* [Logout](https://review.udacity.com/)

[Return to "Deep Learning" in the classroom](https://review.udacity.com/)

[DISCUSS ON STUDENT HUB](https://review.udacity.com/)

Generate TV Scripts

* [**REVIEW**](https://review.udacity.com/)
* [**CODE REVIEW**](https://review.udacity.com/)
* [**HISTORY**](https://review.udacity.com/)

Meets Specifications

You have done a tremendous job on the project!! Keep it up.

Few things you could try in your free time:

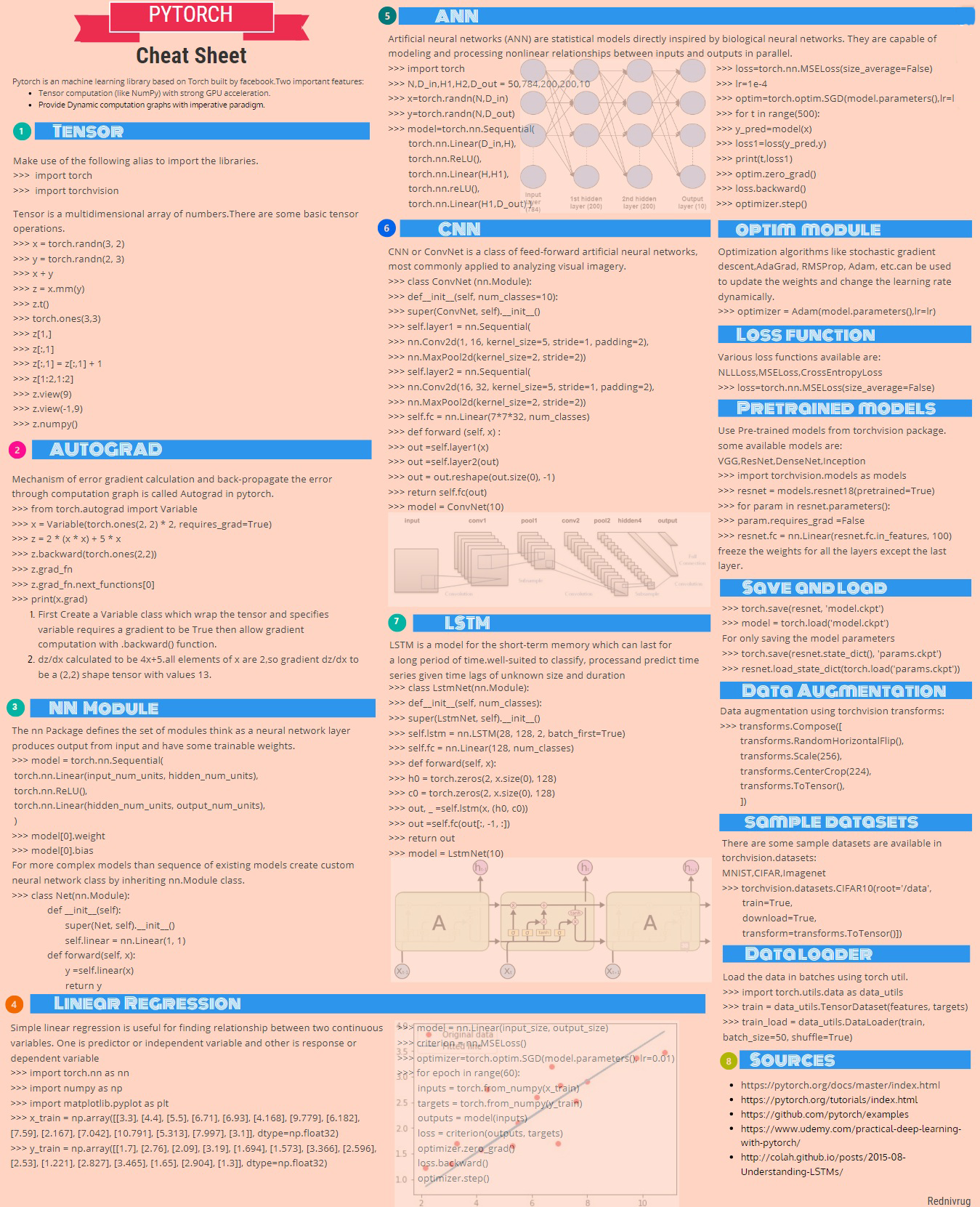
1. Play with seq length(Think of seq length as average sentence length in the training corpus)
2. Try with GRU and play with hyperparameters accordingly

Please find the pointers to interesting articles on applications of LSTM's

1. Language translation: <https://chunml.github.io/ChunML.github.io/project/Sequence-To-Sequence/>
2. Sequence tagging: <https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/>

and a link to the deep learning dictionary: <https://towardsdatascience.com/the-deep-learning-ai-dictionary-ade421df39e4>  
Good luck :)

PyTorch cheatsheet:

[](https://udacity-reviews-uploads.s3.us-west-2.amazonaws.com/_attachments/24010/1553188782/0.png)

**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).**

Good use of dictionary comprehensions.

**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.**

**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.**

In the batch data function,  
sequencing is done perfectly  
the code is clean and readable  
data is converted to tensors and  
Data loader is returned successfully.

**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.**

RNN class contains **init**, forward and init\_hidden functions which are implemented perfectly.  
LSTM layers are stacked and fully connected layers are used on top of it.  
Good work!

Further Resource: Keras has a neat API for visualizing the architecture, which is very helpful while debugging your network. pytorch-summary is a similar project in PyTorch.

**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.**

Perfect choice of hyperparameters. Nothing negative to call out here.

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

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

Nice to see that you have played with different hyperparameter settings and you were able to get the optimal hyperparameters and meet the project requirements.

**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.**

Generated script looks good to me.

[**DOWNLOAD PROJECT**](https://review-api.udacity.com/api/v1/submissions/1759942/archive)

RETURN TO PATH