UDACITY

Back to Artificial Intelligence Nanodegree and Specializations

Machine Translation

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Your project demonstrates that you have fully understood the concepts and some maturity in this technology.

Unfortunately, your model does not produce the right translations yet.

There are only a few changes that need to be performed to improve the `final_model` accuracy.

Please have a look at my comments about your final model, change your code accordingly and resubmit your project.

In case of doubts, do not hesitate to contact your mentor or open a new question in the forums.

Good luck with your next project submission!

Submitted Files

The following files have been submitted: **`helper.py`**, **`machine_translation.ipynb`**, **`machine_translation.html`**

Preprocess

The function `tokenize` returns tokenized input and the tokenized class.

Your `tokenize` function is correctly implemented. Well done!

The function `pad` returns padded input to the correct length.

Great job here! Your `pad` function works flawlessly!

Models

The function `simple_model` builds a basic RNN model.

You have successfully implemented `simple_model`. Well done!

There are multiple solutions to this part of the project. Below, I have copied an alternative version just in case you would like to experiment with it:

```
            def simple_model(input_shape, output_sequence_length, english_voc
ab_size, french_vocab_size):
                """
                Build and train a basic RNN on x and y
                :param input_shape: Tuple of input shape
                :param output_sequence_length: Length of output sequence
                :param english_vocab_size: Number of unique English words in
 the dataset
                :param french_vocab_size: Number of unique French words in th
e dataset
                :return: Keras model built, but not trained
                """

                learning_rate = 0.005
                model = Sequential()
                model.add(LSTM(100, input_shape = input_shape[1:], dropout =
0.2, return_sequences = True))
                model.add(Dense(french_vocab_size))
                model.add(Activation('softmax'))
                model.compile(loss=sparse_categorical_crossentropy,
```

```
                            optimizer=Adam(learning_rate),
                            metrics=['accuracy'])
                return model
```

Hope it helps!

The function `embed_model` builds a RNN model using word embedding.

Your `embed_model` is also implemented correctly. Well done!

You could try to increase the number of epochs to get a better accuracy.

I have also copied an alternative version of this model which performs quite well. I do encourage you to give it a try!

```
            def embed_model(input_shape, output_sequence_length, english_voca
b_size, french_vocab_size):
                """
                Build and train a RNN model using word embedding on x and y
                :param input_shape: Tuple of input shape
                :param output_sequence_length: Length of output sequence
                :param english_vocab_size: Number of unique English words in
 the dataset
                :param french_vocab_size: Number of unique French words in th
e dataset
                :return: Keras model built, but not trained
                """

                learning_rate = 0.005
                model = Sequential()
                model.add(Embedding(english_vocab_size, 100, input_shape = in
put_shape[1:]))
                model.add(LSTM(100, dropout = 0.2, return_sequences = True))
                model.add(Dense(french_vocab_size))
                model.add(Activation('softmax'))
                model.compile(loss=sparse_categorical_crossentropy,
                            optimizer=Adam(learning_rate),
                            metrics=['accuracy'])
                return model
```

The Embedding RNN is trained on the dataset. A prediction using the model on the training dataset is printed in the notebook.

The function `bd_model` builds a bidirectional RNN model.

Well done here!

It is normal that the accuracy decreases compared to the previous model.

In the next sections, we will see the power of these layers when combined together.

The Bidirectional RNN is trained on the dataset. A prediction using the model on the training dataset is printed in the notebook.

The function `model_final` builds and trains a model that incorporates embedding, and bidirectional RNN using the dataset.

Your model_final is correctly implemented.

It is only needed a few changes to make it accurate enough to correctly translate both sentences.

Please find here a few things you may try in your model to increase its accuracy!

- Decrease the learning rate to 0.001
- Increase the number of epochs from 10 to 20-30.
- Increase units to 512
- Add a hidden Dense layer with 2*french_vocab_size units
- Use LSTM layers instead of GRU layers. <-- I would leave this change to the last. In this project, GRU layers usually perform better than LSTM layers. Nevertheless, it depends on your final model architecture.

Please notice that I would not perform all these changes at the same time. I would try one by one, noticing which effect has in our model accuracy.

Prediction

The final model correctly predicts both sentences.

You almost got it. You have got a good accuracy, but it is not enough yet.

Please, follow my comments in the previous section and I am pretty sure that you will get both sentences perfectly translated.

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

Student FAQ