# How to Develop a Seq2Seq Model for Neural Machine Translation in Keras

by **Jason Brownlee** on August 7, 2019 in **Deep Learning for Natural Language Processing** 234

Share Tweet Share

The encoder-decoder model provides a pattern for using recurrent neural networks to address challenging sequence-to-sequence prediction problems, such as machine translation.

Encoder-decoder models can be developed in the Keras Python deep learning library and an example of a neural machine translation system developed with this model has been described on the Keras blog, with sample code distributed with the Keras project.

In this post, you will discover how to define an encoder-decoder sequence-to-sequence prediction model for machine translation, as described by the author of the Keras deep learning library.

After reading this post, you will know:

- The neural machine translation example provided with Keras and described on the Keras blog.
- How to correctly define an encoder-decoder LSTM for training a neural machine translation model.
- How to correctly define an inference model for using a trained encoder-decoder model to translate new sequences.

**Kick-start your project** with my new book Deep Learning for Natural Language Processing, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let’s get started.

- **Update Apr/2018**: For an example applying this complex model, see the post: How to Develop an Encoder-Decoder Model for Sequence-to-Sequence Prediction in Keras

How to Define an Encoder-Decoder Sequence-to-Sequence Model for Neural Machine Translation in Keras
Photo by Tom Lee, some rights reserved.

## Sequence-to-Sequence Prediction in Keras

Francois Chollet, the author of the Keras deep learning library, recently released a blog post that steps through a code example for developing an encoder-decoder LSTM for sequence-to-sequence prediction titled "A ten-minute introduction to sequence-to-sequence learning in Keras".

The code developed in the blog post has also been added to Keras as an example in the file lstm_seq2seq.py.

The post develops a sophisticated implementation of the encoder-decoder LSTM as described in the canonical papers on the topic:

- Sequence to Sequence Learning with Neural Networks, 2014.
- Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

The model is applied to the problem of machine translation, the same as the source papers in which the approach was first described. Technically, the model is a neural machine translation model.

Francois' implementation provides a template for how sequence-to-sequence prediction can be implemented (correctly) in the Keras deep learning library at the time of writing.

You will be able to use this understanding to develop similar models for your own sequence-to-sequence prediction problems.

## Machine Translation Data

The dataset used in the example involves short French and English sentence pairs used in the flash card software Anki.

The dataset is called "Tab-delimited Bilingual Sentence Pairs" and is part of the Tatoeba Project and listed on the ManyThings.org site for helping English as a Second Language students.

The dataset used in the tutorial can be downloaded from here:

- French – English fra-eng.zip

Below is a sample of the first 10 rows from the *fra.txt* data file you will see after you unzip the downloaded archive.

```
Go. Va !
Run! Cours !
Run! Courez !
Wow! Ça alors !
Fire! Au feu !
Help! À l'aide !
Jump. Saute.
Stop! Ça suffit !
Stop! Stop !
Stop! Arrête-toi !
```

The problem is framed as a sequence prediction problem where input sequences of characters are in English and output sequences of characters are in French.

- **Input Sequences**: Padded to a maximum length of 16 characters with a vocabulary of 71 different characters (10000, 16, 71).
- **Output Sequences**: Padded to a maximum length of 59 characters with a vocabulary of 93 different characters (10000, 59, 93).

The training data is framed such that the input for the model is comprised of one whole input sequence of English characters and the whole output sequence of French characters. The output of the model is the whole sequence of French characters, but offset forward by one time step.

For example (with minimal padding and without one-hot encoding):

- Input1: [‘G’, ‘o’, ‘.’, ”]
- Input2: [ ”, ‘V’, ‘a’, ‘ ‘]
- Output: [‘V’, ‘a’, ‘ ‘, ‘!’]

## Machine Translation Model

The neural translation model is an encoder-decoder recurrent neural network.

It is comprised of an encoder that reads a variable length input sequence and a decoder that predicts a variable length output sequence.

In this section, we will step through each element of the model’s definition, with code taken directly from the post and the code example in the Keras project (at the time of writing).

The model is divided into two sub-models: the encoder responsible for outputting a fixed-length encoding of the input English sequence, and the decoder responsible for predicting the output sequence, one character per output time step.

The first step is to define the encoder.

The input to the encoder is a sequence of characters, each encoded as one-hot vectors with length of *num_encoder_tokens*.

The LSTM layer in the encoder is defined with the *return_state* argument set to *True*. This returns the hidden state output returned by LSTM layers generally, as well as the hidden and cell state for all cells in the layer. These are used when defining the decoder.

```
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
```

The decoder input is defined as a sequence of French character one-hot encoded to binary vectors with a length of *num_decoder_tokens*.

The LSTM layer is defined to both return sequences and state. The final hidden and cell states are ignored and only the output sequence of hidden states is referenced.

Importantly, the final hidden and cell state from the encoder is used to initialize the state of the decoder. This means every time that the encoder model encodes an input sequence, the final internal states of the encoder model are used as the starting point for outputting the first character in the output sequence. This also means that the encoder and decoder LSTM layers must have the same number of cells, in this case, 256.

A *Dense* output layer is used to predict each character. This *Dense* is used to produce each character in the output sequence in a one-shot manner, rather than recursively, at least during training. This is because the entire target sequence required for input to the model is known during training.

The Dense does not need to be wrapped in a *TimeDistributed* layer.

```
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

Finally, the model is defined with inputs for the encoder and the decoder and the output target sequence.

```
# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

We can tie all of this together in a standalone example and fix the configuration and print a graph of the model. The complete code example for defining the model is listed below.

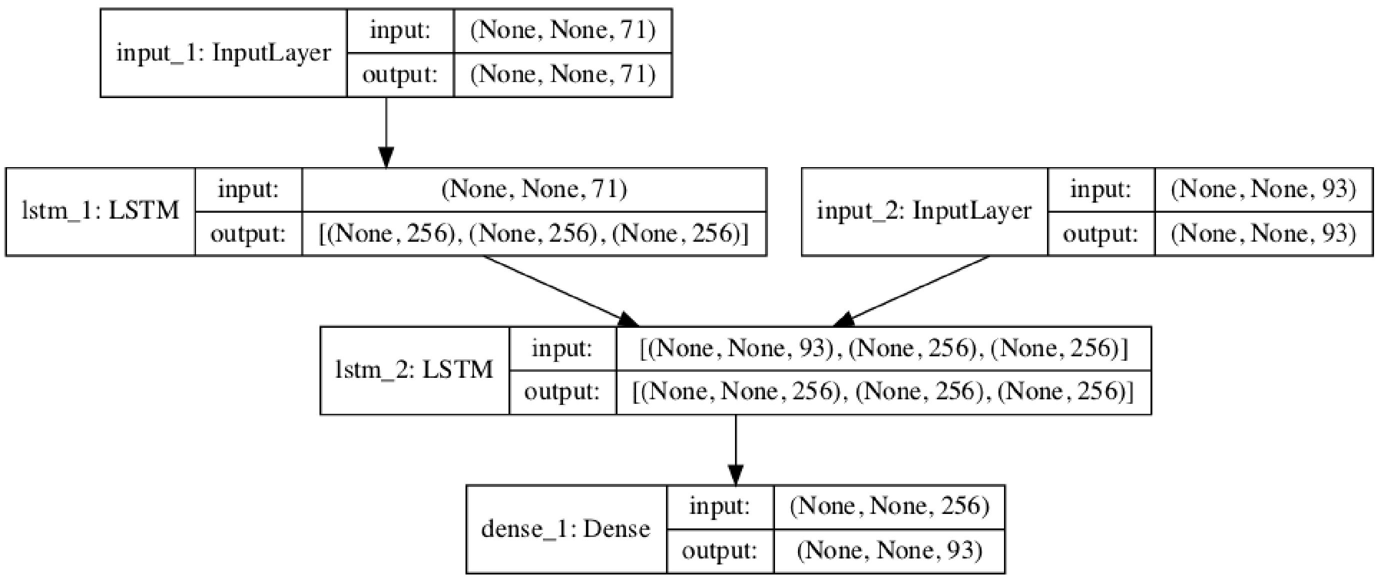
```
from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
from keras.layers import Dense
from keras.utils.vis_utils import plot_model
# configure
num_encoder_tokens = 71
num_decoder_tokens = 93
latent_dim = 256
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
# Set up the decoder, using `encoder_states` as initial state.
```

```
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
# plot the model
plot_model(model, to_file='model.png', show_shapes=True)
```

Running the example creates a plot of the defined model that may help you better understand how everything hangs together.

Note that the encoder LSTM does not directly pass its outputs as inputs to the decoder LSTM; as noted above, the decoder uses the final hidden and cell states as the initial state for the decoder.

Also note that the decoder LSTM only passes the sequence of hidden states to the Dense for output, not the final hidden and cell states as suggested by the output shape information.



Graph of Encoder-Decoder Model For Training

## Neural Machine Translation Inference

Once the defined model is fit, it can be used to make predictions. Specifically, output a French translation for an English source text.

The model defined for training has learned weights for this operation, but the structure of the model is not designed to be called recursively to generate one character at a time.

Instead, new models are required for the prediction step, specifically a model for encoding English input sequences of characters and a model that takes the sequence of French characters generated so far

Defining the inference models requires reference to elements of the model used for training in the example. Alternately, one could define a new model with the same shapes and load the weights from file.

The encoder model is defined as taking the input layer from the encoder in the trained model (*encoder_inputs*) and outputting the hidden and cell state tensors (*encoder_states*).

```
# define encoder inference model
encoder_model = Model(encoder_inputs, encoder_states)
```

The decoder is more elaborate.

The decoder requires the hidden and cell states from the encoder as the initial state of the newly defined encoder model. Because the decoder is a separate standalone model, these states will be provided as input to the model, and therefore must first be defined as inputs.

```
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
```

They can then be specified for use as the initial state of the decoder LSTM layer.

```
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(decoder_inputs, initial_state=decoder_states
```

Both the encoder and decoder will be called recursively for each character that is to be generated in the translated sequence.

On the first call, the hidden and cell states from the encoder will be used to initialize the decoder LSTM layer, provided as input to the model directly.

