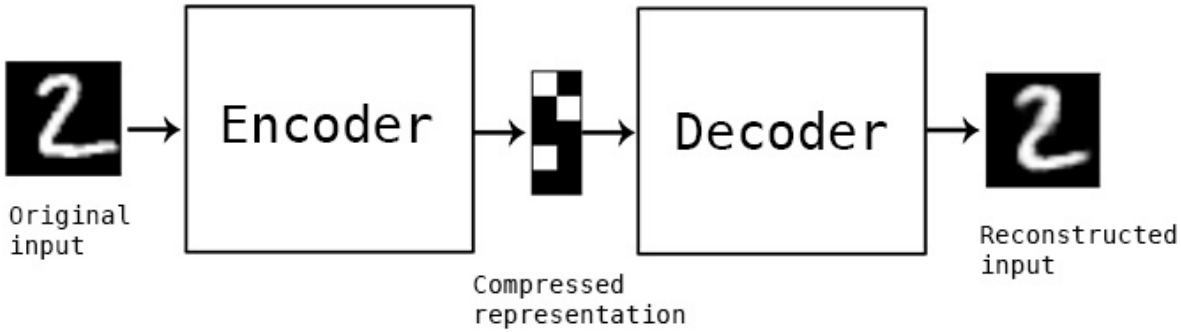
Encoder-Decoder Model for Sequence Prediction

What are autoencoders?



"Autoencoding" is a data compression algorithm where the compression and decompression functions are 1) data-specific, 2) lossy, and 3) learned automatically from examples rather than engineered by a human. Additionally, in almost all contexts where the term "autoencoder" is used, the compression and decompression functions are implemented with neural networks.

Here, we will use an LSTM neural network to learn the encoding of a five digit sequence.

In [1]:
```
# load libraries
import os
import pandas as pd
import numpy as np
from numpy import array
from numpy import array_equal
from numpy import argmax
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.utils import to_categorical
from keras.models import Model
from keras.layers import Input, LSTM, Dense, Dropout
from keras.callbacks import ModelCheckpoint
```

```
Using TensorFlow backend.
```

In [2]:
```
# set random seed
seed = 777
np.random.seed(seed)
```

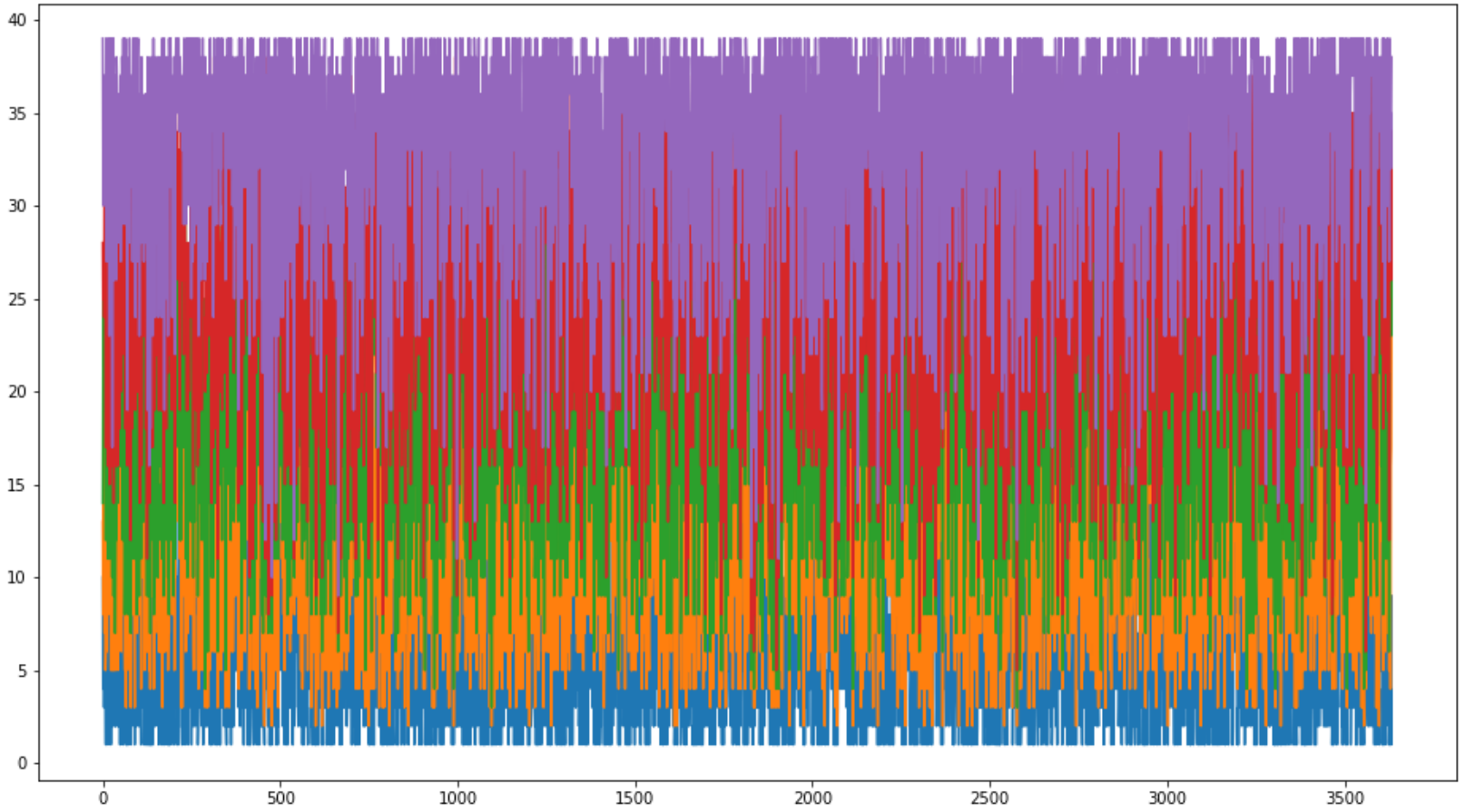
In [3]:
```
series = pd.read_csv('/Users/brent/Sandbox/ai/data/five_sequence.csv', engine='python', header=None)
series.head(10)
```

