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

Great job with code implementation!
My congratulation - you have passed this project!
Good luck in studying!

---

p.s.
In the next project you will generate human faces with deep learning. Here you can see last achievements in this area https://medium.com/intuitionmachine/these-images-are-generated-by-a-deep-learning-gan-6b49062b3959
I think it is great!

Required Files and Tests

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

All the unit tests in project have passed.

Awesome!
All the code snippets and unit tests are running perfectly.

Preprocessing

The function `text_to_ids` is implemented correctly.

Great job with pythonic way of function implementation!

Neural Network

The function `model_inputs` is implemented correctly.

The function `process_decoding_input` is implemented correctly.

The function `encoding_layer` is implemented correctly.

Awesome!

You can also add a dropout at this step if you will see that model overfits.
here is some example how you can use dropout:

- https://danijar.com/introduction-to-recurrent-networks-in-tensorflow/

Here is some good explanations of what is encoder layer:

- https://www.quora.com/What-is-an-Encoder-Decoder-in-Deep-Learning
- https://www.youtube.com/watch?v=FzS3tMl4Nsc

> Autoencoders are networks, which try to reconstruct their own input. **You construct the network so that it reduces the input size** by using one or more hidden layers, until it reaches a reasonably small hidden layer in the middle. As a result **your data has been compressed (encoded) into a few variables**. From this hidden representation the network tries to reconstruct (decode) the input again.

The function `decoding_layer_train` is implemented correctly.

Awesome!

You can also add a dropout at this step.

```
helper = tf.contrib.seq2seq.TrainingHelper(dec_embed_input, target_sequence_length)



    # decoder
    dropout_wrapper = tf.contrib.rnn.DropoutWrapper(dec_cell, output_keep_prob=keep_prob)
    decoder = tf.contrib.seq2seq.BasicDecoder(
        dropout_wrapper, helper, encoder_state,
        output_layer=output_layer)
    # dynamic decoding
    decoder_output, _ = tf.contrib.seq2seq.dynamic_decode(decoder, impute_finished=True, maximum_iterations=max_summary_length)
```

The function `decoding_layer_infer` is implemented correctly.

Well done!

Here we shouldn't use a dropout. It is a prediction/inference step. So we don't need to use dropout at the prediction step.

Here is good explanation of What's the Difference Between Deep Learning Training and Inference?

The function `decoding_layer` is implemented correctly.

The function `seq2seq_model` is implemented correctly.

Neural Network Training

The parameters are set to reasonable numbers.

Good choice hyperparameters!

- **batch_size** - Large batch sizes increase training speed, but can degrade the quality of the model. You can read this very useful discussion about this:
  http://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network
  Best results in this model can be achieved with 128 / 256 batch size.
- **rnn_size** - basically it is a the number of units in your LSTM cell. This number should be large enough that the network can generalize and avoid high bias (underfitting) and on the other hand it shouldn't be too large and shouldn't create a high variance (overfitting) model. Here is good explanation of bias-variance trade-off: http://scott.fortmann-roe.com/docs/BiasVariance.html
- **num_layers** - You've made a great choice here! Our dataset is small here so we don't have reasons to use large number of layers.
- **embedding_size** - Basically embedding sizes should be large enough that the model has capacity to learn. Here the size of our vocabulary is 227 words. So 128 / 256 is an ideal number!

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.

You have quite good translation!

- *he saw a old yellow truck ->*
- *il une un vieux camion ->*
- *he's an old truck* (according google translation)

I think it can become better if you will reduce a batch size.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