On subsequent recursive calls to the decoder, the last hidden and cell state must be provided to the model. These state values are already within the decoder; nevertheless, we must re-initialize the state on each call given the way that the model was defined in order to take the final states from the encoder on the first call.

Therefore, the decoder must output the hidden and cell states along with the predicted character on each call, so that these states can be assigned to a variable and used on each subsequent recursive call for a given input sequence of English text to be translated.

```
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model([decoder_inputs] + decoder_states_inputs, [decoder_outputs] + decoder_s
```

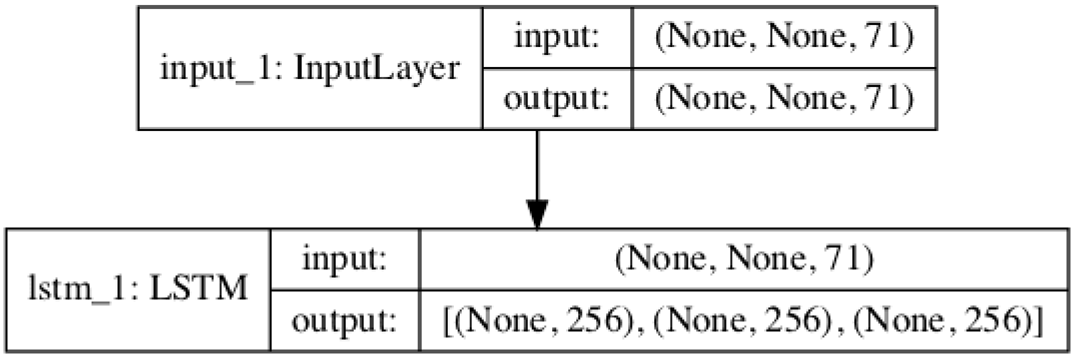
We can tie all of this together into a standalone code example combined with the definition of the training model of the previous section, given the reuse of some elements. The complete code listing is provided below.

```
from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
```

```
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
# plot the model
plot_model(model, to_file='model.png', show_shapes=True)
# define encoder inference model
encoder_model = Model(encoder_inputs, encoder_states)
# define decoder inference model
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(decoder_inputs, initial_state=decoder_state
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model([decoder_inputs] + decoder_states_inputs, [decoder_outputs] + decoder_
# summarize model
plot_model(encoder_model, to_file='encoder_model.png', show_shapes=True)
plot_model(decoder_model, to_file='decoder_model.png', show_shapes=True)
```

Running the example defines the training model, inference encoder, and inference decoder.

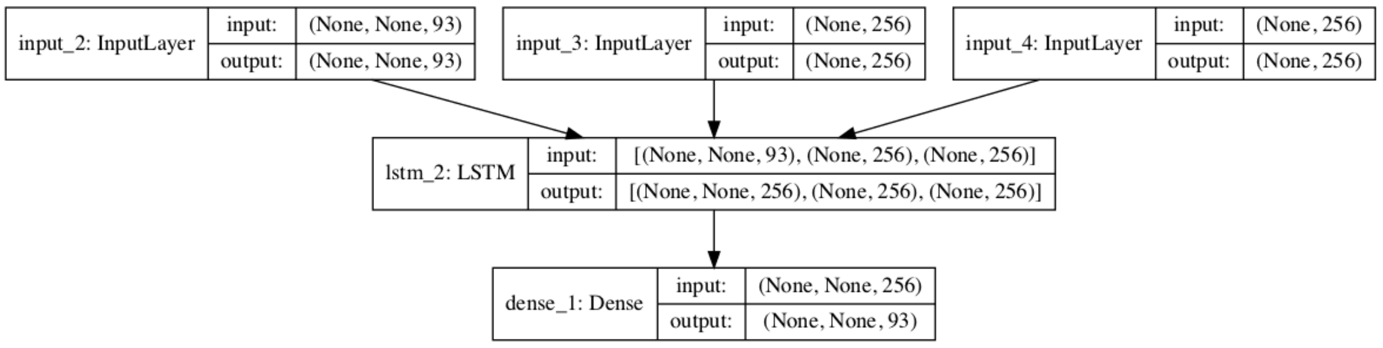
Plots of all three models are then created.



Graph of Encoder Model For Inference

The plot of the encoder is straightforward.

The decoder shows the three inputs required to decode a single character in the translated sequence, the encoded translation output so far, and the hidden and cell states provided first from the encoder and then from the output of the decoder as the model is called recursively for a given translation.


input_2: InputLayer | input: (None, None, 93) | output: (None, None, 93)
input_3: InputLayer | input: (None, 256) | output: (None, 256)
input_4: InputLayer | input: (None, 256) | output: (None, 256)
lstm_2: LSTM | input: [(None, None, 93), (None, 256), (None, 256)] | output: [(None, None, 256), (None, 256), (None, 256)]
dense_1: Dense | input: (None, None, 256) | output: (None, None, 93)

Graph of Decoder Model For Inference

## Further Reading

This section provides more resources on the topic if you are looking to go deeper.

- Francois Chollet on Twitter
- A ten-minute introduction to sequence-to-sequence learning in Keras
- Keras seq2seq Code Example (lstm_seq2seq)
- Keras Functional API
- LSTM API in Keras
- Long Short-Term Memory, 1997.
- Understanding LSTM Networks, 2015.
- Sequence to Sequence Learning with Neural Networks, 2014.
- Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

### Update

For an example of how to use this model on a standalone problem, see this post:

- How to Develop an Encoder-Decoder Model for Sequence-to-Sequence Prediction in Keras

## Summary

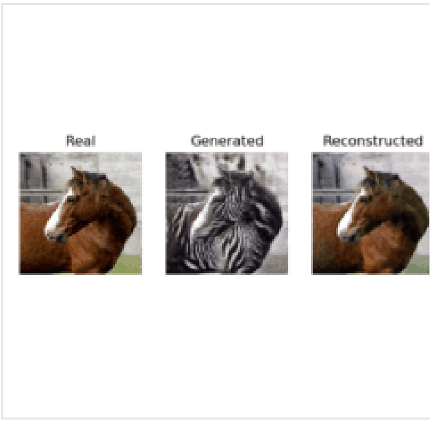
In this post, you discovered how to define an encoder-decoder sequence-to-sequence prediction model for machine translation, as described by the author of the Keras deep learning library.

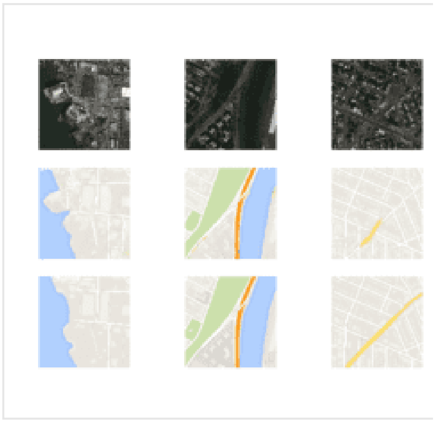
- How to correctly define an encoder-decoder LSTM for training a neural machine translation model.
- How to correctly define an inference model for using a trained encoder-decoder model to translate new sequences.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

Share Tweet Share

## More On This Topic





How to Configure an Encoder-Decoder Model for Neural…

How to Develop a Character-Based Neural Language…

How to Develop a Pix2Pix GAN for Image-to-Image Translation

**About Jason Brownlee**

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.
View all posts by Jason Brownlee →

‹ How to Index, Slice and Reshape NumPy Arrays for Machine Learning

How to Use the Keras Functional API for Deep Learning ›

## 234 Responses to *How to Develop a Seq2Seq Model for Neural Machine Translation in Keras*

**Tom** October 27, 2017 at 11:51 am # REPLY

**nandini** October 31, 2017 at 10:13 pm #

REPLY

i have referenced lstm_seq2seq.py code for requirement,i had executed correctly i got the correct results.
my requirement is that i have to inputs from user and it has to encode to state vectors ,give to decoder ,generate results for given inputs. I have written a logic for it.but i am not able to generate the correct results.

```
userInput="
count=0
while(userInput!='quit'):
userInput = input('enter the english sentences or want to stop (enter quote with quotes)');
userInput=str(userInput)
f = open('testone.txt', 'a' )
if userInput=='quit':
f.close()
else:
f.write( userInput + '\n' )
count=count+1
print("count",count)

# taking inputs from user

#saving in testone.txt file

test_path='testone.txt'
test_texts = []
#target_texts = []
test_characters = set()
#target_characters = set()
lines = open(test_path).read().split('\n')
for line in lines[: min(30, len(lines) – 1)]:
test_text = line

test_texts.append(test_text)
#target_texts.append(target_text)
for char in test_text:
if char not in test_characters:
test_characters.add(char)

test_characters = sorted(list(test_characters))
num_testencoder_tokens = len(test_characters)
max_testencoder_seq_length = max([len(txt) for txt in test_texts])

print('Number of samples:', len(test_texts))
#print('Number of unique input tokens:', num_encoder_tokens)
print("max test encoder seq length",max_testencoder_seq_length)
print("num_testencoder_tokens",num_testencoder_tokens)
test_token_index = dict(
(len(test_texts), max_testencoder_seq_length,num_testencoder_tokens),
dtype='float32')
print("encoder_test_data",encoder_test_data.shape)

for i,test_text in enumerate(test_texts):
for t, char in enumerate(test_text):
encoder_test_data[i, t,test_token_index[char]] = 1.
print("encoder_test_data",encoder_test_data.shape)

encoder_test_inputs = Input(shape=(None,num_testencoder_tokens))
print("encoder_inputs",encoder_test_inputs.shape)

encoder_test_inputs = Input(shape=(None, num_testencoder_tokens))
print("——————-",encoder_test_inputs.shape)

encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_test_inputs)
print("encoder_outputs",encoder_outputs)

# We discard encoder_outputs and only keep the states.
encoder_test_states = [state_h, state_c]

print("encoder_test_states",encoder_test_states)
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs,
initial_state=encoder_test_states)
print('encoder_test_inputs.shape',encoder_test_inputs.shape)
print('encoder_test_states',encoder_test_states)

encoder_test_model = Model(encoder_test_inputs, encoder_test_states)

print(encoder_test_model.summary)

decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
decoder_inputs, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)

decoder_model = Model(
[decoder_inputs] + decoder_states_inputs,
[decoder_outputs] + decoder_states)

# Reverse-lookup token index to decode sequences back to
# something readable.
reverse_input_char_index = dict(
(i, char) for char, i in test_token_index.items())
reverse_testtarget_char_index=dict(
(i, char) for char, i in target_token_index.items())
```

```
# Generate empty target sequence of length 1.
target_seq = np.zeros((1, 1, num_decoder_tokens))
# print("target_seq",target_seq)
# Populate the first character of target sequence with the start character.
target_seq[0, 0, target_token_index['\t']] = 1.

# Sampling loop for a batch of sequences
# (to simplify, here we assume a batch of size 1).
stop_condition = False
decoded_sentence = ”
while not stop_condition:
output_tokens, h, c = decoder_model.predict(
[target_seq] + states_value)
#print("output_tokens",output_tokens)
# Sample a token
sampled_token_index = np.argmax(output_tokens[0, -1, :])
print('sampled_token_index', sampled_token_index)
#sampled_char = reverse_target_char_index[sampled_token_index]
sampled_char = reverse_testtarget_char_index[sampled_token_index]
print("sampled_char",sampled_char)
decoded_sentence += sampled_char

# Exit condition: either hit max length
# or find stop character.
if (sampled_char == '\n' or
len(decoded_sentence) > max_decoder_seq_length):
stop_condition = True

# Update the target sequence (of length 1).
target_seq = np.zeros((1, 1, num_decoder_tokens))
#print("target_seq legnth",len(target_seq))
target_seq[0, 0, sampled_token_index] = 1.

# Update states
states_value = [h, c]

return decoded_sentence

for seq_index in range(count):
#print(encoder_test_data[0:1])
input_seq = encoder_test_data[seq_index: seq_index + 1]
print("input_seq",input_seq.shape)
decoded_sentence = decode_sequence(input_seq)
print('-')
print('Input sentence:', test_texts[seq_index])
print('Decoded sentence:', decoded_sentence)
```

