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 _____ Бессуднов Г. И.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2021

Цель работы

Построить генеративную искусственную нейронную сеть на основе “Алисы в стране чудес”.

Основные теоретические положения

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.

Выполнение работы

По теоретическим сведениям была построена и натренирована модель на основе «Алисы в стране чудес». Был написан собственный `CallBack` класс для проверки работы модели на каждой третье эпохе. Так же был добавлен `CallBack` типа `TensorBoard` для мониторинга модели средствами TensorBoard. Код программы представлен в Приложении А.

Проанализируем сгенерированные тексты на некоторых эпохах:

1. Эпоха 0:

   «the the

the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the»

Видно, что сеть просто генерирует самую часто встречающуюся последовательность символов.

2. Эпоха 6:

   «*e oo the toeee and she woile th the woile th the white th the white th the white th the white the woil*»

   Появляется больше разнообразия слов, но начинают встречаться также несуществующую слова. Смысла нет никакого.

3. Эпоха 12:

   «*e hoot an the woule her head and the tas at the cane and the was oo the tooee, and the was so an anl taee to tee shet shee and the was oo the tooee and the was so an anl thee an in whth the rooee of the goure, and the was so an thee an in whsh the woudd and the was so an anl thee an in whth the rooee of the goure, and the was so an thee an the was oo the tooee and the was so an thee an in whsh the rooee of the caree in the*

had andne the had aeden to the tooee, and the was so an anl tae an ieree anong the had aeden to the tooee, and the was oo toek to her sh the tooee, and the was so an anl tae an ieree anong the had aeden to the tooee, and the was so an anl tae an ieree anong the had aeden to the tooee, and the was oo toek to her sh the tooee, and the was so an anl tae an ieree anong the had aeden to the tooee, and the was so an anl tae an ieree anong the had aeden to the tooee, and the was oo toek to her sh the tooee, and the was so an anl tae an ieree anong the had aeden to the t»

Все так же есть несуществующую слова, но само разнообразие слов стало больше, смысл по-прежнему отсутствует.

Результат после 20 эпох выглядит следующим образом:

«e whst woiee oo the sooee of the sabbit sate to the sooe of the caree in a telyed of the had sote the thneg hardener, and she tas aol sonnk io the ciue of the soeee of the gade,

'i vhnn you dad toue said the monk, the horphon seneiked to herself, 'io soe teat to tee the mors of the goost, and the sae so tuee to the sooe of the sooee of the sabbit sate tiat she was notting an in aoo oo toake the had aoeneed and a lirtle oo the oede of the haree hare and the woede she cade to the that saree on the sipee hareen she caded the hitg of the garee ha the harter whs a little to tee the was ootting oo the tioee her aeain, and the was notting an inr toaee the cad not toen a lote oi the sooe of the sooee of the sooee of the soeee of the gareen,

'ie toept oo the cousorse in whe sagt,' said the monk turtee th the tueer.

'then' said the monk turtle andiily at il sooee.

'then toe threl sare to ae an anl ' said the monse sho and meverpere boing notting to the tooe.

'ieve you saan to be an»

Появляются абзацы и специальные символы как апострофы и некоторые знаки препинания, присутствуют разнообразные слова. Смысла нет, есть несуществующие слова.

Скорее всего результаты были бы лучше, если бы генерация происходила не по буквам, а по словам. В таком случае хотя бы не генерировались несуществующие слова.

TensorBoard был использован для наблюдения за процессом обучения модели. Например, на рис. 1 представлен график потерь для каждой эпохе, на рис. 2 представлены гистограммы активации весов для разных слоев.

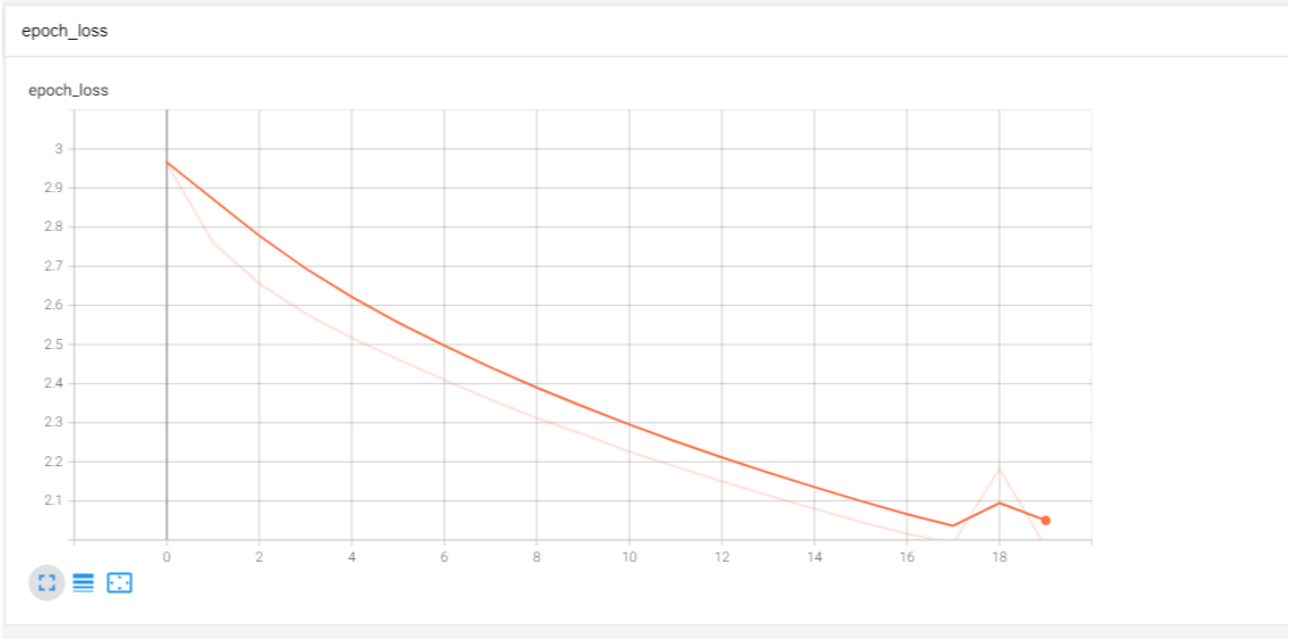

Рисунок 1 - График потерь

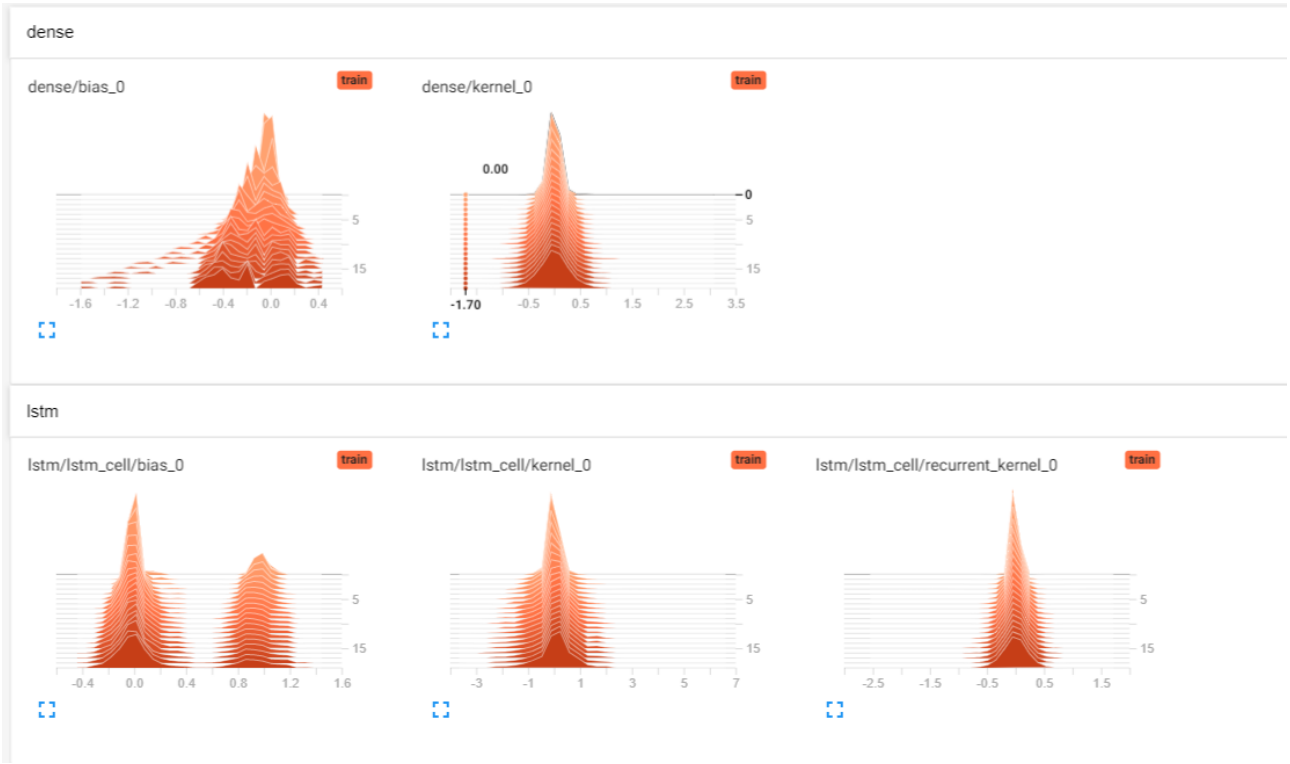

Рисунок 2 - Гистограммы активации слоев

Выводы

В ходе выполнения работы была построена генеративная искусственная нейронная сеть на основе произведения “Алиса в стране чудес”. Так же был реализован класс `CallBack` для наблюдения за процессом обучения модели и проведено наблюдение посредством TensorBoard.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ MAIN.PY

```
import numpy
from keras.callbacks import Callback
from keras.callbacks import TensorBoard
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 pathlib import Path
from keras.models import load_model

def generateText(model):
    start = numpy.random.randint(0, len(dataX)-1)
    pattern = dataX[start]

    result = []
    # generate characters
    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.append(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)]

    result = "".join(result)
    return result
```

```
    class TextGeneratorLogger(Callback):
        def on_epoch_end(self, epoch, logs=None):
            if epoch % 3 == 0:
                text = generateText(self.model)
                with open("generated_" + str(epoch) + ".txt", "w") as
file:
                    file.write(text)

    path = Path("wonderland.txt")
    raw_text = open(path.absolute()).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 = 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}.hdf5"
    checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')

    callbacks_list = [
                        checkpoint,
                        TensorBoard(
                            log_dir='my_log_dir',
                            histogram_freq=1,
                            embeddings_freq=1,
                        ),
                        TextGeneratorLogger()
                    ]


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

    del model
```

```
model = load_model("weights-improvement-20.hdf5")
text = generateText(model)
with open("finalStory.txt", "w") as file:
    file.write(text)
```