UDACITY Logout

PROJECT

Translation From One Language to Another Language

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Required Files and Tests

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

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

Great work. Unit testing is very good practice to ensure that your code is free from all bugs without getting confused and prevent you from wasting a lot of time while debugging minor things. Unit test also help in improving our code standards. For more details read this and I really hope that you will continue to use unit testing in every module that you write to keep it clean and speed up your development and for quality development. Unit testing is highly motivated in industries.

It is not always that if you passed unit test your code is okay, there can be some errors.

Preprocessing

✓ **The function `text_to_ids` is implemented correctly.**

Good Work!! Correctly implemented.

Neural Network

✓ **The function `model_inputs` is implemented correctly.**

Good Work!!

(Following abstract is from Tensorflow documentation)
TensorFlow programs use a tensor data structure to represent all data -- only tensors are passed between operations in the computation graph. You can think of a TensorFlow tensor as an n-dimensional array or list. A tensor has a static type, a rank, and a shape.

In the TensorFlow system, tensors are described by a unit of dimensionality known as rank. Tensor rank is not the same as matrix rank. Tensor rank (sometimes referred to as order or degree or n-dimension) is the number of dimensions of the tensor. For example, the following tensor (defined as a Python list) has a rank of 2:

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

A rank two tensor is what we typically think of as a matrix, a rank one tensor is a vector. For a rank two tensor you can access any element with the syntax t[i, j]. For a rank three tensor you would need to address an element with t[i, j, k].

Link! that might help you with better understanding of rank in Tensor.

✓ **The function `process_decoding_input` is implemented correctly.**

Good Work!!
Removed the last word id from each batch in target_data and concatenated the GO ID to the beginning of each batch.

✓ **The function `encoding_layer` is implemented correctly.**

Good Work!! I would strongly recommend use of dropout here.

What we understand by encoding? I read this definition from internet by Adam Geitgey and found very easy to understand.
When you are trying to tell two faces apart with a computer, you collect different measurements from each face and use those measurements to compare faces. For example, we might measure the size of each ear or the spacing between the eyes and compare those measurements from two pictures to see if they are the same person. The idea of turning a face into a list of measurements is an example of an encoding. Similary we make same encodding for our sentences and our first network helps us to do so.

We discard encoder_outputs because we are not interested in them within seq2seq framework. What we actually want is encoder_final_state — state of LSTM's hidden cells at the last moment of the Encoder rollout. encoder_final_state is also called "thought vector". We will use it as initial state for the Decoder. In seq2seq without attention this is the only point where Encoder passes information to Decoder. We hope that backpropagation through time (BPTT) algorithm will tune the model to pass enough information throught the thought vector for correct sequence output decoding.

AutoEncoders Link: https://www.youtube.com/watch?v=FzS3tMl4Nsc

✓ **The function `decoding_layer_train` is implemented correctly.**

Good Work!! Correctly implemented. I would strongly recommend use of dropout here.

✓ **The function `decoding_layer_infer` is implemented correctly.**

Good Work!! Correctly implemented.

✓ **The function `decoding_layer` is implemented correctly.**

Good Work!! Correctly implemented.

How decoder works, Intuitively?
Answer In the decoder step, a language model is trained on both the output sequence (such as the translated sentence) as well as the fixed representation from the encoder. Since the decoder model sees an encoded representation of the input sequence as well as the translation sequence, it can make more intelligent predictions about future words based on the current word. For example, in a standard language model, we might see the word “crane” and not be sure if the next word should be about the bird or heavy machinery. However, if we also pass an encoder context, the decoder might realize that the input sequence was about construction, not flying animals. Given the context, the decoder can choose the appropriate next word and provide more accurate translations.

Reference

✓ **The function `seq2seq_model` is implemented correctly.**

Good Work!!
But I would recommend not to use dropout here on `infer_logits`.

Neural Network Training

✓ **The parameters are set to reasonable numbers.**

Good choice of Hyper Parameters.

Some Tips on how to choose Hyper Parameters

- **rnn_size** : For each LSTM cell that we initialise, we need to supply a value for the hidden dimension(`rnn_size`), or as some people like to call it, the number of units in the LSTM cell. The value of it is it up to you, too high a value may lead to overfitting or a very low value may yield extremely poor results.
- **batch_size** : Try to observe what happens if you lower it or increase it. Larger batch sizes speed up the training but can degrade the quality of the model. Here is a useful resource to understand this better - http://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network
- **Learning Rate** : Learning rate is the most important hyperparameter, therefore it is very important to understand how to set it correctly in order to achieve good performance. A related but unsurprising observation is that there is a sweet-spot for the learning rate at the high end of the basin. In this region, the performance is good and the training time is small. So while searching for a good learning rate for the LSTM, it is sufficient to do a coarse search by starting with a high value (e.g. 1.0) and dividing it by ten until performance stops increasing.
- **Hidden Layers** : As expected, larger networks perform better, and the required training time increases with the network size.
- **Embedding size** : I've actually found smaller vectors to work better (64 and 128 - note the binary sizes, I think this helps performance by assisting theano with copying chunks of data to and from GPU), likely as that's fewer parameters to learn, and too many can make learning hard for a neural network. Cross validation would help you determine a good size, try varying in magnitude (32,64,128,256) rather than more linear scales, as the relationship between these items and performance tend to be more of an exponential than a linear nature.

Reference Link!

Link that would help in fine tune model.
http://neupy.com/2016/12/17/hyperparameter_optimization_for_neural_networks.html

✓ **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.**

Good Work!! You took care of converting sentence into lower case.

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

Result is decent and your model is understanding the concept as well as context. Good work!!

Now you should consider training your network on larger dataset to help your model work better in general case.
A very good open-source dataset is: Link to a good dataset.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