

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

Бахеров Д.В.

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.

Модель:

```
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)
```

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)]
```

3. Отследим процесс обучения и рассмотрим тексты сгенерированные после 1, 10, 15, эпохи.

После 1 эпохи сеть сгенерировала повторяющуюся последовательность элементов (см. рис. 1).

```
" t was impossible
to say which), and they went on for some while in silence.

alice was just beginnin "
to the and the and the and the and the and the and the and the and the
and the and the and the and the and the and the and the and the and the
he and the and the and the and the and the and the and the and the and
the and the and the and the and the and the and the and the and the and
nd the and the and the and the and the and the and the and the and the
```

Рисунок 1 – Результат после 1 эпохи

После 10 эпохи сеть сгенерировала повторяющуюся последовательность с большим количеством непонятных слов, но уже большей длины (см. рис. 2).

```
" rabbit's little white kid gloves while
she was talking. 'how can i have done that?' she thought. 'i "
n a lortle to toee th the tore '
'i don't tao ao a lertle to the toee ' shi gacter alited an an anpea and the woiee. ''the woued thet wou do a lane '
'i don't tao ao a lare ' said the mock turtle. ''than the wored to tee ao the mooe '
'i d vare the moren tf the tore ' shi gacter alited an an anpea and the woiee. ''then the wored to tee ao a larel torel '

'io wou then toe toee to the wou do a lane ' said the mock turtle. ''than the wored to tee an a lare to the toee '

'io wou then toe toon ' shi toedk the poren shs whet she tente thrt toeee and the woide the war soe th the woiee th the toee and the woide the war soe th the wo
ar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the tar oo the
```

Рисунок 2 – Результат после 10 эпохи

После 15 эпохи сеть сгенерировала текст, в нем можно рассмотреть настоящие слова и словосочетания и много непонятных (см. рис. 3).

```
" tone; 'seven jogged my
elbow.'

on which seven looked up and said, 'that's right, five! always lay "
the same toree and the toee th the toeer oo the toee of the coure, and the woidk tae an
lottle to tee the was oo the toee of the coure, and the woidk tae anon in the care of t
was oo the toee of the coure, and the woidk tae anon in the care of the toee and the wo
the coure, and the woidk tae anon in the care of the toee and the woidk tae anon in the
woidk tae anon in the care of the toee and the woidk tae anon in the care of the toee
```

Рисунок 3 – Результат после 15 эпохи

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 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(int_to_char[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([int_to_char[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 = int_to_char[index]
        print(result, end="")
```

```

pattern.append(index)
pattern = pattern[1:len(pattern)]

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)

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)

```