Generate TV Scripts

- Review
- Code Review
- History

Meets Specifications

Congratulations on meeting all specifications for this project! This was definitely not an easy project to pass, given the complexity of the task and concepts being tested. Good luck with the rest of the course!

You could look to apply what you've learned to the following problems:

- Generate your own Bach music using like DeepBach.
- Predict seizures in intracranial EEG recordings on Kaggle.

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).

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

Good job. Here's a good resource discussing more preprocessing steps that you can try:

Preprocessing text before using an RNN

Batching Data

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

Overall, good work implementing the `batch_data` function. You may also choose to create a generator that batches data similarly but returns x and y batches using `yield`. A generator allows you to create a function that behaves like an iterator in a fast and clean approach and they do not store their contents in memory.

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.

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

The RNN class has been implemented correctly.

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.

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

The cross entropy loss is below 3.5. Excellent work!

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

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 resembles the one in the dataset!