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»

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель.

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

Ход работы.

Была построена нейронная сеть, код которой представлен в приложении А.

После первой эпохи сеть выдала текст из повторяющихся слов «toe».

После десятой эпохи.

Seed:

she said, by way of keeping up

the conversation a little.

"tis so,' said the duchess: 'and the " 3 soeer so tee toiee if the mort of the sooe-- 'i mave to the toee tf the tore ' said the kock turtle she would seit oo the woul. "the worl of the sore tf the toie ' 'i dan toe kint ti the toate ' said the caterpillar. 'wou don't teie the mont ' said the datter. 'i mene tou the lore tfrt ti the toie. 'i mene the tore tf the tore ' said the kock turtle she woule seit oo the woul. "the worl of the sore tf the toie ' 'i dan toe kint ti the toate ' said the caterpillar. 'wou don't teie the mont ' said the datter. 'i mene tou the lore tfrt ti the toie. 'i mene the tore tf the tore ' said the kock turtle she woule seit oo the woul. "the worl of the sore tf the toie ' 'i dan toe kint ti the toate ' said the caterpillar. 'wou don't teie the mont ' said the datter. 'i mene tou the lore tfrt ti the toie. 'i mene the tore tf the tore ' said the kock turtle she woule seit oo the woul. "the worl of the sore tf the toie ' 'i dan toe kint ti the toate ' said the caterpillar.

Выводы.

В ходе выполнения лабораторной работы была построена и обучена нейронная сеть, генерирующая текст на основе «Алисы в стране чудес» Кэролла. Была написана функция, позволяющая пользователю вводить свой текст. Чем дольше обучается сеть, тем более связным становится текст.

ПРИЛОЖЕНИЕ А

```
import numpy
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import LSTM
import tensorflow.keras.callbacks
from tensorflow.keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
import sys

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)

X = numpy.reshape(dataX, (n_patterns, seq_length, 1))
X = X / float(n_vocab)
y = np_utils.to_categorical(dataY)

def genSequence(model):
    start = numpy.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 = 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)]

class GenCallback(tensorflow.keras.callbacks.Callback):
    def __init__(self, epochs):
        super(GenCallback, self).__init__()
        self.epochs = epochs
```

```
    def on_epoch_end(self, epoch, logs={}):
        if epoch in self.epochs:
            genSequence(model)

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, GenCallback([0, 5, 10, 15, 19])]

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