Instructions for homework submission

1. Complete one part in this homework:
   - Programming questions: Complete the given skeleton `Python` code. **For questions requiring visualization, analysis, and discussion, please include your solution in the same LaTeX document.**
2. Submit your work to Gradescope including:
   - A completed `Python` code: `FirstName_LastName_HW5.py`.
   - **Two pretrained model files: `lstm.pt` and `transformer.pt`**
   - **For this assignment, there is only one submission portal on `Gradescope`.**
   - Failure to follow submission instruction will result in loss of marks.
3. Start early!
4. Total: 100 points.

If you find that training your model is taking too long or if you require additional computational power, you are encouraged to explore the High Performance Research Computing (HPRC) resources available at Texas A&M University. For more information and to access these resources, please visit `https://hprc.tamu.edu/`.

This project utilizes a subset of the IMDB database[1]. We acknowledge and appreciate their efforts in collecting and maintaining this data.

[1] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Learning word vectors for sentiment analysis.* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 142–150.

Programming Question

In this homework, you are ask to perform a sentiment analysis using two classic models, a LSTM and a encoder only Transformer. In particular, you are given the IMDB dataset, which contains thousands of movie reviews, and you want to understand whether their tone is positive or negative using your machine learning models. You will implement a few primary components:

1. A `Vocabulary` class, design to build the vocabulary (map your text token to indices) for your language models. Your implementation is expected to handle input text token outside the vocabulary.

   (Note: You are given three tokens to start. Please do not change the index assigned to these tokens.)

2. A PyTorch Dataset class, `IMDBDataset`, designed to load text and its label from a given parquet file. You are asked to implement the `IMDBDataset` class such that:
   - It inherits from `torch.utils.data.Dataset`.
   - It handles data for both `LSTM` and `TransforemerEncoder`. In particular, you are expected to return tensor for indexed text and its corresponding label for `LSTM`, and an attention mask for `TransformerEncoder` on top of that. Note: please use 1 for real tokens and 0 for padding in your attention mask.
   - The `__len__` method returns the total number of samples in your dataset.

3. A `load_and_preprocess_data` function. This function is expected to handle data loading, preprocessing, and return a dataloader can be directly used for your model. Again, this needs to work with both training data and your own split of validation data, and both `LSTM` and `TransformerEncoder`.
   - It must accept a `shared_vocab` argument and capable of utilizing the same vocabulary across training and testing.
   - It must accept a `data_type` argument and return two values representing `dataloader` and a built vocabulary if this argument is set to train, and return one value representing `data_type` otherwise.
   - It must accept a `model_type` argument and handle the different dataloader required by `LSTM` and `TransformerEncoder` accordingly.
   - Autograder will call this function with training data to build the vocabulary, and then load the test data using the returned vocabulary.

4. A LSTM model, `LSTM`. At least one `nn.LSTM` layer is expected.

5. A Transformer model, `TransformerEncoder`. At least one `nn.TransformerEncoder` layer is expected. You also need one more `PositionalEmbedding` class to complete the implementation.

You will then train your models on the training split of the IMDB dataset, save your best trained model to `lstm.pt`, `transformer.pt` and aim for **85%** classification accuracy on our reserved test set.

Best Practice & Tip

Only one more tips for this assignment.

1. **Avoid overfitting:** Create a validation split and monitor validation accuracy. Adjust your hyperparameter as needed.

Important Constraints

- You are **NOT** allowed to use pretrained model from internet. Your submitted model weights need to be learned from scratch.
- **DO NOT** attempt to manipulate your training data in a way that gives you an unfair advantage.
- The total size of your submission package must not exceed 100 MB.

Failure to follow these constraints will disqualify your submission.