Assignment 2

Text Classification

Submission deadline: 15.03.2024

General Instructions:

- Submission via moodle, must include code (and all necessary files to run it).
- The code must be written in Python 3.6 or higher, and must run.

Assignment:

In this assignment you will be required to design, construct, and train an RNN model that classifies input lyric (a sequence of words) to the artist that composed it.

For this assignment, there would not be design or resource limitation, beyond that which Colab imposes. Try reaching as high accuracy as possible.

Methodology:

1. Use LSTM to process the song as a sequence of words. The class torch.nn.LSTM incorporates all the options learned in class. Use the cell type that performs best for this task.
2. First, tokenize the input sequence using pytorch tokenizer. To do this, use torch.nn.Embedding(N,E), where N is the number of tokens in the entire vocabulary, and E is the dimension to which you embed the tokens to. This creates a trainable layer of embedding.
   The input of the embedding class, once constructed, is a **Tensor** of n integer indices.

For example:
Text = “this sentence is long. This is a long sentence”
Tensor of indices = (1, 2, 3, 4, 1, 3, 5, 4, 2)
And the tensor can be given as input to the embedding layer.
You can download and use a pretrained embedding layer, instead of training your own as a trainable layer.

Data:

CSV files songdata_train.csv, songdata_test.csv are available in the assignment box; load them into your Notebook.

- The python file LyricsDataset.py contains a class you can use as datasets for loading the files into the workspace.
- You can use torch class DataLoader() to construct a data loader object for easily iterating through the LyricsDataset dataset.

Useful Tips:

- It may be helpful to “clean” up the original text.
- Feel free to change or add methods to LyricsDataset as you see fit.
- Training may take a long time; especially if the GPU can no longer be used. To prevent unexpected stops, save the model on training step. It is likely for your model to converge within a very low number of epochs.

Report:

1. Report your architecture, number of parameters, training time (real life time) and performance on the test set.
2. Add a plot showing train error, test error, and accuracy as a function of **Global batch steps** performed (**not epochs!)**.
3. Explain **shortly** how your design deals with inputs of varied length.
4. Explain **shortly** what LSTM you propagated forwards, and why.

Submission:

Your submission should include:

- All the networks you were required to design, and the code used to train them.
- A saved file of your trained model, and a code that loads and tests it.
    - See: https://pytorch.org/tutorials/beginner/saving_loading_models.html
- A PDF assignment report.

Good luck!