Above is my code ,please suggest where i am going wrong.

What is the problem exactly?

**nandini** November 1, 2017 at 3:44 pm # REPLY

Actually in original code ,they are testing on already trained code,in my code what i am doing is ,i am giving inputs from console and stored in a file,this inputs i am giving to encoder_model, for console inputs i am not getting proper outputs like we got i n trained data.

Suppose i have given same words from console to encoder model also i am getting wrong results.

Please tell me where i am going wrong.

**Jason Brownlee** November 1, 2017 at 4:15 pm # REPLY

Perhaps you can debug your example with static data?

**nandini** November 1, 2017 at 5:16 pm # REPLY

for static data it is working, but when i am taking inputs from users i am not getting proper results.

**Jason Brownlee** November 2, 2017 at 5:07 am # REPLY

It might be related to how you are reading the input and preparing it for the model.

**Ambika** November 2, 2017 at 6:31 pm # REPLY

How to do predictions for sequence to sequence model using keras,why we are not using train model directly,why we are creating inference model in this scenario ,Please can you explain.

i would like to know,how to predict the model outputs for model unknown inputs in sequence to sequence model using keras.

**Jason Brownlee** November 3, 2017 at 5:15 am # REPLY

This is a sophisticated seq2seq model. For a much simpler architecture see here:
https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

I have unknown inputs in eng, i want to translate these inputs to french.So i converted these textst inputs into vectors,like in seq2sq model they have encoded as constant length vector.
I have given these inputs to inference encoder_model,along with these inputs states(hidden,cell states) like encoder_model = Model(encoder_test_inputs, encoder_test_states)
from these i am extracting state value from encoder_model(),this state value i am passing to decoder model,like that decoder_model.predict(
[target_seq] + states_value),but i am getting scarmbled outputs if i will give unknown inputs to encoder_model.

is there any difference procedure available to predict target sequnce for given unknwon inputs sequnce,Please suggest me for this problem.
I am able to decode the already trained data,if is not trained data i am not able to decode.

**Jason Brownlee** November 3, 2017 at 5:17 am # REPLY

Once the model is trained, you must encode new data using the same procedure as you used for the training data then call model.predict()

**nandu** November 3, 2017 at 4:37 pm # REPLY

in that they are not doing model.predict,they are doing encoder_model.predict and decoder_model.predict() seperately.

**Jason Brownlee** November 4, 2017 at 5:26 am # REPLY

Correct.

**nandini** November 6, 2017 at 5:12 pm # REPLY

How to predict unknown target sequence for unknown inputs using keras? for unknown inputs again i need to create input layer and encoder states and encoder model or else or i need to use same encoder_model its taking parameters as encoder_inputs and encoders_states.

**Jason Brownlee** November 7, 2017 at 9:47 am # REPLY

The inference models could be saved, loaded and used to make predictions as in the above examples.

Please can you explain how this inference decoder_model will work in this examples,i am ok with training of the encoder and decoder model ,inference encoder_model,I didn't understnd the inference decoder_model in this example.

# define decoder inference model
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
in training model we are passing decoder_inputs and intial state as encoder states,but in inference something it is differnent things we are passing ,why what is the reason.

decoder_outputs, state_h, state_c = decoder_lstm(decoder_inputs, initial_state=decoder_states_inputs)

decoder_states = [state_h, state_c]

decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model([decoder_inputs] + decoder_states_inputs, [decoder_outputs]

**Jason Brownlee** November 15, 2017 at 9:52 am # REPLY

The key difference is that we recursively pass in the last state as input, starting with the encoder state for the first pass.

Does that help?

**Jonas** November 15, 2017 at 6:52 am # REPLY

Could you please elaborate more on why TimeDistributed is not need here? The Dense layer in Decoder must output one character for each character in the input sequence, so how is it different here from the case where TimeDistributed must be aplied?

**Jason Brownlee** November 15, 2017 at 9:57 am # REPLY

Great question. Dense can now support time steps!

You can add a TimeDistribted wrapper and it will have the same effect.

I know… Keras is getting a little confusing.

**nandini** November 15, 2017 at 6:27 pm # REPLY

HI Jason,

this requirement is quietly different from language translation tool.

Please do any suggestions for it,how to proceed further.

**Jason Brownlee** November 16, 2017 at 10:27 am # REPLY

See this post for almost exactly this problem:
https://machinelearningmastery.com/learn-add-numbers-seq2seq-recurrent-neural-networks/

**nandini** November 15, 2017 at 9:14 pm # REPLY

while predicting target character .

output_tokens, h, c = decoder_test_model.predict(
[target_seq] + states_value)

# Sample a token
sampled_token_index = np.argmax(output_tokens[0, -1, :])
they are using np.argmax(output_tokesn) ,Could you explain how does it works inorder to predict the target character.

**Jason Brownlee** November 16, 2017 at 10:28 am # REPLY

What is the problem exactly?

argmax? See here:
https://en.wikipedia.org/wiki/Arg_max

**Krtk** November 21, 2017 at 1:38 pm # REPLY

Hi Jason, thanks for making a detailed blog.

Did you try saving and restoring the model for inference later?
I'm following the lstm_seq2seq example where the model.save stored the HDF5 file but when I try restoring the model just for inference, the output is all garbage despite the model providing good test responses when train is followed by inference.

**Jason Brownlee** November 22, 2017 at 10:49 am # REPLY

I have not tried that, sorry.

Hi Jason,

Can we identify the grammer using keras,like what is noun and pronoun ,

Please suggest any procedure is there to identify thr grammer using keras.

**Jason Brownlee** November 23, 2017 at 10:28 am # REPLY

Sure. It could be framed as word classification.

You must prepare a dataset of examples then fit your model.

**ambika** November 22, 2017 at 9:24 pm # REPLY

Hi jason,

do you have any example of word level encoding for language translation ,above examples is for character level encoding rigjt.
if you have example please share the link.

**Jason Brownlee** November 23, 2017 at 10:34 am # REPLY

Yes, I have an example in my book:
https://machinelearningmastery.com/deep-learning-for-nlp/

**nandu** November 24, 2017 at 12:02 am # REPLY

Here provided example ,they are doing character level encoding for encoder and decoder,Same model i would like to work on word level encoding for encoder and decoder?
is word level encoding will better than character level encoding?

in order to word level encoding ,which method i need to follow,please suggest me before going to start.

**Jason Brownlee** November 24, 2017 at 9:46 am # REPLY

It depends on the problem whether word or char level will be better. Char may be more flexible but be slower to train, Word may require larger vocab/memory but train sooner.

Sorry, I'm not sure I follow, can you please restate the question?

**Baptiste Amato** November 26, 2017 at 2:29 am # REPLY

Well explained article! I wonder if these sequence-to-sequence can be applied to images, for example to handle different size images, and apply encoder-decoder for segmentation?

**Jason Brownlee** November 26, 2017 at 7:33 am # REPLY

Perhaps, I'm not sure I follow sorry. Do you have an example?

**Baptiste Amato** November 26, 2017 at 8:59 am # REPLY

Let's say I want to develop a Neural Network that returns, given an image, a sort of contour map (like the cars in blue, the people in green), but my data set has various images from different size, making them impossible to stack in an array and put them directly in a CNN. Would it be possible to apply the sequence-to-sequence idea to these images, as we want an image as result and we don't have a uniform size for the input data?

**Jason Brownlee** November 27, 2017 at 5:42 am # REPLY

Interesting challenge.

I don't think seq2seq is the right framing, but I could be wrong.

There are many ways to frame this type of problem and I'd encourage you to explore a few. Perhaps a network that outputs a red and a green image that you combine downstream, or perhaps the network outputs one image with all pixels colored.

**Baptiste Amato** November 27, 2017 at 10:10 am #

"a network that outputs a red and a green image that you combine downstream": it's pretty simple but I did not think about it… Thank you for your comment 🙂

**Jason Brownlee** November 28, 2017 at 8:36 am #

**ambika** November 28, 2017 at 5:55 pm # REPLY

can we store model states and use it further for testing,
Please suggest me the way to store the states in file ,use it further.

**Jason Brownlee** November 29, 2017 at 8:20 am # REPLY

You can, e.g. pickle the state LSTM variable.

Why do you want to save the state?

**ambika** November 29, 2017 at 5:19 pm # REPLY

I am writting the training model and testing model seperately thats why i want to feed encoder_inputs and encoder states to inference model,thats why i would like to store the states.

