```
# LSTM with Variable Length Input Sequences to One Character Output
# https://machinelearningmastery.com/understanding-stateful-lstm-recurrent-neural-r

# LSTM (Many to One Multiple Character Feature)
# ==============================================

import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.utils import np_utils
from keras.preprocessing.sequence import pad_sequences
from theano.tensor.shared_randomstreams import RandomStreams

# fix random seed for reproducibility
numpy.random.seed(7)

# define the raw dataset
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# create mapping of characters to integers (0-25) and the reverse
char_to_int = dict((c, i) for i, c in enumerate(alphabet))
int_to_char = dict((i, c) for i, c in enumerate(alphabet))

# prepare the dataset of input to output pairs encoded as integers
num_inputs = 100
max_len = 5
dataX = []
dataY = []

for i in range(num_inputs):
  start = numpy.random.randint(len(alphabet)-2)
  end = numpy.random.randint(start, min(start+max_len,len(alphabet)-1))
  sequence_in = alphabet[start:end+1]
  sequence_out = alphabet[end + 1]
  dataX.append([char_to_int[char] for char in sequence_in])
  dataY.append(char_to_int[sequence_out])
  print sequence_in, '->', sequence_out

# convert list of lists to array and pad sequences if needed
X = pad_sequences(dataX, maxlen=max_len, dtype='float32')

# reshape X to be [samples, time steps, features]
X = numpy.reshape(X, (X.shape[0], max_len, 1))

# normalize
X = X / float(len(alphabet))

# one hot encode the output variable
y = np_utils.to_categorical(dataY)
```

```

# create and fit the model
batch_size = 1
model = Sequential()
model.add(LSTM(32, input_shape=(X.shape[1], 1)))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accurac
model.fit(X, y, epochs=500, batch_size=batch_size, verbose=1)

# summarize performance of the model
scores = model.evaluate(X, y, verbose=0)
print("Model Accuracy: %.2f%%" % (scores[1]*100))

# demonstrate some model predictions
for i in range(20):
    pattern_index = numpy.random.randint(len(dataX))
    pattern = dataX[pattern_index]
    print('pattern : {}'.format(pattern))
    x = pad_sequences([pattern], maxlen=max_len, dtype='float32')
    x = numpy.reshape(x, (1, max_len, 1))
    x = x / float(len(alphabet))
    prediction = model.predict(x, verbose=0)
    index = numpy.argmax(prediction)
    result = int_to_char[index]
    seq_in = [int_to_char[value] for value in pattern]
    print seq_in, "->", result
```

```
 - 11s - loss: 0.1466 - acc: 0.9710
Epoch 426/500
 - 11s - loss: 0.1956 - acc: 0.9450
Epoch 427/500
 - 11s - loss: 0.1039 - acc: 0.9830
Epoch 428/500
 - 11s - loss: 0.1049 - acc: 0.9830
Epoch 429/500
 - 11s - loss: 0.1070 - acc: 0.9730
Epoch 430/500
 - 11s - loss: 0.1081 - acc: 0.9750
Epoch 431/500
 - 11s - loss: 0.1075 - acc: 0.9720
Epoch 432/500
 - 11s - loss: 0.1093 - acc: 0.9740
Epoch 433/500
 - 11s - loss: 0.1089 - acc: 0.9730
Epoch 434/500
 - 11s - loss: 0.3108 - acc: 0.9260
Epoch 435/500
 - 11s - loss: 0.1020 - acc: 0.9820
Epoch 436/500
 - 11s - loss: 0.1008 - acc: 0.9810
Epoch 437/500
 - 11s - loss: 0.1018 - acc: 0.9790
Epoch 438/500
 - 11s - loss: 0.1020 - acc: 0.9810
Epoch 439/500
 - 11s - loss: 0.1066 - acc: 0.9760
Epoch 440/500
 - 11s - loss: 0.1029 - acc: 0.9740
Epoch 441/500
 - 11s - loss: 0.1081 - acc: 0.9780
Epoch 442/500
 - 11s - loss: 0.1066 - acc: 0.9790
Epoch 443/500
 - 11s - loss: 0.1037 - acc: 0.9770
