ID1: 208560086

ID2: 301613501

Deep Learning

2022-2023

TEL AVIV UNIVERSITY

EX2 - RNN

➢ **General**

This readme file describes how to run the code in the notebook of exercise 2.
The code is written such that we train all the cases and show the required graphs on a *Tensorboard*.

➢ **Run the code**

Prior to the training or loading weight step, please run the following steps \ functions \ cells:

Cell 1: Mount to google drive.
Cell 2: Import libraries.
Cell 3: Set the device that PyTorch will use for computation based on the available hardware resources.
Cell 4: Load Tensorboard extension.
Cell 5: Run the preprocess_data function.
Cell 6: Run the batchify function which divides the data to batches.
Cell 7: Run the loader function which returns a list of tuples that each tuple contains a sequence of inputs and corresponding outputs for a single batch.
Cell 8: Preprocessing the data
Cell 9: Run the RNN model class
Cell 10: Run the get_model function
Cell 11: Run the repackage_hidden function
Cell 12: Run the train function
Cell 13: Run the evaluate function
Cell 14: Run the test function
Cell 15: Preprocess on data
Cell 16: Run the train_RNN function
Cell 17: Initialize dictionary
Cell 18: Train the LSTM and save the perplexity in dictionary
Cell 19: Train the GRU and save the perplexity in dictionary
Cell 20: Train the LSTM with Dropout and save the perplexity in dictionary
Cell 21: Train the GRU with Dropout and save the perplexity in dictionary
Cell 22: Create summary table

Cell 23: Show the results on Tensorboard

➢ **How to train**

Run the *train_lenet5* function as shown in the following table.

NN Type	Command
LSTM	`train_RNN(batch_size=20, seq_len=20, num_epochs=13, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0, rnn_type='lstm',opt_type='sgd', learning_rate=1, factor=0.5, steps=4, weights_path=None)`

GRU	`train_RNN(batch_size=20, seq_len=20, num_epochs=13, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0, rnn_type='gru',opt_type='sgd', learning_rate=1, factor=0.5, steps=4, weights_path=None)`
LSTM + Dropout	`train_RNN(batch_size=20, seq_len=20, num_epochs=75, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0.5, rnn_type='lstm',opt_type='sgd', learning_rate=1, factor=0.9, steps=10, weights_path=None)`
GRU + Dropout	`d['gru with dropout'] = train_RNN(batch_size=20, seq_len=20, num_epochs=40, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0.5, rnn_type='gru', opt_type='adam', learning_rate=0.001, factor=0.5, steps=10, weights_path=None)`

Function Arguments:

- ➔ batch_size: The batch size used during training and evaluation.
- ➔ seq_len: The sequence length, i.e., the number of time steps in each input sequence.
- ➔ num_epochs: The number of training epochs.
- ➔ embedding_dim: The dimensionality of the word embeddings.
- ➔ hidden_dim: The dimensionality of the hidden state of the RNN.
- ➔ num_layers: The number of layers in the RNN.
- ➔ dropout: The dropout probability to apply between RNN layers.
- ➔ rnn_type: The type of RNN to use (e.g., 'LSTM', 'GRU').
- ➔ opt_type: The type of optimizer to use (e.g., 'SGD', 'Adam').
- ➔ learning_rate: The learning rate for the optimizer.
- ➔ factor: The factor by which the learning rate is reduced in the learning rate scheduler.
- ➔ Steps: The number of steps after which the learning rate is adjusted. This is relevant only when choosing the SGD optimizer. In the Adam optimizer, the learning rate will change automatically if the validation perplexity isn't improving.
- ➔ weights_path (optional): The path to save the weights of the best model. If not provided, a default path is generated based on the RNN type and dropout value.

Output:

- ➢ train_perplexity: The perplexity of the model on the training set after the training process.
- ➢ val_perplexity: The perplexity of the model on the validation set after the training process.
- ➢ test_perplexity: The perplexity of the model on the test set after the training process.

- ➢ **How to test with saved weights**

  Run the *test_lenet5* function as shown in the following table.

NN Type	Command
LSTM	`test_RNN(batch_size=20, seq_len=20, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0, rnn_type='lstm', weights_path='lstm_dpFalse')`

GRU	`test_RNN(batch_size=20, seq_len=20, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0, rnn_type='gru', weights_path='gru_dpFalse')`
LSTM + Dropout	`test_RNN(batch_size=20, seq_len=20, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0, rnn_type='lstm', weights_path='lstm_dpTrue')`
GRU + Dropout	`test_RNN(batch_size=20, seq_len=20, embedding_dim=200, hidden_dim=200, num_layers=2, dropout=0, rnn_type='gru', weights_path='gru_dpTrue')`

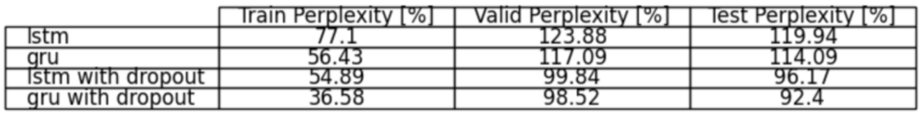
Function Arguments:

➔ batch_size: The batch size used during testing.
➔ seq_len: The sequence length, i.e., the number of time steps in each input sequence.
➔ embedding_dim: The dimensionality of the word embeddings.
➔ hidden_dim: The dimensionality of the hidden state of the RNN.
➔ num_layers: The number of layers in the RNN.
➔ dropout: The dropout probability to apply between RNN layers.
➔ rnn_type: The type of RNN used during testing (e.g., 'LSTM', 'GRU').
➔ weights_path (optional): The path to the saved model weights. If not provided, the function assumes that the weights are already loaded in the model.