**Jason Brownlee** November 30, 2017 at 8:06 am # REPLY

I'd recommend using one model to keep things simple.

**ambika** November 29, 2017 at 5:28 pm # REPLY

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model.save("seq2seq.h5")
here i am loading the model from h5 file
model=model.load(seq2seq.h5)
after that i have stored encoder_inputs,encoders_states of trained model using pickling .
next i am trying to create the inference model using encoder_inputs and encoder_states
but i am not able to create the inference model correctly, i am getting the issue like graph disconnected with input layer 1 .
encoder_model = Model(encoder_inputs, encoder_states

Please suggest why i am getting like this issue.

**Etienne** December 16, 2017 at 1:22 am # REPLY

Hey , I had the same issue ( I have implemented a seq 2 seq model with R but I had the

```
get_encoder_model <- function(model) {
  encoder_inputs = get_output_at(get_layer(model,'encoder_inputs'),0)
  encoder_lstm = get_output_at(get_layer(model,'encoder_lstm'),0)
  state_h <- encoder_lstm[[2]]
  state_c <- encoder_lstm[[3]]
  encoder_states = list(state_h,state_c)
  encoder_model = keras_model(encoder_inputs, encoder_states)
  return(encoder_model)
}

get_decoder_model <- function(model) {
  latent_dim = get_output_shape_at(get_layer(model,'decoder_lstm'),0)[[1]][[3]]
  decoder_inputs = get_output_at(get_layer(model,'decoder_inputs'),0)
  decoder_state_input_h = layer_input(c(latent_dim))
  decoder_state_input_c = layer_input(c(latent_dim))
  decoder_states_inputs = list(decoder_state_input_h, decoder_state_input_c)
  decoder = layer_lstm(units = latent_dim,return_sequences = T,return_state=T,name
  decoder_inf_lstm = decoder(get_output_at(get_layer(model,'decoder_inputs'),0), ir

  set_weights(decoder,keras::get_weights(get_layer(model,'decoder_lstm')))
  decoder_outputs <- decoder_inf_lstm[[1]]
  state_h <- decoder_inf_lstm[[2]]
  state_c <- decoder_inf_lstm[[3]]
  decoder_states = list(state_h,state_c)
  decoder_dense = layer_dense(units = n_features,activation='linear',name = 'decode
  decoder_outputs <- decoder_dense(decoder_outputs)
  set_weights(decoder_dense,keras::get_weights(get_layer(model,'decoder_dense')))
  decoder_model = keras_model(inputs = c(decoder_inputs,decoder_states_inputs), out
  return(decoder_model)
}
```

It is R code but you can adapt it to python. The trick was mainly to create new layers and set weight with those trained by your model.

**Jason Brownlee** December 16, 2017 at 5:34 am # REPLY

Thanks for sharing, I added some formatting.

**ambika** December 19, 2017 at 6:27 pm # REPLY

thanks for your suggestion 🙂

**Benoit** December 27, 2017 at 6:59 pm # REPLY

This should do the trick for Python:

```
model = load_model('s2s.h5')

encoder = model.layers[2]
encoder_inputs = model.layers[0].input
encoder_outputs, state_h, state_c = model.layers[2].output
encoder_states = [state_h, state_c]
```

```
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs] + decoder_states)
```

**Jason Brownlee** December 28, 2017 at 5:22 am # REPLY

Thanks for sharing!

**Reihana** July 29, 2018 at 1:22 pm # REPLY

Is ‘s2s.h5’ model they only thing you saved from your encode-decoder for lated prediction?
I mean in the main code, could we eliminate building the encoder-interface and decoder-interface and just saving the model. Later for prediction, we retrieve and build the interface based on only ‘s2s.h5’?

If I do so I get this error:
TypeError: Tensor objects are not iterable when eager execution is not enabled. To iterate over this tensor use tf.map_fn.

My main confusion is whether we have to build the encoder/decoder_interface at the time of building the model (training) or not?

Thanks in advance!

**yating** August 5, 2020 at 8:59 pm # REPLY

when i buiding the encoder-decoder model ,i get this erro:
ValueError: Graph disconnected: cannot obtain value for tensor Tensor(“input_2:0”, shape=(None, None, 150, 150, 1), dtype=float32) at layer “input_2”. The following previous layers were accessed without issue: [‘input_1’]
do you konw why?

**Jason Brownlee** August 6, 2020 at 6:12 am # REPLY

Sorry to hear that, this will help:
https://machinelearningmastery.com/faq/single-faq/why-does-the-code-in-the-tutorial-not

**shago** January 28, 2021 at 10:01 am # REPLY

Hi Yating , did you find the problem? The same thing happens to me, I run tensorflow 2.3. I guess it is something related to the version of tensorflow. Something strange happens, when I run the code in a notebook cell, first the error appears, and if I run the cell again the error disappears, but the code does not run later when I do the predict.

**Kevin** December 15, 2017 at 8:30 pm # REPLY

Hi Jason,

Thank you for this great article! I created a Tensorflow implementation. This gives another perspective on the implementation side of the seq2seq model. The blog post about the Tensorflow implementation is found on Data Blogger: https://www.data-blogger.com/2017/12/14/create-a-character-based-seq2seq-using-python-and-tensorflow/.

**Jason Brownlee** December 16, 2017 at 5:24 am # REPLY

Thanks for sharing Kevin.

**ambika** December 19, 2017 at 6:46 pm # REPLY

Can we add more lstm layres in encoder and decoder inorder to predict the results correctly, is it a good approach to add more layers in encoder and decoder using keras.

**Jason Brownlee** December 20, 2017 at 5:40 am # REPLY

It can help. Try it and see.

**ambika** December 19, 2017 at 6:50 pm # REPLY

encoder and decoder model is good approach to implement a model ,for algorithm as input to model,output is source code generation for specified language .

i am trying to this problem statement using encoder and decoder model using keras,is this good idea to implement,or is there any apporach is there for this problem statement.

Please do suggest for it.

I think it is a good place to start.

**Etienne** December 20, 2017 at 9:06 pm #

REPLY

Hey, First thank you for your article. I've managed to set up a model with only 1 layer for the enoder and 1 layer for the decoder ( both LSTM) . I would like to train my model with more than 1 LSTM layer.

I have added 1 more LSTM to my decoder and set states for both of them with the encoder states. However when I want to predict a new sequence the ouput is very bad. Have you tried to add more than 1 LSTM with keras ?

**nandini** December 20, 2017 at 9:24 pm #

REPLY

why are you not adding more lstm layers to encoder,can i know reason.
Actually i am also trying for the issue,let me why are you not adding more lstm layers to encoder rather than decoder?

**Etienne** December 21, 2017 at 1:13 am #

REPLY

When I try to create sophisticated model I start with basic model and then i try to improve it.
Sure you can add 8 layers for the encoder and 8 for the decoder. But I prefer build it step by ste.
I w'll try to add layers for the encoder for sure but i don't think that it will solve my problem… The training part is Ok with 2 lstm for the decoder but thr problem comes when i try to predict something with my inference model

**Nandini** December 21, 2017 at 3:08 pm #

REPLY

how can share code of 2 lstm layers ,how you have implemented,same i have tried with 3 lstm layers encoder, and 3 lstm layers with decoder.but while creating inference decoder model , i am getting shaping issue.

**Nandini** December 21, 2017 at 5:03 pm #

how can share code of 2 lstm layers decoder,how you have implemented,same i have tried with 3 lstm layers encoder, and 3 lstm layers with decoder.but while creating inference decoder model , i am getting shaping issue

**Jason Brownlee** December 21, 2017 at 5:25 am # REPLY

I use a single layer for the encoder and decoder to keep things simple in the example.

**Jason Brownlee** December 21, 2017 at 5:24 am # REPLY

You will need to tune the model, perhaps longer training, a new batch size and other config changes.

**Greg** April 14, 2018 at 6:44 am # REPLY

How can I add more LSTM layers to the encoder and decoder? I'm having some trouble with the syntax.

**Jason Brownlee** April 14, 2018 at 6:50 am # REPLY

See this post for a tutorial on how:
https://machinelearningmastery.com/stacked-long-short-term-memory-networks/

**nandin** January 3, 2018 at 4:43 pm # REPLY

given test case i need to generate a script file for it,i am able to generate results some how ok. but i would to recognize in my test case what are the variables they has to be same in script file also,Please give any suggestion for it.

how to recognize variable as per the input.
ex: Define a integer variable a : int a

Define a integer variable add : int add.

how model can recognize a and add variable in test case.
please do any suggestions for it.

**Jason Brownlee** January 4, 2018 at 8:05 am # REPLY

Sorry, I'm not sure I follow.

If you are having trouble defining your machine learning problem, maybe this post will help:

**Nandini** January 9, 2018 at 11:04 pm # REPLY

is it good approach to increase accuracy of encoder and decoder model to adding more lstm layers ?

**Jason Brownlee** January 10, 2018 at 5:26 am # REPLY

It can be, depends on the dataset.

**Hardik Khandelwal** January 12, 2018 at 8:38 pm # REPLY

Thanks for the post. It's really helpful.
Why are we using only 10,000 samples out of 150,000. Is it because of huge memory requirements or is there some other reason.

**Jason Brownlee** January 13, 2018 at 5:32 am # REPLY

Yes, because of the large space and time complexity.

**Soumil Mandal** January 31, 2018 at 5:11 am # REPLY

Thanks for the tutorial, it was quite helpful. In https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html he does mention in FAQ sec how to change the code to convert it into a GRU seq2seq model, but he does not mention how to change the inference model accordingly, any help would be really appreciated, I'm stuck here for quite some time. Thanks again though.

**Ashima** February 20, 2019 at 5:29 am # REPLY

Hi Soumil,

By any chance did you sort this out???

**huaiyanggongzi** February 1, 2018 at 4:55 am # REPLY

**Jason Brownlee** February 1, 2018 at 7:27 am # REPLY

Sure.

**Ajay Prasadh Viswanathan** February 6, 2018 at 9:20 am # REPLY

Hey Jason, Thanks for an awesome article.
I was wondering why you were padding the input with a max number of characters as 16. I could not see where this particular number was explicitly referenced in the model again. So I suppose I could feed non padded input to this model and the sequence to sequence model would still work ?