Out[3]:

	0	1	2	3	4
0	10	13	14	28	39
1	4	8	21	25	33
2	7	18	19	27	30
3	12	17	25	26	30
4	3	12	22	24	37
5	6	7	20	33	34
6	24	31	32	33	36
7	17	21	30	33	36
8	3	5	19	24	32
9	1	6	12	17	34

In [4]:
```
# review dataset
data = series.values
print("Data shape: " + str(data.shape))
print(data)
plt.figure(figsize=(16,9))
plt.plot(data)
plt.show()
```

```
Data shape: (3633, 5)
[[10 13 14 28 39]
 [ 4  8 21 25 33]
 [ 7 18 19 27 30]
 ...
 [ 4 16 26 29 32]
 [ 1 17 23 26 35]
 [ 9 24 31 34 38]]
```



In [5]:
```
# create a base list of the series data
mylist = []
for i in range(len(data)):
    temp = data[i]
    merge = temp.tolist()
    mylist.extend(merge)
print(len(mylist))
print(mylist)
```

```
18165
[10, 13, 14, 28, 39, 4, 8, 21, 25, 33, 7, 18, 19, 27, 30, 12, 17, 25, 26, 30, 3, 12, 22, 24, 37, 6, 7, 20,
33, 34, 24, 31, 32, 33, 36, 17, 21, 30, 33, 36, 3, 5, 19, 24, 32, 1, 6, 12, 17, 34, 6, 24, 30, 37, 39, 3, 6,
21, 22, 27, 5, 9, 15, 18, 29, 6, 11, 25, 33, 36, 8, 10, 15, 16, 30, 1, 8, 11, 20, 29, 5, 17, 20, 30, 39, 2,
24, 29, 36, 39, 18, 20, 21, 22, 23, 7, 10, 12, 15, 23, 7, 9, 11, 31, 39, 6, 17, 20, 27, 32, 9, 10, 21, 25,
27, 1, 5, 14, 31, 34, 10, 28, 32, 35, 39, 8, 18, 27, 29, 32, 6, 18, 22, 26, 39, 4, 5, 10, 16, 17, 14, 29, 31,
35, 38, 2, 10, 28, 29, 38, 10, 11, 17, 21, 39, 4, 5, 9, 14, 28, 3, 11, 20, 23, 26, 5, 12, 26, 30, 34, 18, 22,
27, 29, 37, 2, 5, 7, 28, 33, 4, 10, 11, 21, 31, 2, 11, 21, 28, 37, 8, 9, 12, 16, 26, 4, 27, 31, 36, 38, 9,
21, 33, 34, 37, 9, 16, 22, 28, 35, 4, 5, 14, 16, 35, 2, 19, 21, 24, 36, 12, 21, 29, 30, 36, 2, 22, 24, 26,
30, 19, 21, 23, 27, 28, 12, 15, 16, 18, 36, 6, 15, 19, 30, 34, 1, 23, 26, 28, 33, 4, 14, 23, 24, 37, 3, 11,
16, 20, 27, 4, 6, 23, 24, 33, 11, 18, 27, 34, 35, 5, 22, 23, 34, 38, 8, 12, 35, 36, 37, 2, 8, 18, 22, 33, 1,
10, 12, 25, 35, 3, 22, 35, 38, 39, 4, 14, 24, 29, 34, 21, 22, 27, 30, 34, 8, 23, 26, 27, 29, 2, 6, 8, 26, 35,
