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. 8382 _____ Ефимова М.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.

Порядок выполнения работы.

1. Ознакомиться с генерацией текста;
2. 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, в отчете привести результаты и их анализ.

Ход работы.

Многие из классических текстов больше не защищены авторским правом. Это означает, что возможно 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 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.

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

Была создана и обучена модель искусственной нейронной сети, решающая задачу генерации текста. Модель представлена на рисунке 1.

```
# Создание модели
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'))
# Инициализация параметров обучения
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Рисунок 1 – Модель нейронной сети.

Был написан собственный обратный вызов (CallBack), который позволит отслеживать то, как генерируется текст во время обучения, то есть в какое-то количество эпох генерировать и выводить текст у необученной модели:

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

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

Отследим процесс обучения и рассмотрим тексты сгенерированные после 1, 9 и 15 эпох.

После 1 эпохи сеть сгенерировала последовательность, которая продемонстрирована на рисунке 2

```
Seed:
" ading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thought alice "
```

Рисунок 2 – Результат после 1 эпохи.

После 9 эпохи сеть также сгенерировала последовательность. Результат показан на рисунке 3.

```
Seed:
"  trouble
of getting up and picking the daisies, when suddenly a white
rabbit with pink eyes ran clos "
```

Рисунок 3 – Результат после 9 эпохи.

После 16 эпохи сеть сгенерировала текст, в нём уже просматриваются существующие слова, но также остались и непонятные. Какая-либо смысловая нагрузка отсутствует. Результат показан на рисунке 4.

```
Seed:
" as she could,
for the hot day made her feel very sleepy and stupid), whether
the pleasure of making  "
```

Рисунок 4 – Результат после 16 эпохи.

Обучать сеть на большем количестве эпох оказалось затруднительно, это связано с тем, что обучение происходит слишком долгое время. Но даже из получившихся результатов можно заметить, что с увеличением эпох появляется всё больше осмысленных слов.

Выводы.

В ходе выполнения лабораторной работы была построена и обучена нейронная сеть, которая позволяет генерировать текст на основе произведения Льюиса Кэрролла «Приключения Алисы в Стране Чудес». Был также написан собственный CallBack, позволяющий отслеживать прогресс сети. В результате своего обучения сеть смогла генерировать неосмысленные тексты, в которых порой встречаются существующие слова.

ИСХОДНЫЙ ТЕКСТ

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

ПРИЛОЖЕНИЕ А

```
import numpy
from keras.callbacks import Callback
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

# Загрузка текста
filename = "/content/sample_data/wonderland.txt"
raw_text = open(filename).read()
# Преобразование всех символов в нижний регистр
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 my_callback(Callback):
   def __init__(self, epochs):
      super(my_callback, 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("\"", ''.join([itc[value] for value in pattern]), "\"")

   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)]

n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)

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

# Создание модели
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'))
# Инициализация параметров обучения
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, my_callback([1, 8, 15])]

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