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

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Искусственные нейронные сети»

Тема: «Генерация текста на основе «Алисы в стране чудес»»

Студент гр. 7382 _____ Находько А.Ю.

Преподаватель _____ Жукова Н.В.

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.

Задачи работы.

•Ознакомиться с генерацией текста

•Ознакомиться с системой Callback в Keras

Требования 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

Основные теоретические положения.

Искусственные нейронные сети - совокупность моделей, которые представляют собой сеть элементов - искусственных нейронов, связанных между собой синаптическими соединениями.

Нейронные сети используются как среда, в которой осуществляется адаптивная настройка параметров дискриминантных функций. Настройка происходит при последовательном предъявлении обучающих выборок образов из разных классов. Обучение - такой выбор параметров нейронной сети, при котором сеть лучше всего справляется с поставленной проблемой.

Нейрон — элемент, преобразующий входной сигнал по функции.

Сумматор — элемент, осуществляющий суммирование сигналов поступающих на его вход.

Синапс — элемент, осуществляющий линейную передачу сигнала.

Экспериментальные результаты.

В ходе выполнения лабораторной работы, была построена модель ИНС, которая генерирует текст, код построенной модели представлен в Приложении А.

Также был реализован собственный CallBack, который показывает то как генерируется текст во время обучения, код представлен ниже:

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

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

def gen_sequence(model):
# 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]), "\"")
# 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 = 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)]
```

Также был отслежен процесс обучения при помощи TensorFlowCallBack, в отчете ниже приведены результаты и их анализ.

Номер эпохи	Выведенный результат
1	" oden spades, then a row of lodging houses, and behind them a railway station.) however, she soon mad " and the an the and and the an the and and the an the and and the an the and and the an the and and the an the and and ...
3	" next! if they had any sense, they'd take the roof off.' after a minute or two, they began moving ab " an ...
12	"ying we beg your acceptance of this elegant thimble“; and, when it had finished this short speech, " and the white rabbit were to aeiin an once

	the mote tu tee sai so tae she mart was oo the tooe, and the white rabbit were to ani the past oa tee so tae the tas of the pooe of the courd, and the white rabbit were to ani the past oa tee so tae the tas of the pooe of the courd, and the white rabbit were to ani the past ...
20	" the knave, 'i didn't write it, and they can“t prove i did: there's no name signed at the end.'if " io ' said the manch hare. „ie tou t tou a taid to tey,' she maic thit have a lant lirtle toiee so the tent oo toe tiat shie the was aol the tas aoi the cadl she was aolin tote the thse oh the sas ho was aoong the had so the table bnd sae to theng tas toen in the was aol the was aol the tas aoi the cadl ...

Существует две основные дихотомии при использовании RNN для текста: символьная и словесная, мы рассматриваем последовательности символов. С каждой новой эпохой текст становится более осмысленным и связным, но всё же многие символы не объединились в существующие слова, поэтому можно прийти к выводу, что необходимо большее количество эпох, чтобы сгенерировать полностью человекопонятный текст, но это потребовало бы большее количество времени.

Выводы.

В ходе выполнения лабораторной работы была реализована модель ИНС, которая генерирует текст на основе набора данных Приключения Алисы в

Стране Чудес Льюиса Кэрролла.. Были изучены зависимости между символами и условные вероятности символов в последовательностях, чтобы мы могли, в свою очередь, генерировать совершенно новые и оригинальные последовательности символов. Был написан собственный CallBack, который показывает то, как генерируется текст во время обучения. Также ознакомился с системой Callback в Keras.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
import sys
import numpy
import tensorflow
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])
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)

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

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

def gen_sequence(model):
# 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]), "\"")
# 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 = 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)]

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, gen_callback([1, 3, 12, 20])]

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