Also if we are going to pad input and output to a finite length. Could have not used just a simple sequence classification architecture something like

```
inputs = Input(shape = (in_max_characters, in_char_vocab_size))
```

h1 = LSTM(128)(inputs)

```
outputs = TimeDistributed(Dense(output_char_vocab_size), input_shape =
(out_max_characters, 128))
```

My question is regarding the conceptual and practical necessity of padding in context of sequence to sequence models.

**Ajay Prasadh Viswanathan** February 6, 2018 at 9:24 am # REPLY

I forgot to add the repeat vector in my code earlier. Here is an updated version.

```
inputs = Input(shape = (in_max_characters, in_char_vocab_size))
h1 = LSTM(128)(inputs)
h2 = RepeatVector(out_max_characters)(h1)
outputs = TimeDistributed(Dense(output_char_vocab_size), input_shape =
(out_max_characters, 128))(h2)
```

**Jason Brownlee** February 6, 2018 at 9:28 am # REPLY

For this model, I believe padding is not required as it processes one time step at a time.

**Daniel** February 23, 2018 at 3:50 am # REPLY

In the case of sequence2sequence word based (as explained in the: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html) they introduced an embedding layer. With the embedding layer, how do I set the model to make it able to understand that a single word is a single time step? Do i have to set the embedding's input_dim parameter to 1 that give an output of (?,1,vocab_size) ? Or can I set the embedding's input_dim parameter to maxlen of the corpus sentences that give an output of (?,maxlen,vocab_size) ?

Thank you for your reply

**Jason Brownlee** February 23, 2018 at 12:02 pm # REPLY

With the embedding, your input will be a sequence of integers. The embedding will map integers to the high-dimensional vector.

Therefore, this post will help you with reshaping your sequences of integer inputs:
https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/

Does that help?

**Daniel** February 25, 2018 at 6:07 am # REPLY

Yes, thank you.
But if i have the following code:

i1=Input(shape=(2,))
emb=Embedding(input_dim=1, output_dim= 10 , input_length=1(i1) # tensor (num_sample, 2, 10)
lst=LSTM (16)(emb) # <- is it necessary declare input_shape?

In this case is it necessary declare input_shape of LSTM? emb is a tensor 3D with number of example, time step and dimension of output.

Thank you so much for your time.

**Jason Brownlee** February 25, 2018 at 7:45 am # REPLY

No, the embedding will provide 3d input to the LSTM layer.

**Saket Karve** February 24, 2018 at 7:02 pm # REPLY

I am using a sequence to sequence model to predict keyphrases from a given text article. I am using the exact same model as mentioned in this blog for this purpose. However, I am confused whether predict the output. However, after saving the trained model, how do we make sure the trained weights are used in the encoder_model and the decoder_model while inference?

Actually, I am getting the same output irrespective of the input.

**Jason Brownlee** February 25, 2018 at 7:42 am # REPLY

Perhaps there is something going on with the saved model.

Perhaps try to get it working in memory with a small example, then try saving/loading and reproducing the result to ensure there are no faults.

**Shannon** March 11, 2018 at 4:39 pm # REPLY

Great tutorial. I want to create a sequence to sequence model for images.

Here is an example of my training set :

x1 x2 x3 y1 y2 y3, where x1 x2 and x3 are the input sequence of images and y1, y2, and y3 are the following sequence of images which I want to forecast.

My question is how do I represent x1 x2 and x3 as input to a neural network or to say Encoder-Decoder Sequence-to-Sequence Model?

**Jason Brownlee** March 12, 2018 at 6:26 am # REPLY

That sounds like a challenging problem. Perhaps you can find examples of existing models that output an image that you can use for the output part of the model. The input could be a modified VGG or similar network.

**Zoe** March 24, 2018 at 6:13 am # REPLY

Thank you very much for sharing this great article, Jason.
I don't understand the inference model. why do we need the inference model after we defined our encoder and decoder. Don't we just need to define the inputs/outputs and structure of the encoder/decoder, train this model, and then use it to predict? Very confused now. Thank you for your help in advance.

**Jason Brownlee** March 24, 2018 at 6:33 am # REPLY

I'd recommend writing wrapper functions to train/save/load/predict with models in this form to hide the complexity.

**Parul** April 5, 2018 at 8:43 am # REPLY

How do I fit the model? In your vanilla lstm you had used

model.fit(trainX, trainY, epochs=30, batch_size=64, validation_data=(testX, testY), callbacks=[checkpoint], verbose=2)

What would be the equivalent for that in this model?

**Jason Brownlee** April 5, 2018 at 3:07 pm # REPLY

It is more complex, here's an example:
https://machinelearningmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/

**George M.** April 10, 2018 at 1:32 am # REPLY

Why is the decoder output ahead by one timestep?

**Jason Brownlee** April 10, 2018 at 6:22 am # REPLY

What do you mean exactly George?

**George M.** April 10, 2018 at 8:33 am # REPLY

Dr. Brownlee,

First of all, very great tutorial! I have all you articles bookmarked.

I went through the https://github.com/keras-team/keras/blob/master/examples/lstm_seq2seq.py which has the same code you have here. At line 117 where they create the target values for the output of decoder it says it's ahead of the decoder input by 1 timestep.
Why do you think that is?

Sorry, I wasn't specific enough before.

I don't know, sorry George.

**Vivek** January 19, 2019 at 1:54 am # REPLY

That's basically coz we wanna compare the decoder input data to the decoder target data. So the decoder target data has to be one time step ahead of the decoder input data

**midabomb** April 11, 2018 at 2:00 pm # REPLY

Hello Jason,

In inference mode, we define an encoder_model:

# Define sampling models
encoder_model = Model(encoder_inputs, encoder_states)

Then we use this model here:
states_value = encoder_model.predict(input_seq)

I am used to defining model, then compile and fit, before using it to predict. Here, we just define the model then predict.

Obviously, I am missing something. Could you elaborate a bit on what is going on here behind the scene?
Thanks,
-MDB

**Jason Brownlee** April 11, 2018 at 4:27 pm # REPLY

I believe compilation is only required for fitting the Keras model.

**midabomb** April 12, 2018 at 1:05 am # REPLY

Okay Thanks Jason.

In my example, there is no fitting. Where the weights for the predict is coming from Jason?
-MDB

**Jason Brownlee** April 12, 2018 at 8:47 am # REPLY

**Sukruthi** April 12, 2018 at 5:42 pm # REPLY

HI Jason. How can i save the model then use it another time from the model saved

**changito** April 14, 2018 at 7:06 am # REPLY

Here, the difference between a good post(Github) and an excellent post(this). Thank you amigo!

**Jason Brownlee** April 15, 2018 at 6:14 am # REPLY

Thanks!

**Mayur Jain** April 24, 2018 at 11:32 pm # REPLY

Thank you for the wonderful post.
I have this question that the, LSTM model used for Machine Translation, can be applied for dialogue generation ?

Please let me know if it can be done. If yes, what are major tweaks that needs to be done.

**Jason Brownlee** April 25, 2018 at 6:31 am # REPLY

LSTMs can be used as generative models for language, see here for examples:
https://machinelearningmastery.com/?s=language+model&post_type=post&submit=Search

**bratt** April 24, 2018 at 11:49 pm # REPLY

Hi Jason, your blogs are really great and taught me much, thank you for your opensource work!

I made this code for a prediction, regression task but I want to understand the math behind it as well. Why I go first with return sequence false in this case and what are my activation functions ? they are not named and where do I know which is the default one ? Thank you in advance.

inputs = Input(shape=(timesteps, input_dim))
encoded = LSTM(n_dimensions, return_sequences=False, name="encoder")(inputs)
decoded = RepeatVector(timesteps)(encoded)
decoded = LSTM(input_dim, return_sequences=True, name='decoder')(decoded)

**Jason Brownlee** April 25, 2018 at 6:34 am # REPLY

LSTMs and the encoder-decoder are not suited for regression, unless you have a sequence of inputs and outputs.

You can learn more here:
https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

**bratt** April 25, 2018 at 5:57 pm # REPLY

My inputs are sequences, 100 measurements with each containing 10.000 data points. The point that I do not understand is, which I type of activation function is build in in the model, also in your other link it is not mentioned what the function is ?

**Jason Brownlee** April 26, 2018 at 6:25 am # REPLY

Perhaps use the defaults first in order to evaluate model skill?

**bratt** April 27, 2018 at 1:51 am #

Hi Jason, sorry for the missunderstanding, but you brought it on point. What are the default activation functions of the encoding decoding layers ?

**Jason Brownlee** April 27, 2018 at 6:06 am #

You can see here:
https://keras.io/layers/recurrent/#lstm

**wende** May 11, 2018 at 5:04 pm # REPLY

Hi Jason, how can i adopt it for bidirectional architecture? I keep getting an error on the decoder part
'valueError: Dimensions must be equal, but are 256 and 512 for 'lstm_2_1/MatMul_4' (op: 'MatMul') with input shapes: [?,256], [512,512].

'

Here is an example of creating bidirectional LSTMs:
https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/

**wende** May 12, 2018 at 7:14 pm # REPLY

Thank you, Jason.

**Jason Brownlee** May 13, 2018 at 6:35 am # REPLY

No probs.

**zishan** May 20, 2018 at 10:57 pm # REPLY

Hi Jason, when I try to use this model I got an error with expecting 2 input arrays. I know it has something to do with this line model = Model([encoder_inputs, decoder_inputs], decoder_outputs) and decoder part. I have a target array which my model should encode and decode in the end. but how do I define the decoder_inputs in model.fit(train, train….) ?

**Jason Brownlee** May 21, 2018 at 6:31 am # REPLY

Are you able to confirm that you copied all of the code exactly?

**paul** June 1, 2018 at 9:01 am # REPLY

Hi Jason, i am really new to python and machine learning and have not an IT background, so I might ask not clear enough and would provide details later as necessary. I know that you commented that it is not good for time series but I still want to see my results. How do I do use this encoder decoder model with a sliding window approach ? Could you give me like the steps or blocks how to proceed with that ? Thank you in advance.

**Jason Brownlee** June 1, 2018 at 2:45 pm # REPLY

My best advice is here:
https://machinelearningmastery.com/faq/single-faq/how-do-i-use-lstms-for-time-series-forecasting

Does that help?

