2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

from numpy import array

from keras.preprocessing.text import Tokenizer

from keras.utils import to\_categorical

from keras.preprocessing.sequence import pad\_sequences

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Embedding

# generate a sequence from a language model

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

in\_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]

# pre-pad sequences to a fixed length

encoded = pad\_sequences([encoded], maxlen=max\_length, padding='pre')

# predict probabilities for each word

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 += ' ' + out\_word

return in\_text

# 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 """

# prepare the tokenizer on the source text

tokenizer = Tokenizer()

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

# determine the vocabulary size

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

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

# create line-based sequences

sequences = list()

for line in data.split('\n'):

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

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

sequence = encoded[:i+1]

sequences.append(sequence)

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

# pad input sequences

max\_length = max([len(seq) for seq in sequences])

sequences = pad\_sequences(sequences, maxlen=max\_length, padding='pre')

print('Max Sequence Length: %d' % max\_length)

# split into input and output elements

sequences = array(sequences)

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

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

# define model

model = Sequential()

model.add(Embedding(vocab\_size, 10, input\_length=max\_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 model

print(generate\_seq(model, tokenizer, max\_length-1, 'Jack', 4))

print(generate\_seq(model, tokenizer, max\_length-1, 'Jill', 4))