from numpy import array

from keras.preprocessing.text import Tokenizer

from keras.utils import to\_categorical

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Embedding

# generate a sequence from the model

def generate\_seq(model, tokenizer, seed\_text, n\_words):

in\_text, result = seed\_text, seed\_text

# generate a fixed number of words

for \_ in range(n\_words):

# encode the text as integer

encoded = tokenizer.texts\_to\_sequences([in\_text])[0]

encoded = array(encoded)

# predict a word in the vocabulary

yhat = model.predict\_classes(encoded, verbose=0)

# map predicted word index to word

out\_word = ''

for word, index in tokenizer.word\_index.items():

if index == yhat:

out\_word = word

break

# append to input

in\_text, result = out\_word, result + ' ' + out\_word

return result

# source text

data = """ Jack and Jill went up the hill\n

To fetch a pail of water\n

Jack fell down and broke his crown\n

And Jill came tumbling after\n """

# integer encode text

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts([data])

encoded = tokenizer.texts\_to\_sequences([data])[0]

# determine the vocabulary size

vocab\_size = len(tokenizer.word\_index) + 1

print('Vocabulary Size: %d' % vocab\_size)

# create word -> word sequences

sequences = list()

for i in range(1, len(encoded)):

sequence = encoded[i-1:i+1]

sequences.append(sequence)

print('Total Sequences: %d' % len(sequences))

# split into X and y elements

sequences = array(sequences)

X, y = sequences[:,0],sequences[:,1]

# one hot encode outputs

y = to\_categorical(y, num\_classes=vocab\_size)

# define model

model = Sequential()

model.add(Embedding(vocab\_size, 10, input\_length=1))

model.add(LSTM(50))

model.add(Dense(vocab\_size, activation='softmax'))

print(model.summary())

# compile network

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit network

model.fit(X, y, epochs=500, verbose=2)

# evaluate

print(generate\_seq(model, tokenizer, 'Jack', 6))