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. 8382 Рочева А.К.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2021

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

Выполнение работы.

1. Реализовать модель ИНС, которая будет генерировать текст

Архитектура модели была взята целиком из методических указаний, т.к. изменение параметров часто только ухудшало результаты.

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)

Был создан класс, наследующий класс keras.callbacks.Callback, в котором был переопределен метод on_epoch_end, который в конце каждой эпохи, вызывает функцию, которая генерирует входные данные и подает их на вход недообученной модели. Результат этой функции записывается в текстовый файл. Код класса и функции имеется в приложении.

3. Отследить процесс обучения при помощи TensorFlowCallBack, в отчете привести результаты и их анализ

Содержимое generated.txt (повторы представлены в виде троеточия):

Epoch 0	" aa th the aa a
Epoch 1	"re toe t"
Epoch 5	" to the toie 'i wo toe toet to tee toet ' said the monge 'nn t a lott oo tee toet to tee to the toete '
Epoch 10	to the toiet of the woide ' 'io tou dre t teoe the soiee said alice. 'io whu would ' said the daterpillar. 'ie course tou doon,' said the monk turtle. 'io whu would ' said the daterpillar. …
Epoch 15	" was soe tinel th the hookeo to tee the had soted her and the white rabbit worl a little toice an in wheh so the har hn the woole and the toied of the gourhon she was anl toene to tee the hooph of the tooe of the houpho of the tooe, and the white wab ittt at the could an she was aol soine in the wioe whth al cnoe to the had so tee the was ao tne oane an the mooke, and she was aol so toek the had so tee the white rabbit woth the would her to toek the had totel to toen it was toe tiit soued ' 'whl hodst d teil to tol ' said the muck turtle in a vore of geeage

	'the hoest d aet ro ae a later wail to ' said the mock turtle. ''whet io the montle to tea ' 'i movt then to toe
Epoch 20	could aetor at the was soi tfite to her an on the was of the war affind to the was ano ano the wisee soiee to the was ano ano the wisle sat of the was affind to the was oo tie tabeit to be a fott of the tas of the table, and the whste tae it wuund in a lowele of the sabeit to het an on oo hir ananpe toene the had boennte the was soi tiite tas soe tint hat so bet hn an sor of the batere lf the was ano soe tite she was anl soene to the tas aflind to the tas of the war affind to the was ano ano the wisle sat if a croatuittions to cear taedi th the tabte. and the whst hnt lose to be a foot white was sooning ano the wise whe had boenned of the tabli, and the whste tae it wuund in a lowele of the sabeit to het an on oo hir ananpe toene the had boennte the was soi tiite tas soe tint hat so bet hn an sor of the batere lf the was ano soe tite she was anl soene to the tas aflind to the tas of the war affind to the was ano ano the wisle sat if a croatuittions to cear taedi th the tabte. and

Как видно, с каждой новой эпохой выход становится менее похожим на просто набор букв и начинают прослеживаться очертания связного текста, однако многие символы не объединились в существующие слова. Для ознакомления с этой темой вполне достойный результат. Если же использовать другие способы представления исходных данных и использовать другие параметры сети, то можно добиться лучшего результата.

Вывод.

В ходе выполнения данной работы произведено ознакомление метода генерации текста с помощью рекуррентных нейронных сетей и работа с Callback в Keras.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import sys
from os.path import exists

import keras.callbacks
import numpy as np
from keras.callbacks import ModelCheckpointfrom
keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.models import Sequentialfrom
keras.utils import np_utils


class MyCallback(keras.callbacks.Callback):
    def __init__(self):
        super(MyCallback, self).__init__()

    def on_epoch_end(self, epoch, logs=None):if
        epoch == 1 or epoch % 5 == 0:
            generate_symbols(self.model, epoch)


def generate_symbols(model, epoch):g =
    open('gen_text.txt', 'a')
    gen_symbols = []
    start = np.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    g.write("On epoch " + str(epoch) + " seed is:\n" + "\"" +
''.join([int_to_char[value] for value in pattern]) + "\"\n")
    for i in range(1000):
        X = np.reshape(pattern, (1, len(pattern), 1))X =
        X / float(n_vocab)
        predict = model.predict(X, verbose=0)idx
        = np.argmax(predict)
        result = int_to_char[idx]
        gen_symbols.append(result)
        pattern.append(idx)
        pattern = pattern[1:len(pattern)]
    g.write("generated text is:\n"+"\""+''.join(gen_symbols)+"\"\n")
    g.close()


if__name__== '__main__': 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))

    n_chars = len(raw_text)
    n_vocab = len(chars)

    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)

    X = np.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'))
    filename = 'best-model-weights.hdf5'
    if not exists(filename):
        model.compile(loss='categorical_crossentropy',
optimizer='adam')

        checkpoint = ModelCheckpoint(filename, monitor='loss',
verbose=1, save_best_only=True, mode='min')
        callbacks_list = [checkpoint, MyCallback()]

        model.fit(X, Y, epochs=20, batch_size=128,
callbacks=callbacks_list)
    else:
        model.load_weights(filename)

        model.compile(loss='categorical_crossentropy',
optimizer='adam')

        int_to_char = dict((i, c) for i, c in enumerate(chars))start

        = np.random.randint(0, len(dataX) - 1)
        pattern = dataX[start]
        print('\"', ''.join([int_to_char[value] for value in
pattern]), '\"')

        for i in range(1000):
```

```
    X = np.reshape(pattern, (1, len(pattern), 1))X =
    X / float(n_vocab)
    predict = model.predict(X, verbose=0)idx
    = np.argmax(predict)
    result = int_to_char[idx]
    seq_in = [int_to_char[value] for value in pattern]
    sys.stdout.write(result)
    pattern.append(idx)
    pattern = pattern[1:len(pattern)]
print('\nDone')
```