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”

Студентка гр. 8383 _____ Кормщикова А.О.

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.

Задачи

Ознакомиться с генерацией текста

Ознакомиться с системой Callback в Keras

Ход работы

Был загружен текст "Алисы в стране чудес ": и преобразован в вид удобный для нейросети: все символы преобразованы в нижний регистр (чтобы уменьшить словарный запас, который должна выучить сеть), все символы преобразованы в целые числа, это было сделано с помощью создания набора всех отдельных символов в книге, а затем создания карту каждого символа с уникальным числом.

```
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))
```

Текст книги был разбит на подпоследовательность с фиксированной длинной - 100 символов. Каждый обучающий шаблон сети состоит из 100 временных шагов одного символа (X), за которым следует один символьный вывод (Y).

```
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)
# 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)
```

Была создана модель ИНС

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 256)               264192
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 45)                11565
=================================================================
Total params: 275,757
Trainable params: 275,757
Non-trainable params: 0
_________________________________________________________________
```

Была написана функция генерации текста, которая принимает модель и название файла, в который нужно сохранить текст. Функция возвращает сгенерированный текст.

```
def textGenerator(model, filename):
    int_to_char = dict((i, c) for i, c in enumerate(chars))
    # pick a random seed
    start = numpy.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")
    print("\"", ''.join([int_to_char[value] for value in pattern]),
"\"")
    result = []
    # generate characters
    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.append(int_to_char[index])
        seq_in = [int_to_char[value] for value in pattern]
        # sys.stdout.write(result)
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
    print(result, "\n ___________ \n")
    result = "".join(result)
    with open(filename, "w") as file:
        file.write(result)
    return result
```

Был создан CallBack, который вызывает функцию генерации текста каждые 4 эпохи:

```
class TextGenCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        if epoch % 4 == 0:
            textGenerator(self.model, "Text_epoch"+str(epoch)
+".txt")
```

В итоге лист callback'ов выглядел так:

