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 Стане чудес»

Студент гр. 7383 _____ Власов Р.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. 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

Ход работы.

Для выполнения задания была разработана и использована программа, код которой приведен в приложении А.

Была построена простая рекурентная сеть, показанная на рис. 1.

Для контроля процесса обучения был написан RuntimeTestCallBack, раз в заданное количество эпох генерирующий текст из 100 символов при помощи обучаемой модели. Было принято решение использовать одно и тоже ядро для каждой генерации текста, так как это позволит более наглядно проследить процесс обучения сети.

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

Рисунок 1 — Архитектура сети

В модель также был добавлен ModelCheckpoint CallBack для сохранения модели с наименьшей ошибкой и EarlyStopping CallBack для завершения обучения после того, как параметр ошибки модели перестанет уменьшаться в течении двух шагов подряд.

Обучение было запущено на 100 эпохах. После 54 эпохи обучение остановилось, так как ошибка перестала уменьшаться.

Ниже представлены тексты, сгенерированные моделью в процессе обучения:

Ядро	the collection of project gutenberg-tm electronic works. nearly all the individual works in the col
Эпоха 0	,,
Эпоха 3	e toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe toe to
Эпоха 6	e the tore th the tore the toete th the tore the tore the tore the toete th the tore the toete th th
Эпоха 9	e the toeee the toree the toree the toree the toree the toree the toree the toree the toree the tore
Эпоха 12	e the had and the toee to the the tooee th the toete and the toeee to the toet she woete to the to
Эпоха 15	e the soeee and the woene sas io the war oo the care and the toene tai oo the tart on the tarte sa
Эпоха 18	e the harter, and the whit sar io the woile sa the toiee and the woine tai ioon the thid th the toe
Эпоха 21	e the har hn the was and the war of the care an a lortle oo the care io a lortee oo the sar. 'what
Эпоха 24	e the dar an an fnol of the toeee and whe wai sortle soine oatee the white sab inot oh the toene a
Эпоха 27	e the dare and the sarter of the care on the woide and the whit on the tai oottle ala doond and the

Эпоха 30	e the haree aarer aere an the could, and the whit had betn thet it was a lottle soile, and the thit
Эпоха 33	e the douro, and the whin hn she had botn thtt ali hor oo the was. 'io tos di ant tane, said the m
Эпоха 36	e the harte here an the woide. 'they were to me ds toe woon to see io a little oo teee!' shought al
Эпоха 39	t the coure, whe said to the toree tha white oabdit, and the tai tottleg at the could her in the war
Эпоха 42	e the coure, and she white sabbit sas an the coul of the harter, and she tait oo sas the wirte sail
Эпоха 45	e the care an iere to the thate sf her ou tere an her fane toe kar an inr ain her woice andngr tha w
Эпоха 48	t the harter, and the white rabbit seat on was toen in the was afdin, and she thite rabbit sead tn t
Эпоха 51	t the houhe an ie so cerd aidnn, and the whil on bater at the wirle the tab aton tith the gar and th

Можно наблюдать, что на первых 15 эпохах сеть использует очень ограниченный набор символов, расположенных в одинаковом порядке. К 30 эпохе длина последовательности увеличивается, начинают вырисоввываться слова. К концу обучения последовательность становится похожей на неосмысленный набор слов с большим количеством опечаток.

Ниже представлен текст из 500 символов, который был сгенерирован при помощи обученной модели:

Ядро:

she had accidentally upset the week before.

'oh, i beg your pardon!' she exclaimed in a tone of gre

Сгенерированный текст:

at sertir.

'no iour bade to the thing tf doan ' said the daterpillar.

'iele io,' said the kanter in a tore of the konere thin at shs would the wab sooe of the care an it aounh,
'whal i tas ao a lirg thang i to wet, saed the manch hare.

'le iourd thet arr mace,' said the caterpillar.

'ieme wot toink ' said the daterpillar.

'ieme wot thin io soued to a sabd,' lhe yhat io sheme baal an anl, you know, she manch hare said to aeice,

'no toere to beyit io, said the manch hare.

