UDACITY Logout

‹ Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

Generate TV Scripts

REVIEW CODE REVIEW HISTORY

Meets Specifications

Very impressive submission ! I can see your hard work reflected in your project 🏆 congratulations on achieving this and good luck in your way to master deep learning 😃

Advanced tips:

1. When you want to preprocess a text:
   To understand more what you need to do with your data before using it on your model, here a good link:
   http://datascience.stackexchange.com/questions/11402/preprocessing-text-before-use-rnn
2. Word representation:
   When you have to represent words for a true deeplearning model, you should think deeper to close words, or words with almost the same spelling but with a highly different meaning. From this statement word2vec try to solve this issue. Here a good link to explain it:
   https://www.tensorflow.org/tutorials/word2vec
   Another ressources: https://www.oreilly.com/learning/capturing-semantic-meanings-using-deep-learning
3. Good articles :
   http://karpathy.github.io/2015/05/21/rnn-effectiveness/
   http://colah.github.io/posts/2015-08-Understanding-LSTMs/
4. More advanced about recurrent network, Attention and Augmented Recurrent Neural Networks:
   http://distill.pub/2016/augmented-rnns/

All Required Files and Tests

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

All the unit tests in project have passed.

Pre-processing Data

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 as a tuple (vocab_to_int, int_to_vocab).

Fantastic job passing all the tests 👏

Good!

Your implementation did the job.
Here another example of solution, without sort:

```
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.

This is perfect! Please read up this link to understand what other pre-processing steps are carried out before feeding text data to RNNs

Batching Data

The function `batch_data` breaks up word id's into the appropriate sequence lengths, such that only complete sequence lengths are constructed.

✓

In the function `batch_data`, data is converted into Tensors and formatted with TensorDataset.

✓

Finally, `batch_data` returns a DataLoader for the batched training data.

✓

Build the RNN

The RNN class has complete `__init__`, `forward` , and `init_hidden` functions.

Great Going! You've put everything together perfectly 😉

C Olah and Andrej are two researchers who explains these concepts wonderfully.

The RNN must include an LSTM or GRU and at least one fully-connected layer. The LSTM/GRU should be correctly initialized, where relevant.

Amazing job! this architecture will work great in thins problem. You can experiment a little bit with the hyper parameters, augmenting the size of the batches and check how can this affect the result in the training process. This will help you develop a greater intuition about sequential models.

RNN 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.
- Embedding dimension, significantly smaller than the size of the vocabulary, if you choose to use word embeddings
- Hidden dimension (number of units in the hidden layers of the RNN) is large enough to fit the data well. Again, no real "best" value.
- n_layers (number of layers in a GRU/LSTM) is between 1-3.
- The sequence length (seq_length) here should be about the size of the length of sentences you want to look at before you generate the next word.
- 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.

🚀

```
Enough epochs to get near a minimum in the training loss. Do not hesitate to use a value as big as needed till the loss the improving
Batch size is large enough to train efficiently
Sequence length is about the size of the length of sentences we want to generate
Size of embedding is in the range of [200-300]
Learning rate seems good based on other hyper parameter

Your efforts shows that you have really have executed it again and again to get an optimized value
```

The printed loss should decrease during training. The loss should reach a value lower than 3.5.

Great! this si an indication that your model is doing a good job 😉

There is a provided answer that justifies choices about model size, sequence length, and other parameters.

Generate TV Script

The generated script can vary in length, and should look structurally similar to the TV script in the dataset.

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

Rate this review

DOWNLOAD PROJECT

Rate this review