* [Logout](https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=84086d0b-3730-4799-987d-1eff0da9c77f&bsft_uid=eda41b30-1789-424b-9523-64c937f26cef&bsft_mid=ba5bec3d-987b-4894-87ea-8f73066c96e5&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=fd07f62c-998f-4c97-bd8b-6284dd46d91a)

  Back to Deep Learning

Generate TV Scripts

* [**REVIEW**](https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=84086d0b-3730-4799-987d-1eff0da9c77f&bsft_uid=eda41b30-1789-424b-9523-64c937f26cef&bsft_mid=ba5bec3d-987b-4894-87ea-8f73066c96e5&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=fd07f62c-998f-4c97-bd8b-6284dd46d91a)
* [**CODE REVIEW**](https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=84086d0b-3730-4799-987d-1eff0da9c77f&bsft_uid=eda41b30-1789-424b-9523-64c937f26cef&bsft_mid=ba5bec3d-987b-4894-87ea-8f73066c96e5&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=fd07f62c-998f-4c97-bd8b-6284dd46d91a)
* [**HISTORY**](https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=84086d0b-3730-4799-987d-1eff0da9c77f&bsft_uid=eda41b30-1789-424b-9523-64c937f26cef&bsft_mid=ba5bec3d-987b-4894-87ea-8f73066c96e5&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=fd07f62c-998f-4c97-bd8b-6284dd46d91a)

Meets Specifications

**Congratulations on Passing the Project!**

You've done a great job overall, Kudos! 👏

I've added an additional comment in the review section. Be sure to go through them.

You've done a great project. I can't wait to see what you do next!

Good Luck!

**All Required Files and Tests**

**The project submission contains the project notebook, called “dlnd\_tv\_script\_generation.ipynb”.**

Good Job! You have successfully included ipython notebook.

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

Congrats! All Unit test has​ been successfully 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).**

Great use of enumerate()

Here's another way to do it with more Python sauces:

counts = Counter(text)

vocab = sorted(counts, key=counts.get, reverse=True)

vocab\_to\_int = {word: i for i, word in enumerate(vocab, 1)}

int\_to\_vocab = {i: word for i, word in enumerate(vocab, 1)}

return vocab\_to\_int, int\_to\_vocab

Note: Neural networks have no understanding of the "meaning" of words but need to reference words through identifiers. This concept is also found in the different ways numbers can be used:

* A cardinal number tells "how many"
* An ordinal number indicates the order of things in a set
* A nominal number names something, such as player number in team

**The function token\_lookup returns a dict that can correctly tokenizes the provided symbols.**

Great, all the tokens are correctly added to the dict and a neat indentation!

**Batching Data**

**The function batch\_data breaks up word id's into the appropriate sequence lengths, such that only complete sequence lengths are constructed.**

Appropriate sequence length is provided to perform with the ​better result.

**In the function batch\_data, data is converted into Tensors and formatted with TensorDataset.**

Implemented data conversion into Tensors and has a correctly formatted with Tensor Dataset.

**Finally, batch\_data returns a DataLoader for the batched training data.**

Great work. You have properly return data\_loader value in batch\_data function

**Build the RNN**

**The RNN class has complete \_\_init\_\_, forward , and init\_hidden functions.**

Good Job. Includes all the required method in RNN class.

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

Proper utilization of LSTM function and has one fully connected layer.

**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.**

Successfully set appropriated values to all the hyperparameters included to train model.

Generally recommended values are:

Sequence Length: Chose a sequence length of 50 to give the network a context of approximately 10 script lines on average to consider during training. This can be derived from the fact that the average number of words per line would be around 5.5.

Epochs: Start out with 10 epochs as a first trial run to make sure the network would train and increased it to 20 thereafter. So you can have better look at loss value.

Batch Size: Initially start with 256 as an arbitrary start point and see that if the network was training efficiently if not then try to break it 128 for more batch training per epochs.

Embedding Dimension: the vocab contained ~46,367 unique words. Now you can try to cut this down significantly by 98% to 1000 embeddings.

Hidden Dimension: 128-256 hidden dimensions to give the network a solid amount of features/states to learn from.

n\_layers: Choosing 2-3 layers would have more benefits since you've learned in the coursework. Try to find out which works best

lr: Not too big to cause no convergence and not too small to cause slow convergence.

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

Congrats. This is the main step to pass the test which includes lowering the loss value at least up to​ 3.5.​ Keep it up.

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

Properly explained. Good work.

**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.**

Awesome! You have successfully generated TV script.

[**DOWNLOAD PROJECT**](https://review-api.udacity.com/api/v1/submissions/1685829/archive)

RETURN TO PATH