UDACITY

‹ Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

# Generate TV Scripts

REVIEW

CODE REVIEW

HISTORY

## Meets Specifications

Great job! The code is perfect. Overall, the student has made a good effort and written a very clean and efficient code. Keep up the good work!

### All Required Files and Tests

**The project submission contains the project notebook, called “dlnd_tv_script_generation.ipynb”.**

**All the unit tests in project have passed.**

### Pre-processing Data

The function `create_lookup_tables` create two dictionaries:

- Dictionary to go from the words to an id, we'll call vocab_to_int
- Dictionary to go from the id to word, we'll call int_to_vocab

The function `create_lookup_tables` return these dictionaries as a tuple (vocab_to_int,

int_to_vocab).

This function has been written perfectly. Counter has been used to sort the words according to their frequency. Good job!

The function `token_lookup` returns a dict that can correctly tokenizes the provided symbols.

## Batching Data

The function `batch_data` breaks up word id's into the appropriate sequence lengths, such that only complete sequence lengths are constructed.

This function has been perfectly implemented. Great work!

In the function `batch_data`, data is converted into Tensors and formatted with TensorDataset.

Finally, `batch_data` returns a DataLoader for the batched training data.

## Build the RNN

The RNN class has complete `__init__`, `forward`, and `init_hidden` functions.

This class has been perfectly implemented. Great work!

The RNN must include an LSTM or GRU and at least one fully-connected layer. The LSTM/GRU should be correctly initialized, where relevant.

LSTM has been used.

## RNN Training

- Enough epochs to get near a minimum in the training loss, no real upper limit on this. Just need to make sure the training loss is low and not improving much with more training.
- Batch size is large enough to train efficiently, but small enough to fit the data in memory.

No real “best” value here, depends on GPU memory usually.

- Embedding dimension, significantly smaller than the size of the vocabulary, if you choose to use word embeddings
- Hidden dimension (number of units in the hidden layers of the RNN) is large enough to fit the data well. Again, no real “best” value.
- n_layers (number of layers in a GRU/LSTM) is between 1-3.
- The sequence length (seq_length) here should be about the size of the length of sentences you want to look at before you generate the next word.
- The learning rate shouldn’t be too large because the training algorithm won’t converge. But needs to be large enough that training doesn’t take forever.

Good hyperparameters have been chosen. The network trains as expected. Great job!

**The printed loss should decrease during training. The loss should reach a value lower than 3.5.**

Great job here! The neural network has been trained for a sufficient number of epochs and you've chosen good hyperparameters, so the training looks good- the cost first decreases and then plateaus. This is exactly how it's expected to be.

**There is a provided answer that justifies choices about model size, sequence length, and other parameters.**

Sufficient explanation has been provided. The student has done a good amount of hyperparameter tuning in an organized manner.

## Generate TV Script

**The generated script can vary in length, and should look structurally similar to the TV script in the dataset.**

**It doesn’t have to be grammatically correct or make sense.**

The generated script does look similar to the script in the dataset. For example, you can see that the network has learnt to open and close brackets and to start every line with the name of a character followed by a colon.

DOWNLOAD PROJECT

RETURN TO PATH