DL/DLOps (2023)
DLOps Assignment 2: RNN, LSTM and Docker [100 Marks]
Deadline: 31/03/2023, 23:59:59

Programming instructions:

1. Programming language: Python
2. **Use of PyTorch is compulsory**. No marks shall be given for TensorFlow implementation.

Reporting instructions:

1. Please submit all your working codes as .py or .ipynb files.
2. A single report (PDF) file containing all relevant information, including data pre-processing, observations, results, and analysis across the problem, should be submitted. Do not put snapshots of code in the report.
3. The report should be detailed and clearly explain every step you have followed. All the intermediate outputs and their inferences should be present in the report. The PDF file should be properly named with your complete roll number XYZ (ex: "XYZ_DLOps_Assignment-2.pdf"), with your name and roll number mentioned inside the report as well.
4. Record a video of your terminal that shows that your written codes are working (compulsory).
5. Mention any resources, articles, or (GitHub links that you may have used as references to solve any question of the assignment in the references section of the report.
6. Make sure all the submission files along with the working codes are included in a single zip file.

General instructions:

1. Do not plagiarize from the internet or your peers. The institute's plagiarism policy will be strictly enforced.
2. The assignment will be evaluated out of 50% of the total marks in the event that a report is not submitted.
3. **Use any of the PyTorch Container for both the questions.**

Q1. You have been provided a DATASET, which contains pairs of the words (x,y) i.e. akhbaar अख़बार in which the first word is a Latin word(words we usually type while chatting with friends in WhatsApp) and the second word is its corresponding word in native script. Your main goal is to train a seq2seq model which takes as input the romanized string and produces the corresponding word in native script.

For Example, Jabki yah Jainon se km hai. ⇒ जबकि यह जैनों से कम है। **[75]**

a) Build a seq2seq model which contains the following layers - [20]

(i) input layer for character embeddings
(ii) one encoder which sequentially encodes the input character sequence (Latin)
(iii) one decoder which takes the last state of the encoder as an input and produces one character output at a time (native).

Please note that the dimension of input character embeddings, the hidden state of encoders and decoders, the cell(RNN and LSTM), and the number of layers in the encoder and decoder should be passed as an argument.

(Note:- For Reference you may refer to this Blog, but the implementation must be in PyTorch only.)

b) Now train your model using the standard train, test, and val data provided in the dataset. Try below mentioned hyperparameters and draw the correlation table along with the plot (loss/accuracy VS. hyperparameter) for both RNN and LSTM. [20]

(i) Input embedding size: 16, 64
(ii) number of encoder layers: 1,3
(iii) number of decoder layers: 1,3
(iv) hidden layer size: 16,64

c) Based on the above parts please try to answer the following- [10]

(i) Does RNN takes less time to converge than LSTM? Reason to support your answer.
(ii) Does dropout leads to better performance? Proof to support your answer.
(iii) Using a smaller size in hidden layer does not give good results? Proof to support your answer.

d) Now add an attention network to your LSTM seq2seq model and train your model along with the hyperparameter tuning. (you can use single or multiple attention layers) [25]

(i) Plot the accuracy/loss.
(ii) Report the test accuracy.
(iii) Does the attention-based model performs better than the LSTM? Proof to support your answer.

Q2. The dataset you have been given is Individual household electric power consumption dataset. **[25]**

(i)Split the dataset into train and test (80:20) and do the basic preprocessing. [10]

(ii) Use LSTM to predict the global active power while keeping all other important features and predict it for the testing days by training the model and plot the real global active power and predicted global active power for the testing days and comparing the results. [8]

(iii) Now split the dataset in train and test (70:30) and predict the global active power for the testing days and compare the results with part (ii). [7]