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!

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

This project is one of the toughest in the course, the model is state of the art and a version of it is being used everyday in google translate. Also please note these projects will be part of your portfolio projects from this course, so make sure you get the best of out of them. You are on the right track here, hyper parameter tuning is crucial part, with a bit of tuning you will get an awesome model! The core of the seq2seq model is the encoder and decoder layer. There is a reason why we consider the choice for encoder and decoder embeddings crucial, to understand why lets go a bit in detail about them:

- ## Encoder

  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. Similarly we make same encoding 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 through the

thought vector for correct sequence output decoding.
AutoEncoders Link: https://www.youtube.com/watch?v=FzS3tMl4Nsc

- # Decoder

  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.

Hope this helps!

All the best in your deep learning journey! 👋🏻

Required Files and Tests

The project submission contains the project notebook, called “dlnd_language_translation.ipynb”.

All the unit tests in project have passed.

Preprocessing

The function `text_to_ids` is implemented correctly.

Nicely done.
But a more pythonic way of implementing this function is to use list comprehension:

```
source_sentences = [sentence for sentence in source_text.split('\n')]
target_sentences = [sentence + ' <EOS>' for sentence in target_text.split(
'\n')]
source_id_text= [[source_vocab_to_int[word] for word in sentence.split()] for
 sentence in source_sentences]
target_id_text = [[target_vocab_to_int[word] for word in sentence.split()] fo
r sentence in target_sentences]
```

You will find that using list comprehensions are a bit more faster and efficient in memory than using explicit loops. 👍

Neural Network

The function `model_inputs` is implemented correctly.

The function `process_decoding_input` is implemented correctly.

A proper way to do is to use tf.strided_slice to remove the last word id.

The function `encoding_layer` is implemented correctly.

Good job using dropouts here! 👍

The function `decoding_layer_train` is implemented correctly.

Good effort here!
Please add impute_finished = True to the dynamic decode

The function `decoding_layer_infer` is implemented correctly.

Good effort here!
Please add impute_finished = True to the dynamic decode

The function `decoding_layer` is implemented correctly.

Suggestion

You have used a dropout here to define a decoder cell, this will be really useful for regularisation in the training layer but when the same passes through the inference layer this will lead to loss of information, however you wont notice a difference here because the default keep_prob here is set to 1 and that makes the dropout

redundant, but for making your code reusable and scalable please do remove the dropouts here and try to incorporate dropouts in the decoding_layer_train and encoding_layer
Here is good explanation of What's the Difference Between Deep Learning Training and Inference?

Try using reuse_variables() function with the decoding_scope instead of redundantly making context managers

```
with tf.variable_scope("decode") as decoding_scope:
         train_decoder_out = decoding_layer_train(...)
         decoding_scope.reuse_variables()
         infer_decoder_out= decoding_layer_infer(...)
```

Please do make sure that the reuse_variable() function and the consecutive code is indented to be at the same level as the previous context manager.Please do read more about Sharing Variables in Tf Also the output layer can be created in the same context manager.

The function `seq2seq_model` is implemented correctly.

Neural Network Training

The parameters are set to reasonable numbers.

You need to be careful with your hyper parameters selection here, Basically embedding sizes should be large enough that the model has capacity to learn. Here the size of our vocabulary is 227 words. The embedding size is what defines the amount of information that gets encoded and decoded from the sentences. So make sure this is around ~128-227. Your data has a vocabulary of 227 words and when you encode information from 227 words to 256 dimensions you tend to add noise to the model, so please do be careful here.

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

Awesome job getting over 90% accuracy here! 👏

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.
Your model translations are spot on! 😎

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review