Output:

➢ test_loss: The loss value of the model on the test set.
➢ test_perplexity: The perplexity of the model on the test set.

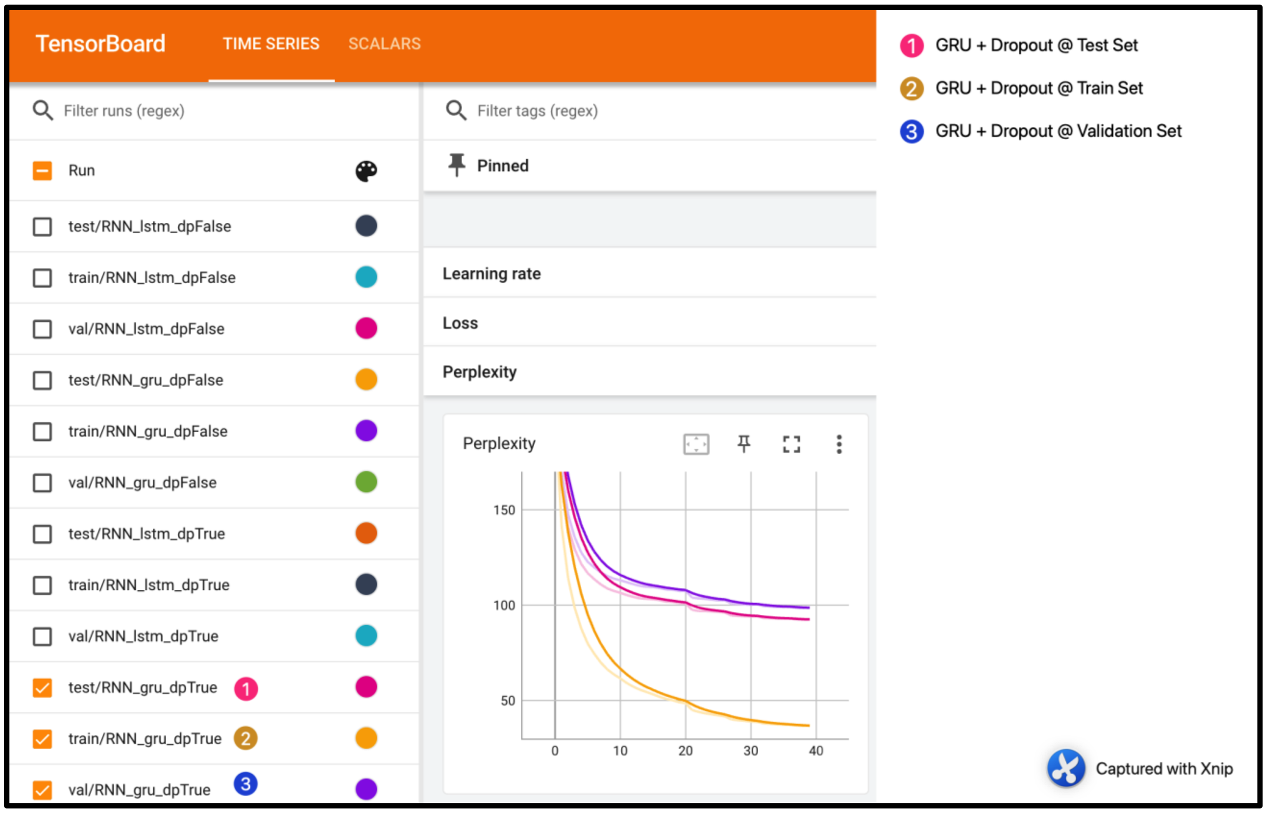
➢ By running Cell 22 you can see a **table of the summary**



	Train Perplexity [%]	Valid Perplexity [%]	Test Perplexity [%]
lstm	77.1	123.88	119.94
gru	56.43	117.09	114.09
lstm with dropout	54.89	99.84	96.17
gru with dropout	36.58	98.52	92.4

➢ By running Cell 23 (`%tensorboard --logdir logs`) you can see the Tensorboard visualization of all the graphs. Then follow the illustration below in order to choose the relevant graph\s you want to see. It is possible to compare all the train-test combinations.

➢ For example - if we want to compare the test and the train vs. test vs. validation sets of the **GRU + Dropout**, we will choose train/RNN_gru_**dpTrue**, test/RNN_gru_**dpTrue** and val/RNN_gru_**dpTrue** (accordingly)



1 GRU + Dropout @ Test Set

2 GRU + Dropout @ Train Set

3 GRU + Dropout @ Validation Set