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

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Искусственные нейронные сети»

Тема: Генерация текста на основе “Алисы в стране чудес”

Студент гр. 7382 Гиззатов А.С.

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

Ход работы.

1.Построили рекурентую сеть, код которой представлен ниже.

```
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')
```

2.Написали callback для генерации текста после заданных эпох:

```
def print_seq(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)]


class MyCallBack(callbacks.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:
            print_seq(self.model)
```
3.Обучили модель и вызвали callback после 3,26,32,35 эпох

3 эроха:

Seed:

" very poor speaker,' said the king.

here one of the guinea-pigs cheered, and was immediately suppre "

th the toet to

26 эроха:

Seed:

" edgehogs,

the mallets live flamingoes, and the soldiers had to double themselves

up and to stand on "

the courd,

'the was a little oittle toan a sirt,' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

'lete toe cane 'it a lert wir,' said the manch hare.

'lete toe cane 'it a lort of toon?' said the manch hare.

32 эпоха:

Seed:

" her going, though she

looked back once or twice, half hoping that they would call after her:

the las "

her toite wosl the war oo the whit oa thit an inr tas of toene on the toeee of the caree.

'lo aanre fin to toe thin,' said the king.

anice was no the sirt hird aerur the thet seee the careenilgly was she whr hn oo toenk oo the toee as she sas of the gareen, and the whst on and tosed of the girter was ano the gias of the grrpt oa the har hn whi hade heteerel ta the wood she was a little woitel woued belu and thit sosel on the toede of the tabdit was on the whst on the hareen, and the whst on and wostd th the wordd at sel ctuld oo the dourt, and whs whil t elen thted an anl oi the thid oo the court, and when hl soen a sien ail the tiie of the harter,

'io whu a dathe i lan oo the doecess? the sooe to tee the wood tou doo to tie toe thatl oo the than had hoon the white rabbit was anong ter the hareen, and tee toond the was soinking at the could,

'the dir't tiink io the siatt woule ' said the ming,

35 эпоха:

Seed:

" ppose they are the jurors.' she said

this last word two or three times over to herself, being rather "

to tee the huophon. and the woile sas agonn a looe tr tee sooe as at anlle to she toike and whn world he donn the woide an anl oo the toid wo the toene on the soiee an in was soeneing ano the tirt wfs sne tan aoo the wan oo the toeee tf the sooee of the care tith the sas of the pibeter and ths woine that soeee the had

hene to the toide of the sabdi to tee the har her ano oo the samd tf the soeee an it was soe time of taek th the toede an she courd,
'the wast an anl an anl ' said the daterpillar.

'ieve tou toint,' said the manch hare.

'iere aon meae!' she manch hare said to the jury,
'ne you sein to toe thin to tey,' said the daterpillar.

'ieve tou toint,' said the manch hare.

'iere aon meae!' she manch hare said to the jury,
'ne you sein to toe thin to tey,' said the daterpillar.

'ieve tou toint,' said the manch hare.

'iere aon meae!' she manch hare said to the jury,
'ne you sein to toe thin to tey,' said the daterpillar.

'ieve tou toint,' said the manch hare.

'iere aon meae!' she manch hare said to the jury,
'ne you sein to toe thin to tey,' said the daterpillar.

'ieve tou toint,' said the manch hare.

В итоге сеть научилась писать некоторые предлоги типа and,was,an и слова said,she. Но ничего осмысленного написать она не может.

Выводы.

Была обучена рекуррентная сеть для генерации текстов на основе «Алисы в стране чудес». Был написан callback, с помощью которого отслеживался прогресс сети

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))
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])


def print_seq(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)]


class MyCallBack(callbacks.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:
            print_seq(self.model)


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))

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([0, 2, 3, 4, 5, 6, 7, 10, 11, 12,
15,20,21,24,25,30,31,32,34,35])]
model.fit(X,
          y,
          epochs=35,
          batch_size=156,
          callbacks=callbacks_list)

folder = '.'
filename = ''
min = 100000
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')
print('FULL MODEL')
print_seq(model)
print("\nDone.")
```