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”»

Студент гр. 7382 _____ Государкин Я.С.

Преподаватель _____ Жангиров Т.Р

Санкт-Петербург

2020

Задание

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.

- Ознакомиться с генерацией текста
- Ознакомиться с системой Callback в Keras

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

Теоретическая часть

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.

Эти эксперименты не ограничиваются текстом, вы также можете поэкспериментировать с другими данными ASCII, такими как размеченные документы в LaTeX, HTML или Markdown и другие.

Ход работы

1. Реализовать модель ИНС, которая будет генерировать текст

Воспользуемся рекуррентными нейросетями, а именно слоем GRU (упрощённый LSTM), они хорошо подходят для обработки текстов, потому что могут хранить свое состояние и принимают текущее решение с учётом предыдущих решений.

В полученной конфигурации добавился 1 слой одномерной свёртки и пулинга с L2 и дропаутом, после которого идёт рекуррентный слой GRU.

Оптимизатор: Adam, lr=0.001

batch_size = 128

loss_func: categorical_crossentropy

epochs = 10

Model:

```python
self.features = Sequential([

    Conv1D(filters=128, kernel_size=3, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    MaxPool1D(pool_size=2),
    Dropout(0.1),

    GRU(input_shape=input_shape, units=256),
    Dropout(0.1),

    Dense(units=self.patterns, activation='softmax')
])
```

Была произведена попытка сгенерировать текст после третьей эпохи. Вот результат:

Seed:

" it as you go on? it's by far the most

confusing thing i ever heard!'

'yes, i think you'd better lea "

soue the soete to the soete to the sooee so the soete to the soete th the so

Как можно видеть, много повторений и это пока что бредогенератор. Увеличим кол-во эпох до 10. Таким образом сеть будет более обученная и сможет генерировать более осмысленный текст.

Seed:

" lice, looking down with wonder at

the mouse's tail; 'but why do you call it sad?' and she kept on pu "

sie tas to the sart on the sase thing she was no the sast on the sase then she was no the tast on the tase then in the could,

'whet i cen" said the mouke, and the moek turtle so the tabti '

'nhr, io i sene thet,' said the maree hare.

'io a gon' taad the moree, and the marter ar ael fonneent to the tiet.

when in i denl tas toe tiene,' said the honk,

'ne the tait to the sast on the sase thing so the she tas io the linee.

"it shen io whe sait to the toet,' said the honk,

'nee wont hn wou teit to the toeen the hooet to the tiet th the siet.

whet she soieg sas a lire tiet, and tha sait to al anl the sase thing so the bouthen.

'weel in i denl tien iave,' said the honk,

'nee woit in wou teit to the toeer th the soeen.

'it she said thi horseen, taid the mocke.
'i whnn i cenn tay the mirele,' said the mink, 'then in a long tiet to be and the saiein of the sieen.
" ch phe tiee thi hoose so the say a lire ti the toeen,
shen whe soeen so bl she lanee,

Как можно видеть появились более сложные конструкции, можно разобрать некоторые слова.

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 написан след. callback: в началле обучения генерируется seed, на основе которого каждые 2 эпохи генерируется текст длиной в 200 символов.

Seed:
" y will do next! if they had any sense, they'd take the
roof off.' after a minute or two, they began "

Результаты. До обучения:
v(((((vv(((((dvd((((vv(d(((((vv(((((dvd((((vv(d(((((vv(((((dvd((((vv(d(((((vv(((((dvd(((
(vv(d(((((vv(((((dvd((((vv(d(((((vv(((((dvd((((vv(d(((((vv(((((dvd((((vv(d(((((vv(((((d
vd((((vv(d(((((vv(((((dvd

Результаты. 1 эпоха:

ahe aaa tahe aaa tahe ata tahe aaa tahe ata tahe aaa tahe ata tahe aaa tahe ata tahe aaa tahe ata tahe aaa tahe ata tahe aaa tahe ata tahe aaa tahe ata tahe aaa tah

Результаты. 3 эпоха:

e soot to the soote an she soot to the soote an she soot to the soote an she soot to the soote an she soot to the soote an she soot to the soote an she soot to the soote an she soot to the soote an sh

Результаты. 5 эпоха:

e tahe awo iaor tahe twhe dtn tahe twhe dtn tahe twhu tahe tthe dtn tahe twhu tahe tthe kto aaod tthu sahe tthe dtn tahe twhe dto tahe twhe dto tahe twhe dto tahe twhe dto

Результаты. 7 эпоха:

to herself and the hoopde th the goothen and the had not th the garter and the pabt th the larter and the pabte the saster and the pabt th the larter and the pabtet the had botn the tast on the had b

Результаты. 10 эпоха:

Seed:" *oise and*

confusion, as the large birds complained that they could not taste

theirs, and the small on "

the whit of the goute aer the whit oa the courd, and the gottd to the gort of the goute aer the was aoligreng to the gort and the was aoiigreng the thet the whi gott and the gort of the goute aer the whit on the while the whit on the while the whit on the while the while the whit on the while the while the while the while the whit on the gout, and the gottd the whit on the gout of the gorte to the gort of the goute aer the was aoligreng the white the goutd aer the whit oa the courd aer the was aoligr of the goore

and the was aoi the was aowt and the tas to the goo of the goor, and the gottd tae she gott and the fout of the gorte aer the was aoligreng the whit to the goot, and the gottd toe dod the was ao inc tooe the goor of the gorte aer the whit on the while the whit on the while the whit on the while the while the whit on the while the while the while the while the whit on the gout, and the gottd the whit on the gout of the gorte to the gort of the goute aer the was aoligreng the w

Как можно видеть в процессе обучения модели генерируемый текст становится всё более приобретающим черты реального текста. Становится больше правильных не бредовых слов, они становятся более разнообразными, появляются знаки препинания. Тем не менее до более менее внятного текста ещё очень далеко.