3, 5, 10, 18, 39, 5, 10, 18, 21, 39, 13, 23, 31, 36, 38, 1, 3, 11, 34, 36, 10, 17, 18, 35, 38, 1, 9, 22, 32,
34, 9, 22, 29, 32, 34, 3, 7, 12, 16, 19, 2, 6, 8, 19, 36, 1, 9, 15, 19, 37, 7, 10, 13, 28, 37, 12, 14, 24,
25, 28, 3, 5, 13, 34, 39, 13, 17, 22, 31, 39, 7, 14, 16, 36, 37, 7, 24, 29, 33, 36, 1, 3, 7, 12, 37, 7, 10,
14, 20, 33, 7, 12, 30, 31, 34, 2, 9, 11, 18, 31, 9, 15, 26, 33, 36, 13, 14, 18, 26, 27, 5, 10, 27, 30, 32, 3,
8, 12, 31, 37, 4, 10, 19, 38, 39, 3, 17, 18, 28, 31, 4, 7, 11, 32, 39, 1, 3, 19, 30, 35, 5, 24, 26, 27, 34,
3, 8, 15, 19, 24, 19, 25, 30, 33, 39, 2, 28, 29, 31, 34, 11, 13, 15, 21, 27, 2, 7, 8, 32, 37, 7, 8, 13, 25,
27, 3, 9, 18, 20, 38, 7, 9, 10, 24, 39, 8, 18, 25, 30, 39, 9, 16, 18, 27, 32, 7, 9, 10, 31, 36, 2, 8, 9, 16,
23, 7, 8, 13, 14, 29, 4, 10, 12, 27, 31, 9, 11, 18, 31, 34, 3, 9, 26, 29, 31, 1, 4, 11, 16, 28, 8, 25, 27,
29, 38, 3, 17, 27, 29, 33, 18, 21, 25, 27, 35, 19, 26, 30, 33, 36, 5, 16, 22, 27, 31, 7, 10, 22, 31, 32, 3,
12, 13, 23, 35, 10, 15, 22, 26, 27, 1, 12, 28, 32, 34, 4, 5, 22, 32, 33, 14, 16, 23, 24, 28, 10, 16, 25, 35,
36, 1, 6, 14, 18, 29, 4, 10, 18, 25, 29, 23, 24, 25, 35, 36, 10, 11, 16, 21, 30, 1, 6, 9, 11, 35, 2, 13, 15,
18, 19, 4, 8, 11, 34, 38, 1, 9, 16, 24, 30, 6, 7, 28, 30, 33, 3, 21, 27, 31, 33, 1, 5, 9, 14, 16, 2, 10, 17,
29, 36, 1, 20, 22, 29, 34, 1, 4, 7, 17, 18, 12, 15, 19, 31, 35, 1, 10, 19, 27, 37, 8, 15, 16, 18, 38, 8, 14,
15, 19, 37, 1, 22, 26, 31, 32, 1, 9, 15, 17, 25, 1, 16, 18, 20, 23, 5, 9, 12, 19, 32, 26, 30, 36, 37, 39, 2,
4, 8, 28, 31, 10, 17, 27, 32, 35, 5, 8, 9, 36, 38, 2, 7, 11, 19, 33, 8, 12, 18, 21, 24, 16, 26, 34, 36, 37,
21, 22, 24, 28, 30, 4, 15, 20, 26, 36, 7, 16, 20, 22, 28, 8, 23, 29, 35, 37, 3, 17, 20, 31, 38, 2, 8, 10, 34,
38, 4, 7, 15, 22, 32, 13, 14, 15, 22, 24, 1, 6, 12, 19, 33, 5, 12, 32, 33, 34, 7, 15, 16, 22, 32, 2, 6, 27,
