CSC 4343 Homework 3

Your task in this homework is to develop and train a transformer model to predict interactions between biological sequences. T-cells are a type of immune cells in our body that recognize anomalies in cells caused either by external factors (e.g., viruses) or internal ones (e.g., cancer). The recognition is through the interaction between its receptor (TCR) and the antigen presented on the cell surface, which is determined mainly by two Amino-Acid (AA) sequences (one from the receptor and the other from the antigen). Your model should take two sequences as input and predict whether the TCR and the antigen that present the sequences will interact.

The data is in the "data.csv" file. The following shows some examples. You can open the csv file in an excel spreadsheet. The "antigen" column gives the Amino-Acid sequences of the antigens and the "TCR" column shows the AA sequences of the T-cell receptors. The "interaction" column shows whether the pair interacts (1 indicates interaction and 0 indicates no interaction). The sequences contain upper case letters representing amino acids. There are total 20 different AAs in human body.

```
antigen,TCR,interaction
GILGFVFTL,CASSSRSSYEQYF,1
NLVPMVATV,CASSPVTGGIYGYTF,1
GLCTLVAML,CSARDGTGNGYTF,1
NLVPMVATV,CASRPDGRETQYF,0
NLVPMVATV,CASSETGFGNQPQHF,0
NLVPMVATV,CASSLAPGATNEKLFF,1
NLVPMVATV,CASSLAPGTTNEKLFF,1
```

Note that the antigen sequences and the TCR sequences may be of different lengths. You should have a preprocessing function to pad the sequences (using the letter "X") so that all the antigen sequences have the same length and so are all the TCR sequences. (The length of the antigen sequences may not necessarily be the same as that of the TCR sequences.)

Implement the model in **Figure 1** on the right to make interaction prediction. For the two transformers, use nn.TransformerEncoder with 3 layers of nn.TransformerEncoderLayer. Each layer has d_model=256 and nhead=8. You should use an nn.Embedding layer to translate the AA tokens to embedding vectors.

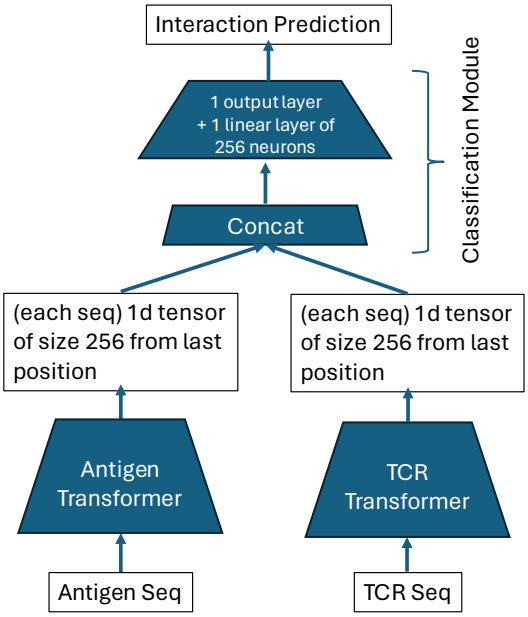

Figure 1 Full Prediction Model

Before training the full model to predict interaction, you should pre-train the transformers. Pretrain the antigen transformer using the antigen sequences and pretrain the TCR transformer using the TCR sequences. Pretraining here is the same as the training process in your HW2. That is, you train the model to predict the next token (AA) given the tokens (AAs) that the model sees so far. (Do not use bos/eos tokens as in your HW2. Only predict real AA tokens in this pretraining.)

After pretraining the two transformers, you can combine them with the classification module to obtain the full prediction model. Use a 3-fold cross validation (CV) to evaluate the prediction performance of the full model. In each iteration of the cross validation, 1/3 of the data is left aside and the other 2/3 is used to train the model. The trained model is then used to make predictions on the left-aside data and the prediction accuracy should be calculated. The average accuracy of the 3 iterations is the benchmark measurement of the model's performance. To save time, you can use the whole dataset for pretraining. CV needs to be conducted only for the full model that makes the interaction prediction.

Also construct a full prediction model using transformers not pretrained. Use 3-fold CV to benchmark the performance of such model as well. Compare the performance between the model using transformers without pretraining and the one with pretraining. Calculate how much gain in performance the pretraining can provide (i.e., the difference of the average accuracy from 3-fold CV between the two models).

Homework Submission:

Upload a Python file (not .ipynb file) named **SeqModels.py** in moodle. You need have the following functions in the file:

1. A function **make_tfm_model(T)** which takes a parameter **T** (a string with value either "tcr" or "antigen" and returns a PyTorch object (untrained, subclass of nn.Module) that implements the transformer model for the corresponding tcr/antigen sequences.
2. A function **make_predict_model(M_antigen, M_tcr)** which takes two parameters: **M_tcr** is the TCR transformer model and **M_antigen** is the antigen transformer model, and return the full model that makes the prediction on the TCR-antigen interaction.
3. A function **pretrain_tfm_model(M, D, n_epochs)** which takes a transformer model **M** and a training data set **D** (a list of sequences) and pretrain the model **M** as specified above for **n_epochs**. At the end of each epoch, print out the average loss during that epoch.
4. A function **train_model(M, L_antigen, L_tcr, Interaction, n_epochs)** which trains the full prediction model **M** passed to it for **n_epochs** using the following data: **L_antigen**, a list of antigen sequences, **L_tcr**, a list of TCR sequences, and **Interaction**, list of the numbers (0/1) indicating interaction or not. (Clearly these three lists are of the same length.) At the end of each epoch, print out the average loss during that epoch.
5. A function **load_trained_model()** which should download your trained model saved somewhere online and load it from the downloaded file. The function then returns the model. Do not train the model from scratch in this function. Same as HW1/2, you should not upload the model file to moodle. Instead, you should share it in your google drive (or other online storage which can be *shared by link*).
6. A function **predict(M, L_antigen, L_tcr)** which takes a full prediction model **M** and **L_antigen**, a list of antigen sequences, **L_tcr**, a list of TCR sequences. (Again the two lists are of the same length.) For each corresponding pair of sequences in **L_antigen** and **L_tcr**, the function should

use the model to make a prediction on their interaction. Return a list of the prediction results (0/1).

7. In a comment section at the beginning of the .py file, report the result from your 3-fold cross validations. For each fold/run, report the accuracy of the trained model on the left-aside test data. Also report the average of the 3-folds (total 4 numbers/measures for a model). Report the measures for both the model with and the one without pretraining. Show the performance gain by pretraining and discuss briefly whether it meets your expectation.

Make sure we can import these functions from the .py file without running your training code. If we want to train the model, we will explicitly call the training functions.

We will also test your model by code similar to the following:

```
from SeqModels import make_seq_model, pretrain_seq_model, make_predict_model,
train_model, load_trained_model, predict

m_tcr = make_tfm_model('tcr')
m_ant = make_tfm_model('antigen')

pretrain_tfm_model(m_tcr, list_tcr_seq, 10)
pretrain_tfm_model(m_ant, list_antigen_seq, 10)

model = make_predict_model(m_ant, m_tcr)

train_model(model, list_antigen_seq, list_tcr_seq, list_interact, 10)

tm = load_trained_model()
pred = predict(tm, new_list_antigen, new_list_trc)
```