UDACITY

‹ Return to "Artificial Intelligence Nanodegree and Specializations" in the classroom

Machine Translation

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations for passing Machine Translation project 🎉

Stay Udacious!

Submitted Files

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

All files are included in the submission.

Preprocess

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

Good job implementing `tokenize` function that returns sequence and tokenizer 👍

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

Good job adding `pads` at the end of sequences 👍

Models

The function `simple_model` builds a basic RNN model.

Well done! All hyperparameter settings are optimized.

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

Embedding layer is correctly implemented with good output vector.

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.

Good to see you provide functional and sequential implementation with this `bd_model`.

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.

Good job for adding `Embedding`, `Bidirectional`, and `Encoder-Decoder` layers in your `model_final`. 👍

You can also implement with Sequential model as follows:

```
model = Sequential()
model.add(Embedding(input_dim=english_vocab_size,output_dim=128,input_length=input_s
hape[1]))
model.add(Bidirectional(GRU(256,return_sequences=False)))
model.add(RepeatVector(output_sequence_length))
model.add(Bidirectional(GRU(256,return_sequences=True)))
model.add(TimeDistributed(Dense(french_vocab_size,activation='softmax')))

learning_rate = 0.001
model.compile(loss=sparse_categorical_crossentropy,
                  optimizer=Adam(learning_rate),
                  metrics=['accuracy'])
return model
```

Prediction

The final model correctly predicts both sentences.

Great job for getting perfect translations on both sentences and achieving 97.02% accuracy score 👏

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review