I am a bit confused since I am new to ML, is there a big difference in the two models you described here and on your post here https://machinelearningmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/

**Jason Brownlee** June 5, 2018 at 6:39 am # REPLY

Yes, the linked approach matches the definition of the method as described in the research papers.

The above approach is a simplification of the approach that is easier to implement and understand and gives similar performance.

**duwa** June 9, 2018 at 12:26 pm # REPLY

Hi sir,
i have followed your tutorial and everything working fine until i train model to the fullest. I have 8gb ram and another 8gb on swap. I can see the ram going to the fullest through hardware monitors. Is there any way to avoid memory issues, or train the model again and again for different dataset ? P.S- I'm newbie to this stuffs 😳

```
===========================
Using TensorFlow backend.
English Vocabulary Size: 8773
English Max Length: 9
German Vocabulary Size: 15723
German Max Length: 17
Traceback (most recent call last):
File "train_model.py", line 85, in
trainY = encode_output(trainY, eng_vocab_size)
File "train_model.py", line 48, in encode_output
y = array(ylist)
MemoryError
```

**Jason Brownlee** June 10, 2018 at 5:58 am # REPLY

Sorry to hear that.

Perhaps try running the example on EC2 with more RAM?
Perhaps try using progressive loading instead of loading all data into memory?

**Ritaprava Dutta** October 20, 2019 at 4:14 pm # REPLY

**Reihana** July 24, 2018 at 8:12 am # REPLY

Hi Json,

How we can see the score of model? I mean the precision, recall, … on train set?

**Jason Brownlee** July 24, 2018 at 2:30 pm # REPLY

Generally, we use measures like BLEU or ROGUE to evaluate the performance of a NMT system.

**tom** July 26, 2018 at 5:12 am # REPLY

Hi Jason,
I read your blogs and its really easy to follow. But one thing confused me I saw somewehre a model for seq2seq with a repeatvector. Could you please tell the difference ? are there any papers describing the model with the inference and the model with the repeatvector? thanks in advance.

**Jason Brownlee** July 26, 2018 at 7:48 am # REPLY

It is a simpler design where the output of each time step is conditional on the context vector only. The internal state of the encoder is not shared with the decoder.

Works just as well if not better and is a lot simpler to implement in Keras.

More here:
https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

**Reihana** July 31, 2018 at 3:51 am # REPLY

Hi Jason,
Thanks a lot for this beautiful tutorial.

I have a very huge problem with the logic behind the interface model.
Interface model is going to predict the label for the data that we don't have label for it. am I correct?

So then, the the decode_sequence(input_seq):
we have this:
# Generate empty target sequence of length 1.
target_seq = np.zeros((1, 1, num_decoder_tokens))
# Populate the first character of target sequence with the start character

How we could do that? I mean why we are setting the input decoder like this when we know that it is going to be used in predicting the output data.

For sure I was expecting that since this is a vector of zero, no matter what are the encoder_states, the model will underfit everything to zero which is doing that for me now!

**Jason Brownlee** July 31, 2018 at 6:12 am # REPLY

Perhaps try this much simpler approach:
https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

**Reihana** July 31, 2018 at 8:25 am # REPLY

Just noticed that the nature of Decoder is designed to decode one token at a time.
Meaning that it is not taking a sequence of tokens and predict a sequence of tokens.
That being said, the decoder input is being used to keep track of each token being discovered at a timestep, and because of that it just have the starting point to predict the next token.

Got it!
Thanks Jason! your blog is my first recommendation to everyone!

**Jason Brownlee** July 31, 2018 at 2:56 pm # REPLY

Correct.

**Monika Jain** August 2, 2018 at 11:01 pm # REPLY

Hi, I need a single encoder and three decoders branching out parallely out of it. Can somebody suggest how to code for that?

**Jason Brownlee** August 3, 2018 at 6:03 am # REPLY

Try a multi-output model:
https://machinelearningmastery.com/keras-functional-api-deep-learning/

**betsi** August 9, 2018 at 12:54 pm # REPLY

**Jason Brownlee** August 9, 2018 at 2:05 pm # REPLY

Perhaps try this simpler tutorial instead:
https://machinelearningmastery.com/develop-neural-machine-translation-system-keras/

**Angel** August 12, 2018 at 6:34 pm # REPLY

Hello Jason 😳
Really thanks a lot for the tutorial it's very helpful

I'm still rather new to deep learning,…I have tried to train the model on the dataset, but I wonder how can I reload my saved model to test it on a new set of data?

**Jason Brownlee** August 13, 2018 at 6:16 am # REPLY

I give examples of saving and loading Keras models here:
https://machinelearningmastery.com/save-load-keras-deep-learning-models/

**Richa Ranjan** August 20, 2018 at 8:45 am # REPLY

One of the best tutorials I've seen for sequence2sequence!

Just one doubt, how are the inference models connected to the actual model that we trained? While making predictions, if I try to load a previously trained model, I get different results than at the time of training. Basically, I'm not clear on how the model, encoder_model and the decoder_model are linked?

This may be a repeated query, but I haven't had any luck with the answers. Any help would be appreciated. Thanks in advance!

**Jason Brownlee** August 20, 2018 at 2:15 pm # REPLY

They are linked by using the same weights, but different interfaces to the data.

**Vishwas** August 28, 2018 at 8:00 pm # REPLY

Hi Jason, Is there Back propogation taking place in Encoder?,If yes,then how loss function is calculated as we dont have any target variables

Yes, we are backpropagating from the output to the decoder to the encoder.

**Vishwas** August 29, 2018 at 3:01 pm # REPLY

Hello Jason,
Thanks for your quick response, But I had one more question.

In the encoder step will be creating a thought vector for the given input sentence(incase of Machine Translation) and giving it to the Decoder. So during backpropogation , In the encoder we are not performing any prediction,So why do we need to update the weights?

*Hope you could give a brief description on this ,As I was not able to find any material which explains Backprop in Seq2Seq model.*

Thanks!

**Jason Brownlee** August 30, 2018 at 6:25 am # REPLY

Think of it as one large model, errors are propagated back as per any other neural net.

**Amel Musić** September 23, 2018 at 5:12 am # REPLY

Hi,
thank you for this great tutorial. Can you please share code for stacked layers for us. From what I can see, lot of us are struggling to stack decoder and encoder.

I've stacked encoder like this:
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder_hidden = LSTM(latent_dim, return_state=True, return_sequences=True)(encoder_inputs)
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_hidden)

And decoder like this:
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.

decoder_hidden = LSTM(latent_dim, return_sequences=True, return_state=True)(decoder_inputs, initial_state=encoder_states)

decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_hidden)

Can you please post how would you stack additional layer during training and inference.
Thnx

**Amila Gunawardana** November 7, 2018 at 1:24 pm # REPLY

I'm working on Seq2Sql (natural language to SQL) project and I want to detect specific words in a question

eg: what is the last name of Rick?

for this example, Rick should be detected as the required word.

SELECT last_name FROM student WHERE first_name='Rick' ; and also the Column name

eg: how old is leo?

for this example,leo should be detected as the required word.

How can I write a machine learning program to detect it in any question? Please help.
I have written from select column detection model and I'm stuck with where clause.

**Jason Brownlee** November 7, 2018 at 2:50 pm # REPLY

You will need thousands or millions of examples of inputs and outputs to learn from.

Perhaps model the input and output at the word level?

**Nandini** December 5, 2018 at 11:06 pm # REPLY

I would like to store and load these encoder and decoder model another machine without training the input samples ,Directly i would like to give input samples for predictions ,it has to predict instantly ,

Kindly suggest on it .

**Jason Brownlee** December 6, 2018 at 5:56 am # REPLY

This post shows you how to save:
https://machinelearningmastery.com/save-load-keras-deep-learning-models/

**Nandini** December 6, 2018 at 6:18 pm # REPLY

**Jason Brownlee** December 7, 2018 at 5:19 am # REPLY

I believe you no longer need to compile models after loading.

**John Alexander** December 9, 2018 at 10:28 am # REPLY

I'm confused about how to test the model. What I've done is doing cross validation, but the model always return meaningless output for the test data. But giving excellent output with training data. I got train acc about 99% and 97% with the validation data. But not with the test data. Any idea? Thank you

**Jason Brownlee** December 10, 2018 at 6:04 am # REPLY

Accuracy is a poor measure for NLP problems, I would encourage you to use BLEU, or similar.

**hossein** December 25, 2018 at 11:30 pm # REPLY

Hello Jason,
We used lstm_seq2seq model without any change in the model. The dataset used is the same as the fra-eng (in http://www.manythings.org/anki/fra-eng.zip), except the output is equal to the input which is eng-eng.
This model has good results in short sentences (length 1 or 2), but in larger-length sentences (larger than 3), the results are not good.

Example of good results:
input: Got it output: Got it
input: Got it output: Got it

Example of bad results:
Inpute : Is anybody here? output: Is anyobe youre?
input: It was fabulous. output: It was afbucky.

**Jason Brownlee** December 26, 2018 at 6:44 am # REPLY

Perhaps confirm your libraries are up to date?
Perhaps try re-fitting the model a few times?

Thanks Jason for answer
Of course.
We try the different models, such as this model with 512 and 1024 neuron in 2 and 4 layers. But there was no significant change in the output !!!

**Jason Brownlee** December 30, 2018 at 5:37 am # REPLY

Perhaps try tuning the parameters of the optimization algorithm, e.g. learning rate, etc.

**hossein** December 30, 2018 at 4:16 pm #

Jason I want to design a model for text representation in a vector. Each text file represent in a vector. I try to impediment seq2seq encoder-decoder LSTM and then get encoder vector for each text. But I failed!
Can you help me?

**Jason Brownlee** December 31, 2018 at 6:05 am #

Sounds like a good approach. What is the problem exactly?

**Vivek** January 19, 2019 at 1:52 am # REPLY

sir my problem is that here we have one-hot encoded each character in the sentences or sequence, can we use word embeddings like word2vec or specially glove coz glove is appropriate to use as we are not doing any sorta semantic similarity. I want to embed my words into glove embeddings and then send them to the encoder. How wil that work if I am not vectorizing that with one-hot instead using word level embedding.

