Frequency, you need to clip all the values to the range of 50-550. Then you need to build the ID_to_F0 dictionary as explained in the previous section. No need to generate the F0_to_ID dictionary.

Teacher Forcing Mode

In the following explanation, consider we're trying to predict only AU01 (for simplicity)

As explained during the lab session, you need to use teacher forcing when training your Model. In teacher forcing mode, your model should take two inputs and one output:

1. Input 1: the sequence of F0s
2. Input 2: the expected prediction of AU01, shifted by one timestep.
    a. In this first timestep, you place a special token such as <START>, or <SOS> (=start of sequence), or anything you want
    b. You need to add this token to your Vocabulary of Action Units (ID_to_AU and AU_to_ID)
3. Output: the output of the model, which is the original sequence of AU01 but with a special token at its end: <END> or <EOS>. You also need to add this special token to your vocabulary of AUs.

So, the length of your AU Vocabulary will be equal to N+2 where N is the number of AU discrete values that you have.

Loss Function

Your loss function should be "categorical_crossentropy", since our problem consist of predicting "categories" and each "category" is represented by a one hot vector.

Inference Mode

In inference / prediction mode, we try to predict the Action Units without having the ground truth. Therefore, we only use Input 1 (sequence of F0s).

An example of how the inference mode is used in Seq2Seq model is provided in the following link: https://towardsdatascience.com/neural-machine-translation-nmt-with-attention-mechanism-5e59b57bd2ac

Some helpful general Seq2Seq examples

https://gist.github.com/kiwenlau/4e204587e715d95ad567ba0c37b9fa03

https://medium.com/analytics-vidhya/intuitive-understanding-of-seq2seq-model-attention-mechanism-in-deep-learning-1c1c24aace1e

https://www.kaggle.com/residentmario/seq-to-seq-rnn-models-attention-teacher-forcing

https://medium.com/@ayman.shams07/implementing-rnn-to-create-a-language-model-using-sequence-to-sequence-approach-using-keras-in-6e4370f88557

https://nextjournal.com/gkoehler/machine-translation-seq2seq-cpu