CSCI 4820/5820 (Barbosa)
Project 4: Multiclass RNN Classification Exploration Challenge

Due: **See course calendar**. May not be turned in late.

Assignment ID: **proj4**
File(s) to be submitted: **proj4.ipynb, pro4doc.pdf proj4.pdf**

Note: You are encouraged (but not required) to run your code on the **TAMU FASTER** system to get familiar with it.

****** This project requires the **Spring 2024 SIF** on TAMU FASTER (see image below) ******

Objective(s): This is a research assignment where you will have the opportunity to make modifications to a recurrent neural network to improve its performance. **The Top 5 performing solutions will be announced in class**.

Project description: The aim of this project is to classify speeches from central bank officials into the country they are from (the European Union should be treated as a country in this case). You are provided starter code that defines and executes a recurrent neural network (RNN) model for multiclass classification. You may change the type of cells used (RNN, GRU, LSTM) and model options and (hyper)parameters but you may not use other (or hybridized) neural network architectures (such as adding CNNs, transformers, etc.) or other classification frameworks (keep the code in PyTorch).

Dataset description: This is a small dataset of central bank speeches for various countries and the European Union. There are 4,240 excerpts of speeches, which have been truncated to 512 tokens. The speeches are separated from their respective classes by a tab. There are six classes:

- canada
- euro area
- japan
- sweden
- united kingdom
- united states

Starter code description: The starter code is similar to what has been demonstrated in class. It declares the model, trains it, and then tests it. It is set up to access the pre-trained embeddings and the dataset from the instructor's scratch space on TAMU (change this if not using that system). Model class arguments are:

- vocab_size – number of tokens in the vocabulary
- embedding_dim – dimension of pre-trained embeddings
- hidden_dim – dimension of hidden layer
- output_dim – number of outputs (classes)
- rtype – RNN cell type
- n_layers – number of RNN layers
- dropout_rate – the dropout rate
- bidirectional – bidirectionality True or False
- embed – pre-trained embedding matrix or *None* to train own embeddings
- freeze – True or False to freezes pre-trained embeddings (valid only if training own embeddings)

Requirements: Where two weights are shown for an item, the first is for CSCI 4820 and the second for CSCI 5820 credit.

1. (**80%/60%**) Your program must provide the functionality listed below, in a single file named ***proj4.ipynb***:

- Study the starter code and run it. Note: The code has been tested and should run without problems.
- **This is a research project - You are on your own for determining the changes to make to improve the model's performance. You may use all resources at your disposal to make modifications but must document sources. The quality of your code alterations matters. A submission with very few changes that is basically just the starter code will not earn a high grade**. **Your grade will be higher for "bolder" changes that work well.**
- Here are some ideas for changes that go being beyond modifying hyperparameters (this is not exhaustive – be curious and creative in your approach). **More substantive changes are marked with asterisks and will earn extra credit, if they work well**.
    - Maximum sequence length (number of tokens)
    - Train/Validation/Test split
    - Batch size
    - * One-hot class labels
    - * Additional fully connected layers
    - * Converting the network to LSTM
    - * Other changes (as long as they comply with requirements)

    Note: If using TAMU FASTER, the **pre-trained embeddings** are in the instructor's scratch space (the code looks for them there). If not using TAMU FASTER, download the embeddings from the course calendar (https://www.cs.mtsu.edu/~sbarbosa/6350/private/GoogleNews-vectors-negative300-SLIM.zip).

2. (**10%**) Test your program changes as you make them – Ensure your best and final model runs and produces results.

3. (**10%**) Code comments - Add the following comments to your code:

    - Place a Markdown cell at the top of the source file(s) with the following identifying information:

        Your Name
        CSCI <course number>-<section number>
        Project #X
        Due: mm/dd/yy

    - Add a Markdown cell above each code cell that describes the processing done in the code cell.

4. (**CSCI 5820 Students only 20%**) Analysis – Provide a **detailed** analysis (in a Markdown cell) addressing the following:

    a) Fully describe the changes made to get your best performing model
    b) Give an analysis of the path you took to your best performing model by describing at least two interim states (not including the code's starting state or your final model). Focus on why each change improved/decreased performance.
    c) Analyze the confusion matrix, focusing on both correct and incorrect predictions and confusion between classes and address plausible reasons for class confusions.
    d) Discuss the quality of the dataset and any impact it has on classification.

5. Generate a pdf file of the notebook (from the terminal):
    $ *jupyter nbconvert --to html proj4.ipynb*
    $ *wkhtmltopdf proj4.html proj4.pdf*

6. Document all input and output from AI tools in the file ***proj4doc.pdf***

7. Submit required files via the D2L dropbox for this project

Note: Launch interactive Jupyter **Notebook** (not Jupyter Lab) and simply change "tree?" in the URL to "lab") – see below.

1. Initialize a T4 GPU instance on TAMU FASTER with settings shown below left (note that the *Path to singularity image file* should be set to the Spring 2024 image exactly as shown in yellow below – **do not change the path to include your userid**). Refer to the **video** on D2L if you need to. Enter a good estimate for the length of GPU sessions (a **maximum of 3 - 4 hours** is reasonable for this project) … see encircled entry field. Our account is charged for the length of time a GPU is held, not the amount of time it is actually in use. Using the GPU, the starter code runs in less than two minutes.
2. If GPUs are not available, the job will be queued and the notebook won't start. In that case, you can launch a CPU only instance, if desired, by selecting *non-GPU* as the *Node type* (circled in red below right). Running the starter code in CPU-only mode takes around 45-50 minutes.
3. Regardless of method used, **shutdown** Jupyter Lab and **delete** the instance under the Interactive Sessions dropdown when done with the work session (re-instantiate when you're ready to work again).

T4 GPU Session	CPU Session
Type of environment	Type of environment
Containers (Singularity)	Containers (Singularity)
Select the type of environment in which Jupyter is installed. Help me choose	Select the type of environment in which Jupyter is installed. Help me choose
Path to singularity image file	Path to singularity image file
/scratch/user/u.jp60244/sif/csci-2024-Spring.sif	/scratch/user/u.jp60244/sif/csci-2024-Spring.sif
Enter the path to a singularity image file containing the Jupyter app. Recommended that this live under your $SCRATCH directory.	Enter the path to a singularity image file containing the Jupyter app. Recommended that this live under your $SCRATCH directory.
If your container image is on the internet, please see instructions.	If your container image is on the internet, please see instructions.
Number of hours (max 168)	Number of hours (max 168)
1	1
Number of cores (max 64)	Number of cores (max 64)
4	4
Total GB memory (max 240)	Total GB memory (max 240)
32	32
Node type	Node type
T4	non-GPU
• Select a GPU node for software that supports GPU processing.	
Number of GPUs	
1	