Epoch 444/500
 - 11s - loss: 0.1061 - acc: 0.9780
Epoch 445/500
 - 11s - loss: 0.2312 - acc: 0.9430
Epoch 446/500
 - 11s - loss: 0.0990 - acc: 0.9820
Epoch 447/500
 - 11s - loss: 0.0983 - acc: 0.9780
Epoch 448/500
 - 11s - loss: 0.1015 - acc: 0.9810
Epoch 449/500
 - 11s - loss: 0.1009 - acc: 0.9780
Epoch 450/500
 - 11s - loss: 0.1031 - acc: 0.9810
Epoch 451/500
 - 11s - loss: 0.1052 - acc: 0.9750
Epoch 452/500
 - 11s - loss: 0.1025 - acc: 0.9820
Epoch 453/500
 - 11s - loss: 0.1039 - acc: 0.9780
Epoch 454/500
```

```
 - 11s - loss: 0.1021 - acc: 0.9770
Epoch 455/500
 - 11s - loss: 0.1932 - acc: 0.9620
Epoch 456/500
 - 11s - loss: 0.1076 - acc: 0.9760
Epoch 457/500
 - 11s - loss: 0.0963 - acc: 0.9810
Epoch 458/500
 - 11s - loss: 0.0988 - acc: 0.9830
Epoch 459/500
 - 11s - loss: 0.0985 - acc: 0.9800
Epoch 460/500
 - 11s - loss: 0.1002 - acc: 0.9750
Epoch 461/500
 - 11s - loss: 0.1001 - acc: 0.9770
Epoch 462/500
 - 11s - loss: 0.1010 - acc: 0.9780
Epoch 463/500
 - 11s - loss: 0.0987 - acc: 0.9770
Epoch 464/500
 - 11s - loss: 0.1002 - acc: 0.9750
Epoch 465/500
 - 11s - loss: 0.0989 - acc: 0.9780
Epoch 466/500
 - 11s - loss: 0.0996 - acc: 0.9750
Epoch 467/500
 - 11s - loss: 0.3353 - acc: 0.9520
Epoch 468/500
 - 11s - loss: 0.1601 - acc: 0.9770
Epoch 469/500
 - 11s - loss: 0.1488 - acc: 0.9740
Epoch 470/500
 - 11s - loss: 0.0935 - acc: 0.9890
Epoch 471/500
 - 11s - loss: 0.0929 - acc: 0.9840
Epoch 472/500
 - 11s - loss: 0.0942 - acc: 0.9800
Epoch 473/500
 - 11s - loss: 0.0951 - acc: 0.9820
Epoch 474/500
 - 11s - loss: 0.0962 - acc: 0.9790
Epoch 475/500
 - 11s - loss: 0.0952 - acc: 0.9840
Epoch 476/500
 - 11s - loss: 0.0970 - acc: 0.9820
Epoch 477/500
 - 11s - loss: 0.0966 - acc: 0.9770
Epoch 478/500
 - 11s - loss: 0.0960 - acc: 0.9810
Epoch 479/500
 - 11s - loss: 0.0958 - acc: 0.9770
Epoch 480/500
 - 11s - loss: 0.2241 - acc: 0.9590
Epoch 481/500
 - 11s - loss: 0.0878 - acc: 0.9860
Epoch 482/500
 - 11s - loss: 0.0901 - acc: 0.9830
Epoch 483/500
```

```
Epoch 483/500
 - 10s - loss: 0.0904 - acc: 0.9830
Epoch 484/500
 - 11s - loss: 0.0919 - acc: 0.9830
Epoch 485/500
 - 10s - loss: 0.0929 - acc: 0.9810
Epoch 486/500
 - 11s - loss: 0.0911 - acc: 0.9820
Epoch 487/500
 - 11s - loss: 0.0931 - acc: 0.9780
Epoch 488/500
 - 11s - loss: 0.0983 - acc: 0.9810
Epoch 489/500
 - 11s - loss: 0.0924 - acc: 0.9780
Epoch 490/500
 - 11s - loss: 0.0922 - acc: 0.9830
Epoch 491/500
 - 11s - loss: 0.2718 - acc: 0.9590
Epoch 492/500
 - 11s - loss: 0.1772 - acc: 0.9720
Epoch 493/500
 - 11s - loss: 0.0877 - acc: 0.9830
Epoch 494/500
 - 11s - loss: 0.0881 - acc: 0.9860
Epoch 495/500
 - 11s - loss: 0.0894 - acc: 0.9870
Epoch 496/500
 - 11s - loss: 0.0885 - acc: 0.9770
Epoch 497/500
 - 11s - loss: 0.0900 - acc: 0.9850
Epoch 498/500
 - 11s - loss: 0.0891 - acc: 0.9810
Epoch 499/500
 - 11s - loss: 0.0903 - acc: 0.9830
Epoch 500/500
 - 11s - loss: 0.0881 - acc: 0.9830
Model Accuracy: 98.30%
pattern : [19, 20, 21, 22, 23]
['T', 'U', 'V', 'W', 'X'] -> Y
pattern : [21, 22, 23, 24]
['V', 'W', 'X', 'Y'] -> Z
pattern : [0, 1, 2, 3]
['A', 'B', 'C', 'D'] -> E
pattern : [2]
['C'] -> D
pattern : [10, 11, 12, 13]
['K', 'L', 'M', 'N'] -> O
pattern : [1]
['B'] -> C
pattern : [2, 3, 4, 5, 6]
['C', 'D', 'E', 'F', 'G'] -> H
pattern : [16, 17]
['Q', 'R'] -> S
pattern : [19, 20, 21, 22, 23]
['T', 'U', 'V', 'W', 'X'] -> Y
pattern : [3, 4, 5, 6, 7]
['D', 'E', 'F', 'G', 'H'] -> I
pattern : [1, 2, 3, 4, 5]
```