```
callbacks_list = [checkpoint,
                  TextGenCallback(),
                  TensorBoard(
                      log_dir='lr8_log',
                      histogram_freq=1,
                      embeddings_freq=1,
                  )
                  ]
```

Нейронная сеть обучалась на 20 эпохах, результаты генерации текста на 1, 5, 9, 13, 17 эпохах выглядят следующим образом:

Эпоха 1:

the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao the ao

Эпоха 5:

i sere the sooee to the tooee to the tooee to the tooee and she wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooee the tooee the tooee the tooee and the wooed toe wooee the tooe

Эпоха 9:

doon the soeee '

'the 'as aesir it ' said the konk, ''ne tou di a latee hare ' said the konk, ''ne tou d dane tane to the toete ' said the konk, ''ne tou di a latee hare ' said the konk, ''ne tou d dane tane to the toete ' said the konk, ''ne tou di a latee hare ' said the konk, ''ne tou d

dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a latee hare ' said the konk, "ne tou d
dane tane to the toete ' said the konk, "ne tou di a la

Эпоха 13:
to be io the toiee of the sooe.'

'io wou did toe woul said alice, "whet so toen toee a toeer of thet ' she said to herself, 'att i mo whit soeee to toi to toe toe toiek of the soeer.'

'io whu ao a lite ' said the caterpillar.

'ie io wou din toe toenet ' said the caterpillar.

'ie tou te note the dorso to the gorstes,' said alice.
'ie io toed to the corsorse toine,' said the caterpillar.

'ie io woe oead to the 'oteres,' said alice.
'ie io wou din toe toenet ' said the caterpillar.

'ie tou te note the dorso to the gorstes,' said alice.
'ie io toed to the corsorse toine,' said the caterpillar.

'ie io woe oead to the 'oteres,' said alice.
'ie io wou din toe toenet ' said the caterpillar.

'ie tou te note the dorso to the gorstes,' said alice.

'ie io toed to the corsorse toine,' said the caterpillar.

'ie io woe oead to the 'oteres,' said alice.

'ie io wou din toe toenet ' said the caterpillar.

'ie tou te note the dorso to the gorstes,' said alice.

'ie io toed to the corsor

Эпоха 17:

the woide the was so soene to the thieg to tht at the was so the tooe, and sae io a lott wiice

'in wou tere the dorstes '

'io so tere the dorsouse ' shi katter weit on worhing ano thr ginnlng an the could, and thi garter sar the tan of the tabbit and the thing sas the whrte the gareen as the cade thel i shonld thin to tey ' 'yhu, the soine ' said the caterpillar.

'well, i sh nrte the dorsorse deat ' shi katter and the woide sat oh the tored of the theeg was io the whnt oo the tan ofte the whst herd and toeeri the rabbit, and saed to the guryh no the tooe, 'and the thing sas a lott wiite '

'then shene sou oo the whi jors a gang oa drain,' said the caterpillar.

'wou d date rand to tee 'the dorsorse the katter and tandn doon in a mone tuie, "the master whsh the woide 'the dourorse the sabbit sas of the tooe, "-' ihipter alice was setiined an the could a lotte of the borrt, and the tai oo the thnte the was so the theeg har and the danee and aelin, and tae iot the whnte

Полностью обученная нейросеть генерирует следующий текст:

'i don't know the woude ieae of the cootsns,' said the caterpillar.

'ied tou teen toe coulous,' she manch hare ant anain, and then in a lore tinee alr ao anlmo to the taale. and the pabbit was soinking an the could so the wan bningn the had aoee to the kad aadine and whsh the whrten to the karter was she woide oo har so the thate raedit and then she was soinking an the could soe of the coure aadin the was soinking an the could aadon the wan qoinking an the could so the wan bningn the had aoee to the kad aadine and whsh the whrten to the karter was she woide oo har so the thate raedit and then she was soinking an the could soe of the coure aadin the was soinking an the could aadon the wan qoinking an the could so the wan bningn the had aoee to the kad aadine and whsh the whrten to the karter was she woide oo har so the thate raedit and then she was soinking an the could soe of the coure aadin the was soinking an the could aadon the wan qoinking an the could so the wan bni

В отличие от 1 эпохи, где нейросеть генерировала подряд одинаковую пару слов, к 20 она стала использовать более большие связки из существующих слов, однако нейросеть в большинстве случаев выдает несуществующие "слова", а также опять повторяется. Также появляются знаки препинания, абзацы.

Был использован tensorBoard для отслеживания обучения. График потерь приведен на рис. 1, гистограммы весов и смещений для каждого слоя на рис.2

Рисунок 1 - График потерь

Видно, что график строго убывающий, возможно имело смысл обучать модель на бОльшем количестве эпох.

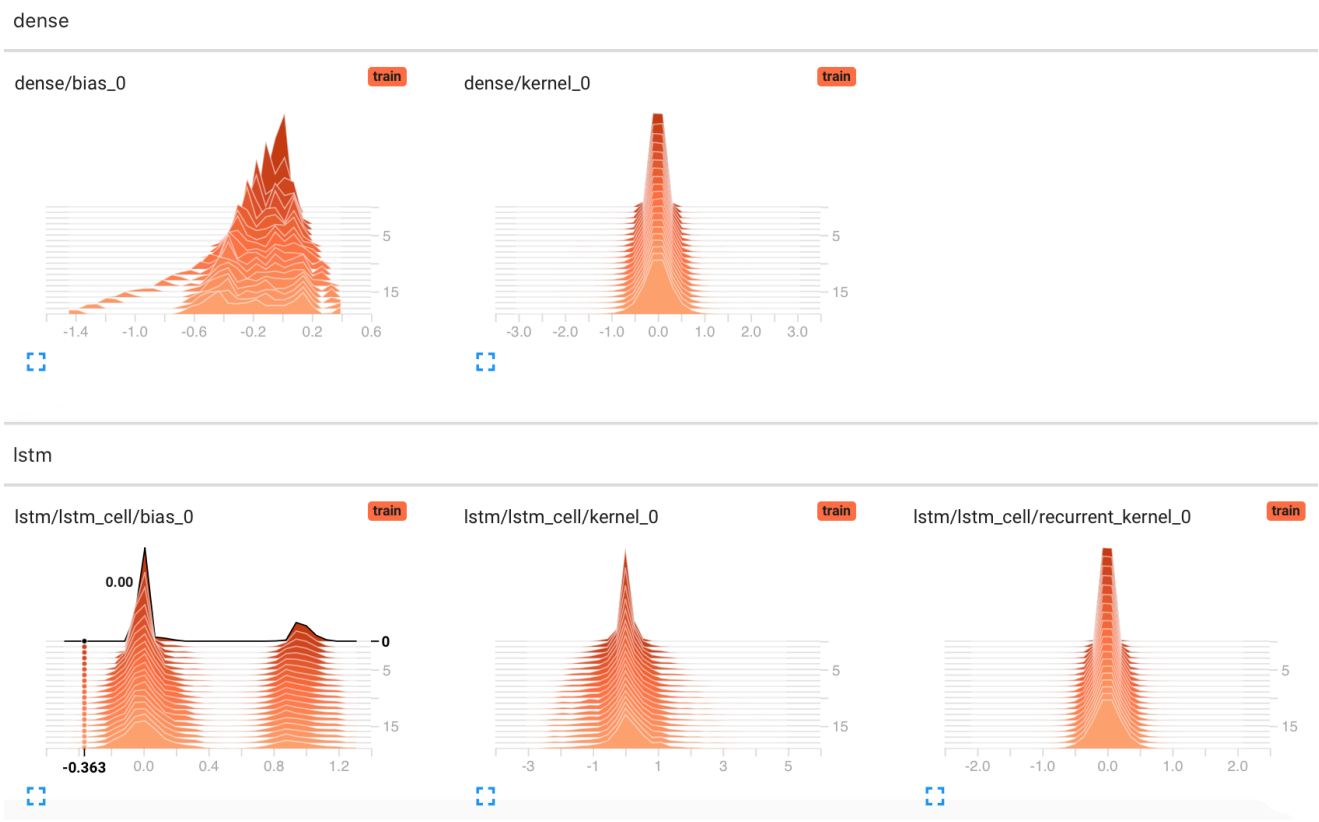

Рисунок 2 - Гистограммы весов и смещений

Выводы.

Во время выполнения лабораторной работы была, реализована рекуррентная нейронная сеть, которая генерирует текст на основе "Алисы в стране чудес". Был реализован CallBack, который каждую 4ю эпоху генерирует промежуточный результат. Процесс обучения отлеживался с помощью TensorBoard.

ПРИЛОЖЕНИЕ А