Thanks

**Jason Brownlee** January 19, 2019 at 5:46 am # REPLY

Perhaps you can use the embedding for chars, I don't have an example, sorry.

**Sangeeth** February 9, 2019 at 10:34 am # REPLY

What is the loss function here?. Can I use Loss function same as that of Auto-encoders (reconstruction Loss + Regularization Loss), if I am using RNN encoder decoder for time series analysis?. Thanks,

**Jason Brownlee** February 10, 2019 at 9:39 am # REPLY

No loss function, we are just explaining an architecture.

More on loss functions here:
https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

**Sangeeth** February 10, 2019 at 3:59 pm # REPLY

Thank you.
Is it the loss function which is actually doing the dimension reduction or the number of hidden layers?. I guess we have to use conditional probability to find the next best word using beam search in the decoder?. It would be really great if it would be possible to create a post which explains the complete implementation of RNN encoder decoders from scratch?. Thanks,

**Jason Brownlee** February 11, 2019 at 7:56 am # REPLY

I do have many, perhaps start here:
https://machinelearningmastery.com/start-here/#lstm

**Ashima** July 11, 2019 at 8:22 am # REPLY

Hi Sangeet,

I am also doing something similar for my project.

Were you able to calculate conditional probability at decoder to predict the sequence using this model?

Thanks

**fatma** April 1, 2019 at 8:36 pm # REPLY

I am a PhD student and I am work in a semantic index. In addition to, I need a seq2seq trained model for English to any language to use it in my work. where can I find one?

I don't know sorry, perhaps try a google search?

**JY** April 2, 2019 at 12:02 pm # REPLY

Hello, Dr. Brownlee.
I am a bit confused about the exact workings of sequence to sequence model, particularly as to how the number of decoder RNN cells is determined.
In the case of word-based English-to-French translation for example, does one have to make the number of decoder cells fit to the longest french sentence (e.g. the longest target instance)? So the longest possible sentence generated is the same as the longest target instance in the training data?

**Jason Brownlee** April 2, 2019 at 2:21 pm # REPLY

Good question, I recommend using this simpler approach based on an LSTM autoencoder:
https://machinelearningmastery.com/develop-neural-machine-translation-system-keras/

**Aaron** April 2, 2019 at 11:23 pm # REPLY

Where did you come up with the latent_dim of 256? What is the purpose / importance of that number?

**Jason Brownlee** April 3, 2019 at 6:42 am # REPLY

It is arbitrary. More here:
https://machinelearningmastery.com/faq/single-faq/how-many-layers-and-nodes-do-i-need-in-my-neural-network

**Aaron** April 4, 2019 at 1:06 am # REPLY

Does the output dimensionality have to be the same as the input dimension? For instance, could I have an input dimension of (n_samples, 120 steps per sample, 20 features) with an output of (n_samples, 7 steps per sample).

IE – I'm using 120 steps with 20 features each to predict 7 steps, where I don't need the features

**Jason Brownlee** April 4, 2019 at 7:57 am # REPLY

Input and output lengths can vary with an encoder-decoder model.

**Aaron** April 4, 2019 at 12:40 pm # REPLY

So what would my decoder_input look like? would it be Input(shape=(None,)) or Input(batch_shape=(None, 7)) ?

**Jason Brownlee** April 4, 2019 at 2:14 pm # REPLY

Input shape is unrelated to output shape. It sounds like your input shape would be [n, 120, 20], output shape would be [n, 7]

**Aaron** April 4, 2019 at 11:25 pm #

Right – sorry about the confusion – trying to figure out how the input for my decoder layer will look, tried a couple of things, but got dimensionality errors

**mridul ahmed** April 15, 2019 at 2:59 am # REPLY

AttributeError Traceback (most recent call last)
in
9 latent_dim = 256
10 # Define an input sequence and process it.
—> 11 encoder_inputs = Input(shape=(None, num_encoder_tokens))
12 encoder = LSTM(latent_dim, return_state=True)
13 encoder_outputs, state_h, state_c = encoder(encoder_inputs)

~\Anaconda3\lib\site-packages\keras\engine\input_layer.py in Input(shape, batch_shape, name, dtype, sparse, tensor)
176 name=name, dtype=dtype,
177 sparse=sparse,
–> 178 input_tensor=tensor)
179 # Return tensor including _keras_shape and _keras_history.
180 # Note that in this case train_output and test_output are the same pointer.

~\Anaconda3\lib\site-packages\keras\legacy\interfaces.py in wrapper(*args, **kwargs)
89 warnings.warn('Update your `' + object_name + '` call to the ' +
90 'Keras 2 API: ' + signature, stacklevel=2)
—> 91 return func(*args, **kwargs)
92 wrapper._original_function = func

38 prefix = ‘input’
—> 39 name = prefix + ‘_’ + str(K.get_uid(prefix))
40 super(InputLayer, self).__init__(dtype=dtype, name=name)
41

~\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py in get_uid(prefix)
72 “””
73 global _GRAPH_UID_DICTS
—> 74 graph = tf.get_default_graph()
75 if graph not in _GRAPH_UID_DICTS:
76 _GRAPH_UID_DICTS[graph] = defaultdict(int)

AttributeError: module ‘tensorflow’ has no attribute ‘get_default_graph’

i am facing this kind of error while running the above code

from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
from keras.layers import Dense
from keras.utils.vis_utils import plot_model
# configure
num_encoder_tokens = 71
num_decoder_tokens = 93
latent_dim = 256
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don’t use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation=’softmax’)
decoder_outputs = decoder_dense(decoder_outputs)
# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
# plot the model
plot_model(model, to_file=’model.png’, show_shapes=True)
# define encoder inference model
encoder_model = Model(encoder_inputs, encoder_states)

decoder_outputs, state_h, state_c = decoder_lstm(decoder_inputs, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model([decoder_inputs] + decoder_states_inputs, [decoder_outputs] +
decoder_states)
# summarize model
plot_model(encoder_model, to_file='encoder_model.png', show_shapes=True)
plot_model(decoder_model, to_file='decoder_model.png', show_shapes=True)

**Jason Brownlee** April 15, 2019 at 7:54 am # REPLY

Sorry to hear that, I have not seen this error before.

I have some suggestions here:
https://machinelearningmastery.com/faq/single-faq/can-you-read-review-or-debug-my-code

**Kyle Caron** April 21, 2019 at 12:46 am # REPLY

Hey Jason,

I know BLEU score is a common metric for problems such as this because multiple different translations can be considered correct. Since accuracy is a bad measure, how can you measure loss on the training and validation steps over number of epochs? Would that plot be uninterpretable?

If it is uninterpretable, whats a good approach to test for over/under fitting for this model structure?

Best,
Kyle

**Jason Brownlee** April 21, 2019 at 8:23 am # REPLY

Good question Kyle.

Loss is still a reliable measure of the fit of the model because it is the metric that is being optimized by SGD.

**Neha** May 15, 2019 at 10:44 pm # REPLY

Hi Jason,

Just a small question..

if we have cardinality of 250 for inputs to the encoder decoder model, can we define LSTM latent

**Jason Brownlee** May 16, 2019 at 6:32 am # REPLY

Test different numbers of units and compare performance.

**Neha** May 28, 2019 at 12:11 am # REPLY

Hi Jason,

When tested with LSTM units greater than cardinality of input sequence, the autoencoder performs better. What intuition do we get out of this scenario?

**Jason Brownlee** May 28, 2019 at 8:17 am # REPLY

Overparameterization + regularization is very powerful.

**Neha** May 28, 2019 at 7:44 pm #

Thanks Jason.

**Shrikrishna** June 25, 2019 at 8:01 pm # REPLY

Hi Jason.

it's an awesome artice, thank you for such a great post.

I am working on Abstractive Text Summmarization using GloVe embeddings.
AutoEncoder is used for the purpose of encoding.

After executing the above code how should I get the summarized text.

Top 3 lines for configuration are as below

# configure
num_encoder_tokens = EMBED_SIZE #EMBED_SIZE = 50
num_decoder_tokens = 93
latent_dim = LATENT_SIZE #Y-Dimension of the sentence vector

Code Snippent:-

encoding_0 = encoder_model.predict(next(test_gen)[0])
encoding_1 = encoder_model.predict(next(test_gen)[1])

My simple question is how can we get the summarized text after defining the decoder inference model

ValueError: Error when checking model : the list of Numpy arrays that you are passing to your model is not the size the model expected. Expected to see 3 array(s), but instead got the following list of 1 arrays:
[array([[[ 0. , 0. , 0. , …, 0. ,
0. , 0. ],
[ 0. , 0. , 0. , …, 0. ,
0. , 0. ],

Can you please help me out.

Thanks.

**Jason Brownlee** June 26, 2019 at 6:41 am # REPLY

The error suggests that the shape of the input does not match the expectations of the model.

You can change the input or change the model.

**Shrikrishna** July 3, 2019 at 5:36 pm # REPLY

Thanks Jason.
I got the issue resolved.

**Jason Brownlee** July 4, 2019 at 7:40 am # REPLY

I'm happy to hear that.

**RC** September 6, 2019 at 3:42 pm # REPLY

Hi..Great post and the comments…A Noob question..How do we decide the variable latent_dim ?

**Jason Brownlee** September 7, 2019 at 5:20 am # REPLY

Perhaps trial and error.

**Alejandro Oñate Latorre** October 1, 2019 at 6:07 am # REPLY

If you create the encoder-decoder models. Then you train the model and save your weights:
model.save('s2s.h5')

When we want to load the weights we should:
– Create the models again.
– Load the weights in the model (instead of training it) -> model.load_weights('s2s.h5')
– Make predictions ???

And nothing else is necessary?

Here is my doubt. How do prediction models (encoder_model & decoder_model) have access to training?

Is it necessary to do something so that the prediction models (encoder_model & decoder_model) have their weights trained?

Help, I'm lost in this issue!

**Jason Brownlee** October 1, 2019 at 7:02 am # REPLY

You can save the weights and models together. No need to create the models again.

So, yes, but save one file, load one file, make predictions. An example is here:
https://machinelearningmastery.com/save-load-keras-deep-learning-models/

No access to training is needed, the learning has occurred and is captured in the weights.