'le pouh the

Сгенерированный текст представляет собой диалог. Можно разобрать некоторые слова, но реплики остаются неосмысленными. В тексте нет повторений. Интересно, что каждая строка текста оформлена в едином стиле, а повторение некоторых сочетаний слов имеет некоторый смысл.

Выводы.

В ходе выполнения лабораторной работы была разработана рекуррентная сеть для генерации текстов, она была обучена на основе сказки «Алиса в Стране чудев». Был написан CallBack, с помощью которого осуществлось отслеживание процесса обучения сети. В результате обучения сеть научилась генерировать неосмысленные тексты, в которых встречаются слова.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import os
import requests
import numpy as np
from keras import Sequential
from keras.callbacks import ModelCheckpoint, EarlyStopping,
Callback
from keras.layers import LSTM, Dropout, Dense
from keras.utils import np_utils

BOOK_URL = "https://www.gutenberg.org/cache/epub/11/pg11.txt"
BOOK_PATH = "./alice's_adventures_in_wonderland.txt"
MODEL_PATH = "./bestModel.hdf5"

EPOCHS = 100
BATCH_SIZE = 256

char_to_int = None
int_to_char = None
n_vocab = None


class RuntimeTestCallback(Callback):
    def __init__(self, pattern, frequency=1):
        super(RuntimeTestCallback, self).__init__()
        self.seed = pattern
        self.frequency = frequency

    def on_epoch_begin(self, epoch, logs={}):
        if (epoch + 1) % self.frequency == 0:
            res = generateText(self.seed, self.model, 100)
            with open("generated_on_{}.txt".format(epoch), "w")
as f:
                f.write(res)


def loadData():
    global char_to_int
    global int_to_char
    global n_vocab
    if not os.path.exists(BOOK_PATH):
        f = requests.get(BOOK_URL)
        open(BOOK_PATH, 'wb').write(f.content)
    raw_text = open(BOOK_PATH).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)
    dataX = []
    dataY = []
    seq_length = 100
    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)
    X = np.reshape(dataX, (n_patterns, seq_length, 1))
    X = X / float(n_vocab)
    Y = np_utils.to_categorical(dataY)
    return X, Y, dataX, dataY


def buildModel(shapeIn, shapeOut):
    model = Sequential()
    model.add(LSTM(256, input_shape=(shapeIn[1], shapeIn[2])))
    model.add(Dropout(0.2))
    model.add(Dense(shapeOut[1], activation='softmax'))
    model.compile(loss='categorical_crossentropy',
optimizer='adam')
    return model


def trainModel():
    start = np.random.randint(0, len(dataX) - 1)
    seed = dataX[start]
    print("Seed: \"", ''.join([int_to_char[value] for value in
seed]), "\"")
    checkpoint = ModelCheckpoint(MODEL_PATH, monitor='loss',
verbose=1, save_best_only=True, mode='min')
    earlyStopping = EarlyStopping(monitor='loss', verbose=1,
mode='min', patience=2)
    customCallback = RuntimeTestCallback(seed, 1)
    callbacks_list = [checkpoint, earlyStopping, customCallback]
    model = buildModel(X.shape, Y.shape)
    model.fit(X, Y, epochs=EPOCHS, batch_size=BATCH_SIZE,
callbacks=callbacks_list)


def generateText(pattern, model, length=100):
    pattern = pattern.copy()
    res = ""
    for i in range(length):
        x = np.reshape(pattern, (1, len(pattern), 1))
        x = x / float(n_vocab)
        prediction = model.predict(x, verbose=0)
        index = np.argmax(prediction)
        result = int_to_char[index]
```

```
        res += result
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
    return res


X, Y, dataX, dataY = loadData()

trainModel()

start = np.random.randint(0, len(dataX) - 1)
seed = dataX[start]
print("Seed: \"", ''.join([int_to_char[value] for value in
seed]), "\"")
model = buildModel(X.shape, Y.shape)
model.load_weights(MODEL_PATH)
text = generateText(seed, model, 500)
with open("generatedText.txt", "w") as f:
    f.write(text)
```