

**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

Черницын П.А.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

## **Цель работы.**

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.

## **Порядок выполнения работы.**

1. Ознакомиться с генерацией текста;
2. Ознакомиться с системой Callback в Keras.

## **Требования.**

1. Реализовать модель ИНС, которая будет генерировать текст;
2. 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);
3. Отследить процесс обучения при помощи TensorFlowCallBack, в отчете привести результаты и их анализ.

## **Ход работы.**

Многие из классических текстов больше не защищены авторским правом. Это означает, что возможно скачать весь текст этих книг бесплатно и использовать их в экспериментах, например, при создании генеративных моделей. Возможно, лучшее место для получения доступа к бесплатным

книгам, которые больше не защищены авторским правом, это Проект Гутенберг.

В данной лабораторной работе будем использовать в качестве набора данных «Приключения Алисы в Стране Чудес» Льюиса Кэрролла. Мы собираемся изучить зависимости между символами и условные вероятности символов в последовательностях, чтобы мы могли, в свою очередь, генерировать совершенно новые и оригинальные последовательности символов.

Была построена нейронная сеть, разработанный код представлен в приложении А.

Была создана и обучена модель искусственной нейронной сети, решающая задачу генерации текста. Модель представлена на рис. 1.

```
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')
```

Рисунок 1 – Модель нейронной сети.

Был написан собственный обратный вызов (Callback), который позволит отслеживать то, как генерируется текст во время обучения, то есть в какое-то количество эпох генерировать и выводить текст у необученной модели:

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

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

Отследим процесс обучения и рассмотрим тексты сгенерированные после 7, 12, 22 и 30 эпох.

После седьмой эпохи сеть сгенерировала пустая последовательность, которая продемонстрирована на рис. 2

```
Epoch 00007: loss improved from 2.99666 to 2.99357, saving model to weights-improvement-07-2.9936.hdf5
Seed:
" it flashed across her mind that she had
never before seen a rabbit with either a waistcoat-pocket, o "
```

Рисунок 2 – Результат после 7 эпохи.

После 12 эпохи сеть сгенерировала повторяющуюся последовательность. Результат показан на рис. 3.

```
Epoch 00012: loss improved from 2.94469 to 2.91699, saving model to weights-improvement-12-2.9170.hdf5
Seed:
" as close behind it when she
turned the corner, but the rabbit was no longer to be seen: she found
he "
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao ao
ao ao ao ao ao ao ao
Done.
```

Рисунок 3 – Результат после 12 эпохи.

После 22 эпохи сеть сгенерировала опять повторяющуюся последовательность, но уже большей длины.

```
Epoch 00022: loss improved from 2.78891 to 2.78028, saving model to weights-improvement-22-2.7803.hdf5
Seed:
" so alice soon began
talking again. 'dinah'll miss me very much to-night, i should think!'
(dinah was "
tee toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe t
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 t
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 t
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 t
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 t
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 t
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 t
oe toe toe toe toe toe
Done.
```

После 30 эпохи в сгенерированном тексте присутствуют уже 3 уникальных слова.



## ПРИЛОЖЕНИЕ А

```
import numpy
from keras.callbacks import Callback
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
from os import listdir
from os import path

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

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

    def random_generate(_model, epoch=0):
        start = numpy.random.randint(0, len(dataX) - 1)
        pattern = dataX[start]
        print("Seed:")
        print("\n", ''.join([int_to_char[value] for value in pattern]), "\n")
        text = []
        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]
            text.append(result)
            print(result, end="")
            pattern.append(index)
            pattern = pattern[1:len(pattern)]
```

```

print("\nDone.")
with open('text_{}.txt'.format(epoch), 'w') as file:
    file.write(''.join(text))

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)

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, Mycallback([6, 11, 21, 29])]

model.fit(X, y, epochs=30, batch_size=128, callbacks=callbacks_list)

folder = '.'
filename = ''
min = 999999
for name in listdir(folder):
    full_name = path.join(folder, name)
    if path.isfile(full_name) and full_name.find('.hdf5') != -1:
        model_loss = int(full_name.split('.')[2])
        if min > model_loss:
            min = model_loss
            filename = full_name

print(filename)
model.load_weights(filename)
model.compile(loss='categorical_crossentropy', optimizer='adam')
random_generate(model)

```