**Ritaprava Dutta** October 20, 2019 at 4:10 pm # REPLY

For decoder input why do we consider the French text? Shouldn't it just be the return states from encoder?

**Jason Brownlee** October 21, 2019 at 6:15 am # REPLY

It is useful to know what was generated before.

**Vaibhav Vijay kotwal** October 29, 2019 at 3:43 am # REPLY

This seems to be teacher forcing method? Is there a implementation without teacher forcing method?

No, I believe all of my examples use teacher forcing – because it is so effective.

**jeffchen** October 29, 2019 at 3:39 pm #

REPLY

Hi Jason,

I'm wondering know how can I change lstm code to gru code, I know gru just have 1 states so I change the code below :

“”””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””

```
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = GRU(latent_dim, return_state=True)
encoder_outputs, encoder_states = encoder(encoder_inputs)
decoder_inputs = Input(shape=(None, num_encoder_tokens))
decoder_gru = GRU(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _= decoder_gru(decoder_inputs,
initial_state=encoder_states)
decoder_dense = Dense(num_encoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit([encoder_input_data, decoder_input_data], decoder_target_data,
batch_size=batch_size,
epochs=5,
validation_split=0.2)
model.save('s2s.h5')

encoder_model = Model(encoder_inputs, encoder_states)

decoder_states_inputs = Input(shape=(latent_dim,))

decoder_outputs, decoder_states = decoder_gru(
decoder_inputs, initial_state=decoder_states_inputs)

decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model(
[decoder_inputs] + decoder_states_inputs,
[decoder_outputs] + decoder_states) #error
```

“”””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””””

I got mistake at the end of the code which I type "#error".

And this is what I got:ValueError: Dimensions must be equal, but are 98 and 256 for 'add_3' (op: 'Add') with input shapes: [1,?,?,98], [?,256].

Sorry to hear that, I have some suggestions that might help:
https://machinelearningmastery.com/faq/single-faq/can-you-read-review-or-debug-my-code

**Koohong Kang** December 2, 2019 at 5:49 pm # REPLY

Have you ever checked the accuracy of the validation dataset? When I set the model.fit() with validation_split=0.2, the accuracy is under 88%. By the way, I wanted to improve that figure, so I stacked the LSTM layers. I also tried to use dropout and other optimizer. However, it was getting worse. Could you give me any comments to improve the result? Moreover, for teacher-forcing, we do not need to set the initial_state of the encoder LSTM?

**Jason Brownlee** December 3, 2019 at 4:49 am # REPLY

Yes, I have suggestions here:
https://machinelearningmastery.com/start-here/#better

**Koohong Kang** December 3, 2019 at 12:55 pm # REPLY

Thank you very much for your swift reply. By the way, I am wondering if you have tried to improve the performance of this model (LSTM seq2seq) on this post.

**Jason Brownlee** December 3, 2019 at 1:34 pm # REPLY

Yes, the suggestions there are full of good ideas, perhaps start with this:
https://machinelearningmastery.com/improve-deep-learning-performance/

**mohammadreza** December 18, 2019 at 1:04 pm # REPLY

Hi.
How can i add “attention” to this code?
I can’t apply “decoderattention” to this code.
Thanks.

**Jason Brownlee** December 18, 2019 at 1:29 pm # REPLY

**Eli** January 22, 2020 at 7:31 am # REPLY

Hi Jason,

I am curious about the '[decoder_inputs] + decoder_states_inputs' line common in Encoder Decoder models. Is this syntax to initialize the decoder inputs with the cell state of the encoder, such that the '+' after the '[decoder_inputs]' allows us to do this initialization?

As always, I'm blown away by how thorough and informative your articles are. I'm sincerely grateful for your work.

**Jason Brownlee** January 22, 2020 at 1:52 pm # REPLY

It concatenates the lists/arrays.

**Anirban Ray** February 10, 2020 at 10:11 pm # REPLY

Hi Jason,

Can you please tell me how to pass varying length integer-coded inputs to the following Model?

il = Input(shape=(None,))
el = Embedding(len(train_vocabulary), 64)(il)
gl = GRU(64)(el)
ol = Dense(3, activation="sigmoid")(gl)
classification_model = Model(inputs=il, outputs=ol)

I prepared the inputs as a list of lists, where sublists are of different length, but I am unable to pass these to the model. However, if I use padding, everything works fine. So, is it not possible to use varying length without padding?

**Jason Brownlee** February 11, 2020 at 5:13 am # REPLY

You must use zero padding.

Perhaps see this:
https://machinelearningmastery.com/data-preparation-variable-length-input-sequences-sequence-prediction/

**Charles David** April 20, 2020 at 4:57 pm # REPLY

How can I get around this?

**Jason Brownlee** April 21, 2020 at 5:47 am # REPLY

Perhaps use a smaller vocab?
Perhaps try running on a machine with more RAM (AWS ec2)?

**Charles David** April 24, 2020 at 5:28 pm # REPLY

Thanks Jason.

I had one more doubt:
Why do you need the inference model or decoding model for testing? How does the training help. The only thing that I can see is that it helps in deciding the encoder inputs. Is there something I am missing. It seems to me that the entire purpose of training is lost.

**Savitha Ramesh** July 7, 2020 at 3:18 am # REPLY

Hi Jason
Are there any pretrained seq to seq models for speech to text. Need to know how to use by transfer learning technique. Any links which has examples?

**Jason Brownlee** July 7, 2020 at 6:43 am # REPLY

I expect there is, perhaps check out:
https://huggingface.co/

**Michael** July 14, 2020 at 1:56 pm # REPLY

Hello Jason, just a small question I'd like to ask:

In the inference model part, when a model is initialised, the decoder hidden state and cell state are to be returned along with an output, which `I totally understand why. The part which I'm not really clear about is addition between decoder_outputs and decoder_states. I suppose both decoder_outputs and decoder_state both represent some distinct values which will be used in the next recursive call of the decoder, so what does it mean to add them together as a single quantity? Thanks.

The h is called the hidden state, but it is really just the output of the layer.
The c is the internal hidden state of the layer.

These come from the previous time step of the encoder and are passed to the decoder as input/initializaiton.

I hope that helps, perhaps try tracing the variables to see how they are used.

**Chung-Hao Ku** July 14, 2020 at 2:21 pm # REPLY

I might have a clue, correct me if I'm wrong. Is it because it has to do with the way LSTM cells are computed?

**Jason Brownlee** July 15, 2020 at 8:11 am # REPLY

Correct, I recommend checking the original paper on the topic.

**Victor** August 13, 2020 at 5:48 am # REPLY

Hi! Jason, what is the difference between the seq2seq architecture from this article and the seq2seq architecture in your LSTMs with Python book page 111, chapter 9.3?
The book's architecture uses RepeatVector and TimeDistributed layers. The above architecture uses return states and creates an independent encoder and decoder. The results from the two architectures are the same? could you explain to me the difference and when is it better to choose one over the other? the two architectures are seq2seq and any could be used to language translation? Thanks in advance.

**Jason Brownlee** August 13, 2020 at 6:23 am # REPLY

Good question.

The example is in the book is simple and efficient and is based on an LSTM autoencoder. I recommend this approach.

The above exmaple is a lot more complex and is based on a dynamic rnn and a is a closer match to the original encoder-decoder paper.

Both appear to have similar performance in practice in my experience.

**Sagnik** November 16, 2020 at 11:59 pm # REPLY only on the last time step output of the lstm , and thus we should end up with only one character prediction instead of a whole sentence. Why is it working? Can you please explain Jason?

**Jason Brownlee** November 17, 2020 at 6:30 am # REPLY

The Dense will output one value for the the LSTM output time step.

**Nipun Agrawal** April 6, 2021 at 10:04 pm # REPLY

Hello,
In my sequence to sequence model i have to pass n number of datapoints parallely for the inference how to do it?

**Jason Brownlee** April 7, 2021 at 5:10 am # REPLY

One sample can have many time steps and many features.

**MS** June 27, 2021 at 7:12 pm # REPLY

Hi. Jason
We need to run the decoder inside a loop for each time_step(max_len) and then use the output of each decoder as an input to the next decoder. To implement this I apply a for loop. How did you implement this mechanism using your training code? Does your code takes care of the mechanism? If so how?

**MS** June 27, 2021 at 9:26 pm # REPLY

I got it Jason..U're using Teacher Forcing. Please correct me if I'm wrong. Very nice blog Jason.

**Jason Brownlee** June 28, 2021 at 7:58 am # REPLY

Yes.

**Jason Brownlee** June 28, 2021 at 7:58 am # REPLY

See this example:
https://machinelearningmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/

**MS** June 28, 2021 at 5:18 pm # REPLY

Thanx for ur time.

**Jason Brownlee** June 29, 2021 at 4:45 am # REPLY

You're welcome.

**Echo Echo** July 17, 2022 at 6:47 pm # REPLY

Hi Jason, you mentioned "The model defined for training has learned weights for this operation, but the structure of the model is not designed to be called recursively to generate one character at a time."

When defining the structure of inference model, can you point out which particular lines of code specified that we'll be using the learned weight of the trained model? It seems that the inference model's code didn't mention the trained model at all?

**James Carmichael** July 18, 2022 at 8:32 am # REPLY

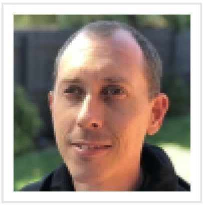
Hi Echo…the final code listing could be used to make actual predictions based upon the previous steps that define and train the model. This last part was not shown in this particular case. I would encourage you to extend the example for your own purposes and let us know your findings on the accuracy of the model.

## Leave a Reply

Name (required)

Email (will not be published) (required)

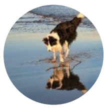
SUBMIT COMMENT



**Welcome!**
I'm *Jason Brownlee* PhD
and I **help developers** get results with **machine learning**.
Read more

**Never miss a tutorial:**

**Picked for you:**


How to Develop a Deep Learning Photo Caption Generator from Scratch

How to Use Word Embedding Layers for Deep Learning with Keras

How to Develop a Neural Machine Translation System from Scratch

Deep Convolutional Neural Network for Sentiment Analysis (Text Classification)

## Loving the Tutorials?

The Deep Learning for NLP EBook is
where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE