

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”

Студент гр.7382

Токарев А.П.

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. Ознакомиться с генерацией текста.
2. Ознакомиться с системой Callback в Keras.

Требования к выполнению задания.

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 генерировать и выводить текст у необученной модели).
3. Отследить процесс обучения при помощи TensorFlowCallBack, в отчете привести результаты и их анализ.

Основные теоретические положения.

Многие из классических текстов больше не защищены авторским правом. Это означает, что вы можете скачать весь текст этих книг бесплатно и использовать их в экспериментах, например, при создании генеративных моделей. Возможно, лучшее место для получения доступа к бесплатным книгам, которые больше не защищены авторским правом, это Проект Гутенберг.

В данной лабораторной работе мы будем использовать в качестве набора данных Приключения Алисы в Стране Чудес Льюиса Кэрролла. Мы собираемся изучить зависимости между символами и условные вероятности символов в последовательностях, чтобы мы могли, в свою очередь, генерировать совершенно новые и оригинальные последовательности символов.

Ход работы.

1. Была построена и обучена модель нейронной сети, которая будет генерировать текст. Код предоставлен в приложении А.
2. Был написан собственный Callback, который показывает то как генерируется текст во время обучения, то есть раз в какое-то количество эпох генерировать и выводить текст у необученной модели.

```
class callback_with_custom_print(callbacks.Callback):  
    def __init__(self, epochs):  
        super(callback_with_custom_print, self).__init__()  
        self.epochs = epochs  
    def on_epoch_end(self, epoch, logs=None):  
        if epoch in self.epochs:  
            custom_print(self.model)
```

```
def custom_print(custom_model):  
    start = numpy.random.randint(0, len(dataX) - 1)  
    pattern = dataX[start]  
    print("Seed:")  
    print("\n", ".join([itc[value] for value in pattern]), "\n")  
    for i in range(1000):  
        x = numpy.reshape(pattern, (1, len(pattern), 1))  
        x = x / float(n_vocab)  
        prediction = custom_model.predict(x, verbose=0)  
        index = numpy.argmax(prediction)  
        result = itc[index]  
        print(result, end="")  
        pattern.append(index)  
        pattern = pattern[1:len(pattern)]
```


" hatter, with an anxious look at the queen, who was reading the list of singers.

[illegible]

После 15 эпохи сеть сгенерировала текст, в котором можно разглядеть несколько внятных слов (см. рис. 3).

and so she went on, taking first one side and then the other, and making quite a conversation "

[illegible]

При дальнейшем более длительном обучении сети, результат был бы еще лучше, предположительно на 50 эпохе уже понятные крупные фрагменты текста.

Выводы.

Была построена и обучена нейронная сеть для генерации текстов на основе «Алисы в стране чудес». Был написан CallBack, с помощью которого отслеживался прогресс нейронной сети. В результате обучения сеть научилась генерировать неосмысленные тексты, в которых встречаются слова.

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 = "/home/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))
itc = 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])

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

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

def custom_print(custom_model):
    start = numpy.random.randint(0, len(dataX) - 1)
```

```

pattern = dataX[start]
print("Seed:")

print("\n", ''.join([itc[value] for value in pattern]), "\n")

for i in range(1000):
    x = numpy.reshape(pattern, (1, len(pattern), 1))
    x = x / float(n_vocab)
    prediction = custom_model.predict(x, verbose=0)
    index = numpy.argmax(prediction)
    result = itc[index]
    print(result, end='')
    pattern.append(index)
    pattern = pattern[1:len(pattern)]

%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

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')

filepath="weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')
callbacks_list = [checkpoint, callback_with_custom_print([1, 10, 15])]

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

## generate ##

# choose filename

filename = 'need to choose from directory with min loss'

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

```



```
custom_print(model)
```