31, 33, 15, 20, 22, 31, 36, 7, 10, 24, 31, 38, 5, 6, 17, 19, 35, 10, 12, 22, 25, 27, 9, 13, 17, 19, 24, 1, 8,
28, 33, 36, 1, 3, 11, 36, 39, 5, 7, 23, 25, 29, 3, 21, 23, 25, 27, 8, 9, 23, 33, 38, 4, 6, 9, 14, 36, 6, 10,
19, 25, 39, 1, 9, 12, 18, 19, 22, 24, 30, 33, 38, 2, 6, 12, 26, 27, 2, 12, 25, 31, 38, 13, 16, 19, 24, 31, 5,
15, 16, 18, 37, 14, 15, 16, 24, 25, 7, 14, 23, 24, 34, 1, 11, 16, 26, 34, 10, 14, 18, 22, 32, 2, 5, 10, 21,
23, 1, 7, 22, 33, 38, 15, 17, 22, 31, 36, 5, 7, 8, 22, 39, 1, 14, 25, 30, 39, 7, 28, 31, 34, 38, 1, 16, 18,
26, 31, 2, 13, 32, 38, 39, 9, 22, 25, 27, 32, 3, 12, 27, 33, 37, 6, 11, 12, 19, 37, 21, 23, 27, 31, 35, 7, 8,
19, 20, 24, 14, 23, 27, 28, 31, 23, 28, 29, 33, 36, 17, 21, 24, 29, 33, 1, 8, 9, 19, 31, 13, 19, 21, 34, 38,
4, 5, 29, 30, 35, 14, 16, 24, 27, 32, 7, 10, 11, 14, 22, 4, 31, 33, 36, 37, 13, 20, 22, 30, 38, 3, 13, 17,
26, 39, 10, 29, 33, 34, 38, 1, 25, 26, 30, 35, 5, 23, 27, 33, 38, 7, 17, 29, 30, 36, 5, 16, 25, 31, 39, 1,
16, 19, 28, 39, 10, 19, 26, 32, 39, 10, 13, 20, 27, 34, 11, 20, 32, 33, 35, 10, 14, 15, 24, 34, 6, 11, 12,
19, 38, 14, 17, 33, 34, 37, 10, 11, 12, 28, 34, 8, 21, 30, 33, 35, 2, 3, 14, 34, 37, 2, 11, 15, 32, 35, 1, 3,
25, 26, 33, 11, 20, 23, 27, 34, 8, 16, 24, 30, 34, 5, 10, 17, 29, 34, 9, 18, 20, 22, 29, 10, 26, 31, 35, 38,
11, 21, 23, 27, 39, 1, 9, 21, 28, 32, 25, 26, 32, 33, 37, 4, 17, 27, 28, 35, 3, 6, 11, 28, 29, 4, 17, 20, 26,
37, 2, 13, 15, 27, 35, 7, 16, 17, 19, 39, 4, 22, 23, 26, 39, 3, 7, 10, 19, 39, 1, 7, 13, 28, 33, 6, 12, 17,
23, 39, 3, 12, 17, 20, 36, 6, 24, 25, 26, 31, 9, 21, 22, 28, 35, 10, 19, 21, 25, 30, 5, 12, 18, 26, 39, 5, 9,
19, 25, 30, 8, 10, 23, 36, 37, 2, 8, 13, 30, 38, 1, 14, 22, 24, 25, 2, 10, 27, 35, 39, 12, 14, 19, 24, 37, 4,
12, 26, 32, 39, 10, 12, 13, 24, 27, 3, 6, 10, 15, 30, 2, 5, 13, 32, 39, 5, 6, 20, 21, 35, 15, 16, 17, 20, 28,
```