Код callback-а:

```python
# callback implementation
class CustomCallback(keras.callbacks.Callback):
    def __init__(self, dataX, model, int_to_char, interval=2, epochs=epochs):
        super(CustomCallback, self).__init__()
        self.dataX = dataX
        self.epochs = epochs
        self.model = model
        self.int_to_char = int_to_char
        self.push_interval = interval

    def on_train_begin(self, logs={}):
        self.sample = []

    def on_epoch_begin(self, epoch, logs={}):
        if (epoch + 1) % self.push_interval == 0 or epoch == 0 or (epoch + 1) == self.epochs:
            start = np.random.randint(0, len(self.dataX) - 1)
            pattern = self.dataX[start]
            genText(gen_len=200, int_to_char=self.int_to_char, model=self.model, dataX=self.dataX, pattern=pattern)
```

Выводы

В данной лабораторной работе была написана и обучена нейронная сеть для генерации текста на основе текста сказки “Алиса в стране чудес”. Нейросеть состоит из одномерного свёрточного, пулинг слоя, дропаутов а также рекуррентного слоя GRU. По мере увелиичения количества эпох при обучении с помощью собственного callback-а было продемонстрировано, что сеть генерирует всё более осмысленные слова и их последовательности, однако недостаточные для реального адекватного текста. Код в приложении А.

ПРИЛОЖЕНИЕ А

```
import tensorflow.keras as keras
import sys

from tensorflow.keras import losses
from tensorflow.keras import optimizers
from tensorflow.keras.layers import Dropout, Dense, GRU, Conv1D, MaxPool1D
from tensorflow.keras.models import Sequential

from tensorflow.keras import initializers
from tensorflow.keras import regularizers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt
import numpy as np

# constants
filename = "alice_in_the_wonderland.txt"
batch_size = 128
epochs = 10



def genText(dataX, int_to_char, model, pattern, gen_len=1000):
        # pick a random seed
        print("Seed:")
        print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")
        # generate characters
        for i in range(gen_len):
```

```
        x = np.reshape(pattern, (1, len(pattern), 1))
        x = x / float(n_vocab)
        prediction = model.predict(x, verbose=0)
        index = np.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)]
        print("\nDone.")


class TextGenerator(keras.Model):
   def __init__(self, patterns=10, input_shape=(1,1)):
      super(TextGenerator, self).__init__()

      self.weight_init = initializers.normal
      self.patterns = patterns

      self.features = Sequential([

         Conv1D(filters=128, kernel_size=3, activation='relu',
kernel_regularizer=regularizers.l2(0.001)),
         MaxPool1D(pool_size=2),
         Dropout(0.1),

               GRU(input_shape=input_shape, units=256),
         Dropout(0.1),
```

```
            Dense(units=self.patterns, activation='softmax')
        ])

    def call(self, inputs):
        x = self.features(inputs)
        return x


# callback implementation
class CustomCallback(keras.callbacks.Callback):
    def __init__(self, dataX, model, int_to_char, interval=2, epochs=epochs):
        super(CustomCallback, self).__init__()
        self.dataX = dataX
        self.epochs = epochs
        self.model = model
        self.int_to_char = int_to_char
        self.push_interval = interval
        start = np.random.randint(0, len(self.dataX) - 1)
        self.pattern = self.dataX[start]


    def on_train_begin(self, logs={}):
        self.sample = []

    def on_epoch_begin(self, epoch, logs={}):
        if (epoch + 1) % self.push_interval == 0 or epoch == 0 or (epoch + 1) == self.epochs:
            genText(gen_len=200, int_to_char=self.int_to_char, model=self.model,
dataX=self.dataX, pattern=self.pattern)
```

```
# load data
raw_text = open(filename).read()
raw_text = raw_text.lower()

# prepare data
unique_chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(unique_chars))
int_to_char = dict((i, c) for i, c in enumerate(unique_chars))

n_chars = len(raw_text)
n_vocab = len(unique_chars)

# convert chars to it's integer keys
dataX = []
dataY = []
seq_length = 100
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 && normalize
X = np.reshape(dataX, (n_patterns, seq_length, 1))
X = X / float(n_vocab)
y = to_categorical(dataY)
```

```
# init model
model = TextGenerator(input_shape=(X.shape[1], X.shape[2]), patterns=y.shape[1])
optimizer = optimizers.Adam(lr=0.001)
loss = losses.CategoricalCrossentropy()

callback = CustomCallback(dataX=dataX, int_to_char=int_to_char, model=model)
model.compile(optimizer=optimizer, loss=loss)
model.fit(X, y, epochs=epochs, batch_size=batch_size, callbacks=[callback])

start = np.random.randint(0, len(dataX) - 1)
pattern = dataX[start]
genText(dataX=dataX, model=model, int_to_char=int_to_char,
gen_len=1000,pattern=pattern)
```