CSC 7343 Deep Learning Systems Final Project

Due Dec. 4th (Late submission will not be graded.)
You may form a group up to 2 persons to do the project.

Project Task

Your task in this project is to develop a pretraining strategy to improve sequence classification. The problem under consideration is to predict whether a T-cell can interact with an antigen. T-cell is 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.

The data for the project 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 (AA) 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
```

The interaction prediction problem can be cast into a sequence classification problem. For example, one can convert the first pair of sequences in the above example into the following "combined" sequence:

```
<cls>GILGFVFTL<sep>CASSSRSSYEQYF
```

where <cls> and <seq> are two additional tokens (tokens besides the AA symbols) marking the start of the combined sequence and the separator for the original sequences. Note that the antigen sequence and the TCR sequences may be of different lengths. In your preprocessing, you need to pad the sequences first so that all the antigen sequences have the same length and so are all the TCR sequences.

A model is given in the "model.py" which can predict for each combined sequence whether there is interaction. The model has two components: a transformer network and a feedforward network (the classifier). The classifier takes the output from the transformer and produces classification results. When you call the model and pass an input, you can use the parameter "classification" to switch the output of the model. When classification=False, the model returns the output of the transformer, a 3D tensor of size (batch, length, states). When classification=True, the model returns the classification logit, a 2D tensor of size (batch, 2). The loss function for the classification is also provided in the "model.py" file.

You should first train the model so that it can make good sequence prediction. Evaluate the model's performance by doing a 3-fold cross validation (CV) on the dataset and calculating the AUC value.

Your main task is to design a (no label) pretraining scheme to pretrain the transformer component of the model. (***"no label" means the pretraining process should not use the interaction information in the data, just the sequences.***) Afterward, the whole model with the pretrained transformer can be further trained (fine-tuned) using the labels for interaction. Your pretraining should use only the sequences in the provided dataset. Evaluate the performance of the model with pretraining by doing a 3-fold cross validation on the dataset. (To save time, you can use the whole dataset for pretraining as long as labels are not used in the pretraining stage. CV can be conducted only for the fine-tuning stage.)

Compare the performance (AUC) between the model without pretraining and the one with pretraining. Measure how much performance gain the pretraining stage can provide.

Submission

Code and models: you should have one .py file. It should include:

1) Your data preprocessing code (as a function);
2) code implementing your pretraining scheme;
3) code for training the model for classification;
4) In a comment section at the beginning of the .py file, put the link to your trained models (one with pretraining and one without). After downloading them, we can call the "load" function with the file name of the download to load the model and perform classifications.

(***Note that we will test your code and models on another dataset that is of the same format as "data.csv" but with different sequences.***)

Writeup: write a short report on what you did, including

1) the pretraining scheme you design and use;
2) the amount of performance gain (or loss?) by conducting pretraining;
3) Analysis of the reason behind the result and any lesson learned.

(**Innovative pretraining schemes different from the GPT or Bert pretraining will be valued. Your pretraining scheme may not provide a significant performance gain but it needs to make sense in a theoretical way.)**

Save the writing in a pdf file. **Submit both the report pdf file and the project.py in moodle**.

If you work as a group, only one person in the group needs to submit the files in moodle. In this case, you must put the names of all your group members on the pdf so that we know who are in the group.