In [6]:
```
# create the initial source set
def create_initial_source(data, inlength):
    isource = list()
    for i in range(inlength):
        s = data[i]
        isource.append(s)
    return isource
```

In [7]:
```
# create target sets and subsequent source sets
def create_targets(data, outlength):
    target, targ, tar = list(), list(), list()

    for j in range(int((len(data) - outlength) / outlength)):
        k = j * 5
        del tar[:]
        for i in range(outlength):
            t = data[i + 5 + k]
            tar.append(t)
        targ.extend(tar)
    p = 0
    for q in range(int(len(targ) / outlength)):
        target.insert(q, targ[p:p+5])
        p = p + 5
    return target
```

In [8]:
```
# create shifted target sequence
def shifted_target(target, outlength):
    sh_target, target_in = list(), list()
    temp_in = [0] * outlength
    for i in range(len(target)):
        templ = target[i]
        for j in range(1,5):
            temp_in[j] = templ[j-1]
        target_in.extend(temp_in)
    v = 0
    for q in range(int(len(target_in) / outlength)):
        sh_target.insert(q, target_in[v:v+5])
        v = v + 5
    return sh_target
```

In [9]:
```
# create source items
def create_source(data, outlength, target):
    source = list()
    isource = create_initial_source(data, outlength)
    source.append(isource)
    j = 1
    for i in range(int((len(data) / outlength)) - 2):
        source.insert(j, target[i])
        j = j + 1
    return source
```

In [10]:
```
# create one-hot vectors for each number
def create_onehot(data, n_features):
    hot_list = to_categorical(data, num_classes=n_features)
    return hot_list
```

In [11]:
```
# one-hot decoder
def one_hot_decode(encode_seq):
    return [argmax(vector) for vector in encode_seq]
```

In [12]:
```
# prepare dataset for encoder-decoder models
def create_dataset(data, inlength, outlength, n_features):
    # create target set
    target = create_targets(data, outlength)
    # create shifted target set
    shift_target = shifted_target(target, outlength)
    # create source set
    source = create_source(data, outlength, target)
    # one-hot encode data values
    target_encoded = create_onehot(target, n_features)
    shift_target_encoded = create_onehot(shift_target, n_features)
    source_encoded = create_onehot(source, n_features)
    # define return values
    X1 = array(source_encoded)
    X2 = array(shift_target_encoded)
    y = array(target_encoded)
    return X1, X2, y
```

In [13]:
```
# create your dataset for the encoder-decoder models
inlength = 5      # input sequence length
outlength = 5     # output sequence length
cardinality = 39  # cardinality of the dataset
n_features = cardinality + 1
X1, X2, y = create_dataset(mylist, inlength, outlength, n_features)
print("Source shape: ", X1.shape)
print("Shifted target shape: ", X2.shape)
print("Target shape: ", y.shape)
```

```
Source shape:  (3632, 5, 40)
Shifted target shape:  (3632, 5, 40)
Target shape:  (3632, 5, 40)
```

In [14]:
```
# test dataset encoding / decoding
print("Source encoded: \n", X1[2])
print("Shifted target encoded: \n", X2[2])
print("Target encoded: \n", y[2])
src_decode = one_hot_decode(X1[2])
shift_target_decode = one_hot_decode(X2[2])
target_decode = one_hot_decode(y[2])
print("---------------------------------")
print("Source encoded: ", src_decode)
print("Shifted target encoded: ", shift_target_decode)
print("Target encoded: ", target_decode)
```

```
Source encoded:
 [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
Shifted target encoded:
 [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Target encoded:
 [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
---------------------------------
Source encoded:  [7, 18, 19, 27, 30]
Shifted target encoded:  [0, 12, 17, 25, 26]
Target encoded:  [12, 17, 25, 26, 30]
```

