UDACITY

PROJECT

Translation From One Language to Another Language

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Your code is awesome!
It shows you really understand RNNs & Tensorflow well and have put considerable effort into this.

Required Files and Tests

The project submission contains the project notebook, called "dlnd_language_translation.ipynb".

The `iPythonNB` and helper files are included.

All the unit tests in project have passed.

Great work. Unit testing is one of the most reliable methods to ensure that your code is free from all bugs without getting confused with the interactions with all the other code. If you are interested, you can read up more and I hope that you will continue to use unit testing in every module that you write to keep it clean and speed up your development.

But always keep in mind, that unit tests cannot catch every issue in the code. So your code could have bugs even though unit tests pass.

Preprocessing

The function `text_to_ids` is implemented correctly.

Clean and concise.
Both source and target are correct mapped to their respective mappings.
Good job using `EOS ID` only at end of target which will help decoder **stop** translating.

Neural Network

The function `model_inputs` is implemented correctly.

Correct.
Placeholders is the building block in computation graph of any neural net (especially in **tensorflow**).

Often I find students confused between `tf.Variable` and `tf.placeholder`. This answer gives correct usecase for both.

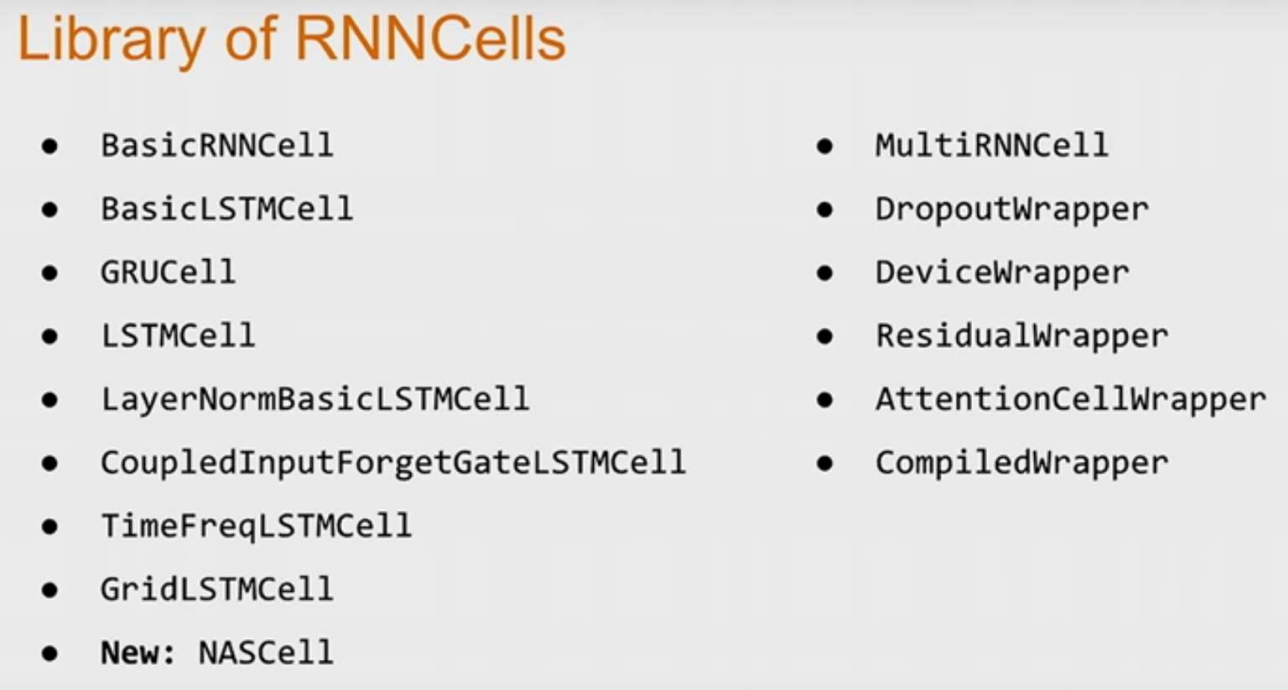
The function `process_decoding_input` is implemented correctly.

Good job concatenating the `GO ID` to the beginning of each batch. ✌
Effectively manipulating elements computational graph is one of the most important skills to acquire building deep models. And you are handling `tf` tensors amazingly.

The function `encoding_layer` is implemented correctly.

Your Encoder RNN layer is implemented perfectly.
`tf.nn.dynamic_rnn` is perfect for variable length time sequences as in our case of translations models. This is more or less the default choice when building seq-to-seq RNN architectures.

Further, there are a lot of other options when choosing `RNNCells` which you must be aware of.



The function `decoding_layer_train` is implemented correctly.

The function `decoding_layer_infer` is implemented correctly.

The function `decoding_layer` is implemented correctly.

Defining scope using `tf.variable_scope` to share variables between training and inference was very important for this part.
Awesome job using both train and inference perfectly.

Pro Tip: When building such systems at scale, the hidden state connecting encoder and decoder is often the bottleneck for computation. **Attention** is an important concept which comes at rescue. This talk by a Google developer and this awesome article on seq2seq and attention is perfect to understand how it works.

The function `seq2seq_model` is implemented correctly.

Congrats on placing all the lego blocks together ! You correctly built a Neural translation system :)

Pro Tip: Google recently released a general-purpose encoder-decoder framework for Tensorflow that can be used for Machine Translation, Text Summarization, Conversational Modeling, Image Captioning, and more. Check out the blog post and Git repo.

Neural Network Training

The parameters are set to reasonable numbers.

20 `epochs` with around 0.005 `learning_rate` took you places. The hyperparams chosen are very good as evident from your training loss.
Further, setting `batch_size` as a power of 2 (256 in your case) is handled efficiently by tensorflow (better so on GPU).

Everything worked perfectly with these setting of hyperparams. Seems to me, you had your **Deep learning hat** on while choosing these. 😉

The project should end with a validation and test accuracy that is at least 90.00%

Language Translation

The function `sentence_to_seq` is implemented correctly.

The project gets majority of the translation correctly. The translation doesn't have to be perfect.

Woah ! Now that is something 👍
This small model is performing too close to Google Translate. Wonder what will happen when trained on an even bigger corpus of data ?

Fun Link: Google Translate need to translate between a number of different pairs, both from A -> B and B -> A. The approach used in assignment will train a number of different models for each different pair of languages. What if we could train a net to learn linguistic mapping between two pair without explicitly training on this pair. This is known as Zero-shot learning in Deep Learning community. Read this to understand how Google accomplishes this in it's translation system.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