In [1]:

```
import numpy
import tensorflow as tf
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
numpy.random.seed(7)
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
char_to_int = dict((c, i) for i, c in enumerate(alphabet))
int_to_char = dict((i, c) for i, c in enumerate(alphabet))
seq_length = 1
dataX = []
dataY = []

for i in range(0, len(alphabet) - seq_length, 1):
    seq_in = alphabet[i:i + seq_length]
    seq_out = alphabet[i + seq_length]
    dataX.append([char_to_int[char] for char in seq_in])
    dataY.append(char_to_int[seq_out])
    print(seq_in, '->', seq_out)
print(dataX)
print(dataY)
X = numpy.reshape(dataX, (len(dataX), seq_length, 1))
X =  tf.sort(X,axis=-1,direction='ASCENDING',name=None)
X= pad_sequences(X, maxlen=seq_length, dtype='float32')
# normalize
X = X / float(len(alphabet))
y = np_utils.to_categorical(dataY)
model = Sequential()
model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['ac
model.fit(X, y, epochs=500, batch_size=1, verbose=2)
scores = model.evaluate(X, y, verbose=0)
print("Model Accuracy: %.2f%%" % (scores[1]*100))
for pattern in dataX:
    x = numpy.reshape(pattern, (1, len(pattern), 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)
model.summary()
```

```
Using TensorFlow backend.
```

```
A -> B
B -> C
C -> D
D -> E
E -> F
F -> G
G -> H
H -> I
```

In [2]:

```
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
seq_length = 3
dataX = []
dataY = []
for i in range(0, len(alphabet) - seq_length, 1):
    seq_in = alphabet[i:i + seq_length]
    seq_out = alphabet[i + seq_length]
    dataX.append([char_to_int[char] for char in seq_in])
    dataY.append(char_to_int[seq_out])
    print(seq_in, '->', seq_out)
print(dataX)
print(dataY)
X = numpy.reshape(dataX, (len(dataX), 1, seq_length))
#X =  tf.sort(X,axis=-1,direction='ASCENDING',name=None)
#X= pad_sequences(X, maxlen=seq_length, dtype='float32')

X = X / float(len(alphabet))

y = np_utils.to_categorical(dataY)

model = Sequential()
model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['ac
model.fit(X, y, epochs=500, batch_size=1, verbose=2)
# summarize performance of the model
scores = model.evaluate(X, y, verbose=0)
print("Model Accuracy: %.2f%%" % (scores[1]*100))
# demonstrate some model predictions
for pattern in dataX:
    x = numpy.reshape(pattern, (1, 1, len(pattern)))
    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)
model.summary()
```

```
['H', 'I', 'J'] -> K
['I', 'J', 'K'] -> L
['J', 'K', 'L'] -> M
['K', 'L', 'M'] -> N
['L', 'M', 'N'] -> O
['M', 'N', 'O'] -> P
['N', 'O', 'P'] -> Q
['O', 'P', 'Q'] -> R
['P', 'Q', 'R'] -> S
['Q', 'R', 'S'] -> T
['R', 'S', 'T'] -> U
['S', 'T', 'U'] -> V
['T', 'U', 'V'] -> Y
['U', 'V', 'W'] -> Z
```

```
['V', 'W', 'X'] -> Z
['W', 'X', 'Y'] -> Z
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
```

In [ ]: