UDACITY

PROJECT

Generate TV Scripts

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

4 SPECIFICATIONS REQUIRE CHANGES

Good job !
You just have some minor correction to do, and after it will be perfect.

Required Files and Tests

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

Good all files are here

All the unit tests in project have passed.

Good

Preprocessing

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 in the a tuple (vocab_to_int, int_to_vocab)

Good, your implementation did the job.
Here another example of solution:

```
word = set(text)
vocab_to_int = { c: i for i, c in enumerate(word)}
int_to_vocab = dict(enumerate(word))
```

The function `token_lookup` returns a dict that can correctly tokenizes the provided symbols.

Build the Neural Network

Implemented the `get_inputs` function to create TF Placeholders for the Neural Network with the following placeholders:

- Input text placeholder named "input" using the TF Placeholder name parameter.
- Targets placeholder
- Learning Rate placeholder

The `get_inputs` function return the placeholders in the following the tuple (Input, Targets, LearingRate)

Good, but I would suggest to avoid using name 'input' for variable, as you override a core function here. Use _input instead, for example.

The `get_init_cell` function does the following:

- Stacks one or more BasicLSTMCells in a MultiRNNCell using the RNN size `rnn_size`.
- Initializes Cell State using the MultiRNNCell's `zero_state` function
- The name "initial_state" is applied to the initial state.
- The `get_init_cell` function return the cell and initial state in the following tuple (Cell, InitialState)

Good, but you shouldn't hard coded the size of the lstm, this is rnn_size. Also, the number of layer you use isn't rnn_size but a variable that you have to add

The function `get_embed` applies embedding to `input_data` and returns embedded sequence.

The function `build_rnn` does the following:

- Builds the RNN using the `tf.nn.dynamic_rnn`.
- Applies the name "final_state" to the final state.
- Returns the outputs and final_state state in the following tuple (Outputs, FinalState)

The `build_nn` function does the following in order:

- Apply embedding to `input_data` using `get_embed` function.
- Build RNN using cell using `build_rnn` function.
- Apply a fully connected layer with a linear activation and `vocab_size` as the number of outputs.
- Return the logits and final state in the following tuple (Logits, FinalState)

Again, the embed_dim shouldn't be hard coded, but must be the rnn_size.

The `get_batches` function create batches of input and targets using `int_text`. The batches should be a Numpy array of tuples. Each tuple is (batch of input, batch of target).

- The first element in the tuple is a single batch of input with the shape [batch size, sequence length]
- The second element in the tuple is a single batch of targets with the shape [batch size, sequence length]

Neural Network 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.
- Size of the RNN cells (number of units in the hidden layers) is large enough to fit the data well. Again, no real “best” value.
- The sequence length (seq_length) here should be about the size of the length of sentences you want to generate. Should match the structure of the data.
  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.
  Set show_every_n_batches to the number of batches the neural network should print progress.

Your rnn_size is too small, but that normal due to your previous implementation.

Here some tips according to your new implementation. Even if there isn't one good set of values, I would advice to use range of value due to the fact that this dataset is small, and a dialogue are most of the time also small.

- num_epochs = [20, 400]

- batch_size = [25, 256] : Depend mainly on the power of your computer ;)
- rnn_size = [25, 512] : Big means more complex structure of the dialogue to learn.
- seq_length = [5, 50] : As each speech are not very long, this should represent the mean length of sentence.
- learning_rate = [0.001, 0.01] : Low to understand the structure of the text
  The values of hyper parameters should be power of 2. Tensorflow optimizes our computation if we do so.

The project gets a loss less than 1.0

Generate TV Script

"input:0", "initial_state:0", "final_state:0", and "probs:0" are all returned by `get_tensor_by_name`, in that order, and in a tuple

The `pick_word` function predicts the next word correctly.

You should add randomness on this process to avoid potential repetitive words.

The generated script looks similar to the TV script in the dataset.

It doesn't have to be grammatically correct or make sense.

While I had remarks on your implementation, the current result is good.

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Rate this review

Student FAQ