UDACITY

Return to Classroom

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
2. Word representation: When you have to represent words for a true deep learning 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 tries to solve this issue.
3. Good blogs : Andrej Karpathy and Christopher Olah
4. More advanced about recurrent network, Attention and Augmented Recurrent Neural Networks

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.

All the unit tests in project have passed.

Fantastic job passing all the tests 👏

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).

Good!

Your implementation did the job.

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

All 10 entries are present !

Batching Data

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

Good Job 👏

- breaks up word id's into the appropriate sequence lengths
- It is recommend to add explanatory comments in between, for example :

```
 # get number of targets we can make
 n_targets = len(words) - sequence_length
 # initialize feature and target
 feature, target = [], []
 # loop through all targets we can make
 for i in range(n_targets):
     x = words[i : i+sequence_length]    # get some words from the gi
ven list
     y = words[i+sequence_length]        # get the next word to be th
e target
     feature.append(x)
```

```
        target.append(y)

    feature_tensor, target_tensor = torch.from_numpy(np.array(feature)),
torch.from_numpy(np.array(target))
    # create data
    data = TensorDataset(feature_tensor, target_tensor)
    # create dataloader
    dataloader = DataLoader(data, batch_size=batch_size, shuffle=True)
     # return a dataloader
    return dataloader
```

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.

- `__init__`, `forward` and `init_hidden` functions are complete

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

The ideal structure is as follows:

- Embedding layer (nn.Embedding) before the LSTM or GRU layer.
- The fully-connected layer comes at the end to get our desired number of outputs.
- Extra marks for not using a dropout after LSTM and before FC layer, as the drop out is already incorporated in the LSTMs, A lot of students will add it and then end up finding convergence difficult
- You can try to add more than one fc:

```
    # init
    self.fcc=nn.Linear(self.hidden_dim, self.hidden_dim)
    self.fcc2=nn.Linear(self.hidden_dim,self.output_size)
    ....
    # forward
```

```
    output,hidden=self.lstm(embedded,hidden)
    lstm_output = output.contiguous().view(-1, self.hidden_dim)
    output= self.fcc(lstm_output)
    output=self.dropout(output)
    output=self.fcc2(output)
```

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 hyperparameter
- 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 is 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.

This is very good, it's repeating a couple of words but ti's mostly fine. The state of the art for text generations are transformers, you might want to give a look to GPT3 for instance 😄

About your problems with dropout I think 0.5 is very aggressive, I will try a smaller amount and definitely not using at the output of the feed-forward output layer. Please contact our mentors for more details 😉

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START