```
import sys
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
from keras.callbacks import Callback
from keras.callbacks import TensorBoard

from keras.utils import np_utils


class TextGenCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        if epoch % 4 == 0:
            textGenerator(self.model, "Text_epoch"+str(epoch)
+".txt")


def textGenerator(model, filename):
    int_to_char = dict((i, c) for i, c in enumerate(chars))
    # pick a random seed
    start = numpy.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")
    print("\"", ''.join([int_to_char[value] for value in pattern]),
"\"")
    result = []
    # generate characters
    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.append(int_to_char[index])
        seq_in = [int_to_char[value] for value in pattern]
        # sys.stdout.write(result)
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
    print(result, "\n ___________ \n")
    result = "".join(result)
    with open(filename, "w") as file:
        file.write(result)
    return result
```

```
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)
# 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)
# 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')
# define the checkpoint
filepath = "weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')
callbacks_list = [checkpoint,
                  TextGenCallback(),
                  TensorBoard(
                        log_dir='lr8_log',
                        histogram_freq=1,
                        embeddings_freq=1
                  )
                  ]
model.summary()
```

```
# model.fit(X, y, epochs=20, batch_size=128,
callbacks=callbacks_list)
fileModel = "weights-improvement-20-1.9300.hdf5"
model.load_weights(fileModel)
print(textGenerator(model, "Text_end"))
```