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

ЛАБОРАТОРНАЯ РАБОТА №8

по дисциплине «Искусственные нейронные сети»

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

Студент гр. 7383 _____ Ханова Ю. А.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

Рекуррентные нейронные сети также могут быть использованы в качестве генеративных моделей.

Это означает, что в дополнение к тому, что они используются для прогнозных моделей (создания прогнозов), они могут изучать последовательности проблемы, а затем генерировать совершенно новые вероятные последовательности для проблемной области.

Подобные генеративные модели полезны не только для изучения того, насколько хорошо модель выявила проблему, но и для того, чтобы узнать больше о самой проблемной области.

Задачи.

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

Ход работы.

Была реализована модель сети для генерации текста. Так же был реализован класс `make_text(tensorflow.keras.callbacks.Callback)`, благодаря которому будет выводиться текст, сгенерированный на 1, 10, 15 и 20 эпохе обучения, для анализа процесса обучения.

После 1 эпохи генерируются набор из трех букв, две а и повторяющаяся буква «а»:

```
Seed:
" ! i shall be late!' (when she thought it over afterwards, it
occurred to her that she ought to have  "
toe  aa  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a
```

После 15 эпохи последовательность повторяющихся наборов букв, некоторые являются существующими словами:

```
Seed:
" of the court
was a table, with a large dish of tarts upon it: they looked so good,
that it made alic "
e tooe the was oo the tine tf thet she was oo the tan oo the tan oo the tan oo the tan oo the tan oo the tan oo the t
```

После 20 эпохи формируются предложения но все еще встречаются несуществующие слова:

```
Seed:
" taking alice by the hand, it hurried
off, without waiting for the end of the song.

'what trial is i "
are an a cand,' said the mucen, tho seil to her ferd and the toeee of the worle  the queen was the rabbit hare aro al
and the poeer was sooe tf the sabdit har and all the soeer,
and the puoer was thly hare ano ali thet, and the horpe the was soit it was toe car and the houpe of the sooe, and th
'the murh to the soier war so then that io ' said the mock turtle,
'and thet so soe why oo the say oo the soin.
the hatter was soon oo hor the hoore of the cour wf thnel hn an once, and the pooe tas sooe of the cour wh the too of
```

Вывод.

В ходе выполнения данной лабораторной работы была разработана модель сети, позволяющая генерировать тексты на основе «Алисы в стране чудес». В процессе обучения сеть генерировала текст, приближенный к реальному, уменьшалось количество повторений и увеличивалось количество существующих слов. Код программы представлен в приложении А.

Приложение А

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

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)


def generateSequence(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]
        seq_in = [int_to_char[value] for value in pattern]
        sys.stdout.write(result)
        pattern.append(index)
        pattern = pattern[1:len(pattern)]


class make_text(tensorflow.keras.callbacks.Callback):
    def __init__(self, epochs):
        super(make_text, self).__init__()
        self.epochs = epochs

    def on_epoch_end(self, epoch, logs={}):
        if epoch in self.epochs:
            generateSequence(model)


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')

model.fit(X, y, epochs=20, batch_size=128,
callbacks=[make_text([0, 9, 14, 19])])
```