МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Искусственные нейронные сети»

Тема: «Генерация текста на основе «Алисы в стране чудес»

Студент гр. 8382 _____ Кобенко В.П.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2021

Цель работы.

Реализовать генерацию текста на основе текста из сказки «Алиса в стране чудес».

Задачи.

- Ознакомиться с с генерацией текста
- Ознакомиться с системой Callback в Keras

Требования.

- Реализовать модель ИНС, которая будет генерировать текст
- Написать собственный CallBack, который будет показывать то, как генерируется текст во время обучения (то есть раз в какое-то количество эпох генерировать и выводить текст у необученной модели)
- Отследить процесс обучения при помощи TensorFlowCallBack, в отчете привести результаты и их анализ

Ход работы.

1. Была создана модель рекуррентной нейронной сети, для прогнозирования следующего символа на основе предыдущих (код программы представлен в приложении А). В качестве функции потерь была выбрана функция categorical_crossentropy.

Архитектура сети:

```
model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
```

2. Для отслеживания процесса генерации текста во время обучения нейронной сети был написан собственный обратный вызов Callback. Он выводит текст у необученной модели через некоторое количество эпох.

Отследим процесс обучения и рассмотрим тексты сгенерированные после 1,

9 и 18 эпох.

Результат после первой эпохи:

Seed:

" old, father william,' the young man said, 'and your hair has become very white; and yet you i "

oe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe

Результат после девятой эпохи:

Seed:

" t to the fifth bend, i think?' 'i had not!' cried the mouse, sharply and very angrily. 'a knot!' sai "

d the morke and the woole tas in the woole and the wooee the was oo the wooee the was oo

Результат после восемнадцатой эпохи:

Seed:

" the knave, 'i didn't write it, and they
can't prove i did: there's no name signed at the end.''if "

io ' said the manch hare.'ie tou t tou a taid to tey,' she maic thit have a lant lirtle toiee so the tent oo toe tiat shie the was aol the was aol the tas aoi the cadl she was aolin tote the thse oh the sas ho was aoong the had so the tabte bnd sae to theng tas toen in the was aol the was aol the tas aoi the cadl she was aolin tote the thse oh the sas ho was aoong the had so the tabte bnd sae to theng tas toen in the was aol the was aol the tas aoi the cadl she was aolin tote the thse oh the sas ho was aoong the had so the tabte bnd sae to theng tas toen in the was aol the was aol the tas aoi the cadl she was aolin tote the thse oh the sas ho was aoong the had so the tabte bnd sae to theng tas toen in the was aol the was aol the tas aoi the cadl she was aolin tote the

thse oh the sas ho was aoong the had so the tabte bnd sae to theng tas toen in the was aol the was aol the tas aoi the cadl she was aolin tote the thse oh the sas ho was aoong the had so the tabte bnd sae to

По сгенерированным текстам видно, что с увеличением количества эпох, увеличивается качество генерируемого текста. Так, после первой эпохи была сгенерирована повторяющаяся последовательность из трех букв, после девятой эпохи была также сгенерирована повторяющаяся последовательность, но уже с бóльшим количеством символов. После восемнадцатой эпохи текст был сгенерирован с меньшими повторениями, и с бóльшим количеством действительно существующих в английском языке слов (я нашла как минимум 14).

Вывод.

В ходе выполнения лабораторной работы была построена модель, генерирующая текст на основе книги «Алиса в стране чудес». Был также написан собственный CallBack, позволяющий отслеживать процесс обучения сети. В результате обучения сеть с каждой эпохой генерировала тексты с всё меньшим количеством повторений и бóльшим количеством существующих слов.

ПРИЛОЖЕНИЕ А

```
import numpy
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
import tensorflow.keras.utils as np_utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import Callback


filename = "wonderland.txt"
raw_text = open(filename).read()
raw_text = raw_text.lower()

chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))
int_to_char = dict((i, c) for i, c in enumerate(chars))

n_chars = len(raw_text)
n_vocab = len(chars)
print("Total Characters: ", n_chars)
print("Total Vocab: ", n_vocab)

seq_length = 100
dataX = []
dataY = []
for i in range(0, n_chars - seq_length, 1):
    seq_in = raw_text[i:i + seq_length]
    seq_out = raw_text[i + seq_length]
    dataX.append([char_to_int[char] for char in seq_in])
    dataY.append(char_to_int[seq_out])
n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)

X = numpy.reshape(dataX, (n_patterns, seq_length, 1))
X = X / float(n_vocab)
y = np_utils.to_categorical(dataY)

class Callback(Callback):
    def __init__(self, epochs):
        super(Callback, self).__init__()
        self.epochs = epochs

    def on_epoch_end(self, epoch, logs=None):
        if epoch in self.epochs:
            generate_sequence(self.model)

def generate_sequence(model):
    start = numpy.random.randint(0, len(dataX) - 1)
```

```
    pattern = dataX[start]
    print("Seed:")
    print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")
    for i in range(1000):
        x = numpy.reshape(pattern, (1, len(pattern), 1))
        x = x / float(n_vocab)
        prediction = model.predict(x, verbose=0)
        index = numpy.argmax(prediction)
        result = int_to_char[index]
        print(result, end='')
        pattern.append(index)
        pattern = pattern[1:len(pattern)]


model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')

filepath="weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save_best_only=True,
mode='min')
callbacks_list = [checkpoint, Callback([1, 9, 18])]

model.fit(X, y, epochs=20, batch_size=128, callbacks=callbacks_list)
```

```
        print("Seed:")
        print("\"", ''.join([int_to_char[value] for value in pattern]),
"\"")
        for i in range(1000):
            x = numpy.reshape(pattern, (1, len(pattern), 1))
            x = x / float(n_vocab)
            prediction = model.predict(x, verbose=0)
            index = numpy.argmax(prediction)
            result = int_to_char[index]
            print(result, end='')
            pattern.append(index)
            pattern = pattern[1:len(pattern)]

    model = Sequential()
    model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
    model.add(Dropout(0.2))
    model.add(Dense(y.shape[1], activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam')

    filepath="weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
    checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')
    #callbacks_list = [checkpoint, Callback([1, 9, 18])]
    callbacks_list = [checkpoint, Callback([1])]
    model.fit(X, y, epochs=1, batch_size=128, callbacks=callbacks_list)
```