In [15]:
```
# create the training, inference_encoder and inference_decoder models
def define_models(n_input, n_output, n_units):
    # define training encoder
    encoder_inputs = Input(shape=(None, n_input))
    encoder = LSTM(n_units, return_state=True)
    encoder_outputs, state_h, state_c = encoder(encoder_inputs)
    encoder_states = [state_h, state_c]
    # define training decoder
    decoder_inputs = Input(shape=(None, n_output))
    decoder_lstm = LSTM(n_units, return_sequences=True, return_state=True)
    decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
    decoder_dense = Dense(n_output, activation='softmax')
    decoder_outputs = decoder_dense(decoder_outputs)
    model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
    # define inference encoder
    encoder_model = Model(encoder_inputs, encoder_states)
    # define inference decoder
    decoder_state_input_h = Input(shape=(n_units,))
    decoder_state_input_c = Input(shape=(n_units,))
    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)
    # return all models
    return model, encoder_model, decoder_model
```

In [16]:
```
# generate target given source sequence
def predict_sequence(infenc, infdec, source, n_steps, cardinality):
    # encode
    state = infenc.predict(source)
    # start of sequence input
    target_seq = array([0.0 for _ in range(cardinality)]).reshape(1,1, cardinality)
    # collect predictions
    output = list()
    for t in range(n_steps):
        # predict the next number
        yhat, h, c = infdec.predict([target_seq] + state)
        # store prediction
        output.append(yhat[0,0,:])
        # update state
        state = [h, c]
        # update target sequence
        target_seq = yhat
    return array(output)
```

In [17]:
```
# define the checkpoint
filepath="weightsV2-{epoch:02d}-{acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='acc', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]
```

In [18]:
```
# instantiate the model instance
train, infenc, infdec = define_models(n_features, n_features, 256)
train.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
```

In [19]:
```
nb_epochs = 500
train.fit([X1, X2], y, epochs=nb_epochs, callbacks=callbacks_list)
```

```
Epoch 1/500
3616/3632 [============================>.] - ETA: 0s - loss: 3.2398 - acc: 0.0710Epoch 00001: acc improved fr
om -inf to 0.07087, saving model to weightsV2-01-0.0709.hdf5

/Users/brent/anaconda3/lib/python3.6/site-packages/keras/engine/topology.py:2344: UserWarning: Layer lstm_2 w
as passed non-serializable keyword arguments: {'initial_state': [<tf.Tensor 'lstm_1/while/Exit_2:0' shape=(?,
256) dtype=float32>, <tf.Tensor 'lstm_1/while/Exit_3:0' shape=(?, 256) dtype=float32>]}. They will not be inc
luded in the serialized model (and thus will be missing at deserialization time).
  str(node.arguments) + '. They will not be included '
```

```
3632/3632 [==============================] - 8s 2ms/step - loss: 3.2390 - acc: 0.0709
Epoch 2/500
3584/3632 [============================>.] - ETA: 0s - loss: 3.0309 - acc: 0.0945Epoch 00002: acc improved fr
om 0.07087 to 0.09438, saving model to weightsV2-02-0.0944.hdf5
3632/3632 [==============================] - 7s 2ms/step - loss: 3.0308 - acc: 0.0944
Epoch 3/500
3616/3632 [============================>.] - ETA: 0s - loss: 2.9273 - acc: 0.1046Epoch 00003: acc improved fr
om 0.09438 to 0.10474, saving model to weightsV2-03-0.1047.hdf5
3632/3632 [==============================] - 6s 2ms/step - loss: 2.9267 - acc: 0.1047
Epoch 4/500
3616/3632 [============================>.] - ETA: 0s - loss: 2.8329 - acc: 0.1168Epoch 00004: acc improved fr
om 0.10474 to 0.11674, saving model to weightsV2-04-0.1167.hdf5
3632/3632 [==============================] - 7s 2ms/step - loss: 2.8325 - acc: 0.1167
Epoch 5/500
3616/3632 [============================>.] - ETA: 0s - loss: 2.7611 - acc: 0.1257Epoch 00005: acc improved fr
om 0.11674 to 0.12572, saving model to weightsV2-05-0.1257.hdf5
3632/3632 [==============================] - 6s 2ms/step - loss: 2.7611 - acc: 0.1257
Epoch 6/500
3584/3632 [============================>.] - ETA: 0s - loss: 2.7239 - acc: 0.1329Epoch 00006: acc improved fr
om 0.12572 to 0.13320, saving model to weightsV2-06-0.1332.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.7235 - acc: 0.1332
Epoch 7/500
3616/3632 [============================>.] - ETA: 0s - loss: 2.6956 - acc: 0.1384Epoch 00007: acc improved fr
om 0.13320 to 0.13827, saving model to weightsV2-07-0.1383.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.6958 - acc: 0.1383
Epoch 8/500
3616/3632 [============================>.] - ETA: 0s - loss: 2.6815 - acc: 0.1392Epoch 00008: acc improved fr
om 0.13827 to 0.13910, saving model to weightsV2-08-0.1391.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.6817 - acc: 0.1391
Epoch 9/500
3584/3632 [============================>.] - ETA: 0s - loss: 2.6668 - acc: 0.1431Epoch 00009: acc improved fr
om 0.13910 to 0.14295, saving model to weightsV2-09-0.1430.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.6668 - acc: 0.1430
Epoch 10/500
3584/3632 [============================>.] - ETA: 0s - loss: 2.6509 - acc: 0.1517Epoch 00010: acc improved fr
om 0.14295 to 0.15187, saving model to weightsV2-10-0.1519.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.6511 - acc: 0.1519
Epoch 11/500
3616/3632 [============================>.] - ETA: 0s - loss: 2.6388 - acc: 0.1559Epoch 00011: acc improved fr
om 0.15187 to 0.15589, saving model to weightsV2-11-0.1559.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.6386 - acc: 0.1559
Epoch 12/500
3584/3632 [============================>.] - ETA: 0s - loss: 2.6260 - acc: 0.1570Epoch 00012: acc improved fr
om 0.15589 to 0.15683, saving model to weightsV2-12-0.1568.hdf5
3632/3632 [==============================] - 5s 1ms/step - loss: 2.6265 - acc: 0.1568
Epoch 13/500
```

