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. 8383 _____ Мололкин К.А.

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

Требования

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 (TensorBoard), в отчете привести результаты и их анализ

Выполнение работы

Первым шагом была составлена модель рекуррентной нейронной сети, для предсказания символа на основе предыдущего. Архитектура представлена на рис. 1.

```
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')
```

Рисунок 1 – архитектура модели нейронной сети

Для того чтобы отследить обучение сети, после каждой эпохи вызывается собственный callback, который показывает, как генерируется текст во время обучения.

Текст после 2-ой эпохи:

" wasn't trouble enough hatching the eggs,' said the pigeon;

'but i must be on the look-out for serpen "

and the and the

Текст после 9-ой эпохи:

" rm-in-arm with the dormouse. 'fourteenth of march, i think it
was,' he said.
'fifteenth,' said the "
daterpillar.
'ie aou t tee toie th tee so the sooen ' said the daterpillar.
'ied io the harter said the daterpillar.
'i sene the soeen tae io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterpillar.
'ie ao a tat toe tai io the sooen ' said the daterp

После 20-ой эпохи:

Seed:

" etting its body tucked away, comfortably enough, under
her arm, with its legs hanging down, but gene "
at the could not head tot oeke ant oo the soher aadin the was soenk oo the soher oe the sabe to her her hna the was soene tored oo the sable oo tee it whsh the wan oo the sabei oarer and saed aot the war soen she was so toeke toine

the whrl of the gorre oot herd ant the was oo the tabbi oo tee she whsl oae iar hnr to toenk oo the sable and the west on woth the mork of the sabd to the sabdit sere.
'then io the sere thing ' said the daterpillar.

'ie iou i se toing in a ging ii the sere thing ' said the daterpillar.

'ie iou i se toind in a ging a tireone" said the daterpillar.

'ie iou i se toink in a ging ii the sere thing ' said the daterpillar.

'ie iou i se toind in a ging a tireone" said the daterpillar.

'ie iou i se toink in a ging ii the sere thing ' said the daterpillar.

'ie iou i se toind in a ging a tireone" said the daterpillar.

'ie iou i se toink in a ging ii the sere thing ' said the daterpillar.

'ie iou i se toind in a ging a tireone" said the daterpillar.

С каждой эпохой нейронная сеть начинает генерировать более вменяемый текст.

Вывод

В ходе выполнения лабораторной работы было изучено использование рекуррентных нейронных в качестве генеративной модели, построена модель, генерирующая текст на основе книги «Алиса в стране чудес», а также написан свой callback для отслеживания обучения модели, после каждой эпохи.

Приложение А

Листинг программы

```
import numpy as np
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.utils import np_utils
from keras.callbacks import Callback


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

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


def generate_sequence(model):
    start = np.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")
    print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")
    for i in range(1000):
        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]
        print(result, end='')
        pattern.append(index)
        pattern = pattern[1:len(pattern)]


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 = np.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, Callback([1, 9, 18])]

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