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”

Студентка гр. 7382 _____ Еременко А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

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 к выполнению задания.

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 генирировать и выводить текст у необученной модели).
3. Отследить процесс обучения при помощи TensorFlowCallBack, в отчете привести результаты и их анализ.

Основные теоретические положения.

Многие из классических текстов больше не защищены авторским правом. Это означает, что вы можете 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 мы 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.

Ход работы.

1. Была построена и обучена модель нейронной сети, которая будет генерировать текст. Код предоставлен в приложении А.

Модель:

```
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_with_custom_print([1, 10, 15])]

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

2. Был написан собственный CallBack, который показывает то как генерируется текст во время обучения, то есть раз в какое-то количество эпох генирировать и выводить текст у необученной модели.

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

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

```python
def custom_print(custom_model):
    start = numpy.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")

    print("\"", ''.join([itc[value] for value in pattern]), "\"")

    for i in range(1000):
        x = numpy.reshape(pattern, (1, len(pattern), 1))
        x = x / float(n_vocab)
        prediction = custom_model.predict(x, verbose=0)
        index = numpy.argmax(prediction)
        result = itc[index]
        print(result, end='')
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
```

3. Отследим процесс обучения и рассмотрим тексты сгенерированные после 1, 10, 15, эпохи.

После 1 эпохи сеть сгенериловала повторяющуюся последовательность состоящую из 4 элементов (см. рис. 1).

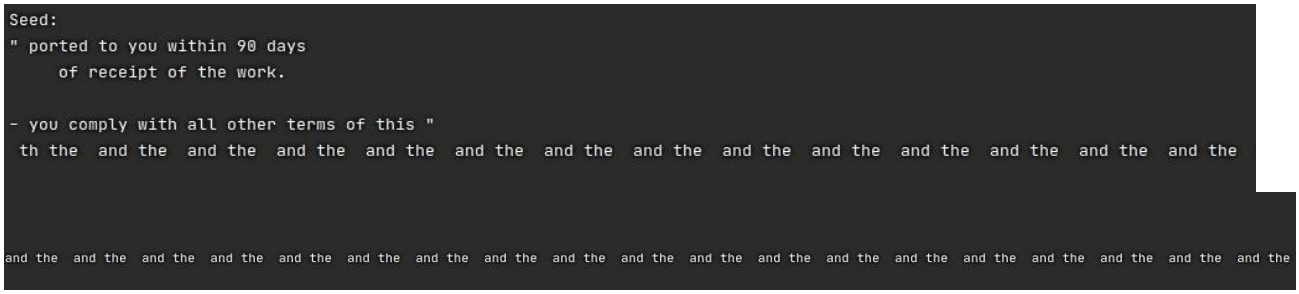

Рисунок 1 – Результат после 1 эпохи

После 10 эпохи сеть сгенериловала повторяющуюся последовательность, но уже большей длины (см. рис. 2).

```
Seed:
" lar--here, bill! catch hold of this
rope--will the roof bear?--mind that loose slate--oh, it's comin "
 to the toee '
'i shan t then to tee toene ' said the macter. ''that s the tar in whe marter ' she gade wo tee toene to tee thet she was to tee thet  she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
she was to tee toeee th the toeee
```

Рисунок 2 – Результат после 10 эпохи

После 15 эпохи сеть сгенериловала текст, в нем можно рассмотреть настоящие слова и много непонятных (см. рис. 3).

```
" linked
to the project gutenberg-tm license for all works posted with the
permission of the copyright "
 work of the project gutenberg-tm works

1.e.2.  1ye to dnl the pooject gutenberg-tm works
 1.e. . detert a.dete to the project gutenberg-

    *ee to pooaict gutenberg-tm electronic works an cneotidn wfoh the project gutenberg-tm works
 1.you are a soodl tf the sook of the sooe       ie an oot io the toiee
    and the whit hnse the soie       io
```

Рисунок 3 – Результат после 15 эпохи

После 20 эпохи сеть уже начала генерировать разборчивый текст, с адекватными словами. Обучать сеть на большем количестве эпох не предоставляется возможном, т.к. сеть с 20 эпохами обучалась ~ 4 часа. Но из вышесказанного, после ~100 эпох, сеть начнет генерировать текст без повторений, в котором можно разобрать слова, хотя также будут встречаться непонятные.

```
Epoch 00020: loss improved from 2.07751 to 2.05627, saving model to weights-impro
./weights-improvement-20-2.0563.hdf5
FULL MODEL
Seed:
" med alice.

'and ever since that,' the hatter went on in a mournful tone, 'he won't
do a thing i ask "
l tas i ral in wiu '
'i mont that you dane to tee thit ' said the monk turtle.
'io a ling taal to heve ' said the monk turtle.
'io a ling taal to teal ' said the monk turtle.
'io a ling taal to toued thet,' said the monk turtle.
'io a ling taal to heve ' said the monk turtle.
'io a ling taal to teal ' said the monk turtle.
'io a ling taal to toued thet,' said the monk turtle.
'io a ling taal to heve ' said the monk turtle.
'io a ling taal to teal ' said the monk turtle.
'io a ling taal to toued thet,' said the monk turtle.
'io a ling taal to heve ' said the monk turtle.
'io a ling taal to teal ' said the monk turtle.
'io a ling taal to toued thet,' said the monk turtle.
'io a ling taal to heve ' said the monk turtle.
'io a ling taal to teal ' said the monk turtle.
'io a ling taal to toued thet,' said the monk turtle.
'io a ling taal to heve ' said the monk turtle.
'io a ling taal to teal ' said the monk turtle.
'io a ling taal to toued thet,' said the monk turtle.
'
```

Рисунок 4 – Результат после 20 эпохи

Выводы.

Была построена и обучена нейронная сеть для генерации текстов на основе «Алисы в стране чудес». Был написан CallBack, с помощью которого отслеживался прогресс нейронной сети. В результате обучения сеть научилась генерировать неосмысленные тексты, в которых встречаются слова.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import sys
from os import listdir, path

import numpy
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, callbacks
from keras.utils import np_utils



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))
itc = 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])
```

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

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


def custom_print(custom_model):
    start = numpy.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")

    print("\"", ''.join([itc[value] for value in pattern]), "\"")

    for i in range(1000):
        x = numpy.reshape(pattern, (1, len(pattern), 1))
        x = x / float(n_vocab)
        prediction = custom_model.predict(x, verbose=0)
        index = numpy.argmax(prediction)
        result = itc[index]
        print(result, end='')
        pattern.append(index)
        pattern = pattern[1:len(pattern)]


n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)

# reshape X to be [samples, time steps, features]
X = numpy.reshape(dataX, (n_patterns, seq_length, 1))
# normalize
X = X / float(n_vocab)
# one hot encode the output variable
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, callback_with_custom_print([1, 10, 15])]

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

## generate ##

# choose filename

filename = 'need to choose from directory with min loss'

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