Out[19]: <keras.callbacks.History at 0x1a18b854a8>

In [20]:
```
# load the best network weights
filename = "weights-1137-0.9993.hdf5"
train.load_weights(filename)
train.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
```

In [20]:
```
# test model set
src = [[2,5,6,12,27],[7,12,31,32,39],[4,23,25,30,39],[1,7,26,34,35],[3,4,24,31,35],[4,16,26,29,32],
[1,17,23,26,35],[9,24,31,34,38],[12,18,25,32,34],[11,13,18,23,30]]
correct = 0
for m in range(len(src)-1):
    X3, _, y1 = create_dataset(src[m], inlength, outlength, n_features)
    out_sequence = predict_sequence(infenc, infdec, X3, outlength, n_features)
    target = one_hot_decode(out_sequence)
    if (m < len(src)-1):
        yhat = src[m+1]
    else: yhat = src[m]
    print('yhat: %s, prediction: %s' % (yhat, target))
    if np.array_equal(yhat, target ):
                    correct += 1
print("The prediction accuracy is: %.2f%%" % (float(correct)/float(len(src))*100.0))
```

```
yhat: [7, 12, 31, 32, 39], prediction: [7, 12, 31, 32, 39]
yhat: [4, 23, 25, 30, 39], prediction: [4, 23, 25, 30, 39]
yhat: [1, 7, 26, 34, 35], prediction: [1, 7, 26, 34, 35]
yhat: [3, 4, 24, 31, 35], prediction: [3, 4, 24, 31, 35]
yhat: [4, 16, 26, 29, 32], prediction: [4, 16, 26, 29, 32]
yhat: [1, 17, 23, 26, 35], prediction: [1, 17, 23, 26, 35]
yhat: [9, 24, 31, 34, 38], prediction: [9, 24, 31, 34, 38]
yhat: [12, 18, 25, 32, 34], prediction: [2, 10, 18, 23, 35]
yhat: [11, 13, 18, 23, 30], prediction: [3, 4, 10, 13, 26]
The prediction accuracy is: 70.00%
```

In [22]:
```
# create new source input
src = [9,24,31,34,38]
X, _, _ = create_dataset(src, inlength, outlength, n_features)
out_sequence = predict_sequence(infenc, infdec, X, outlength, n_features)
output = one_hot_decode(out_sequence)
print("The predicted output sequence is: ", output)
```

```
The predicted output sequence is:  [4, 10, 21, 22, 35]
```

In [ ]: