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”

Студент гр.8382 _____ Терехов А.Е.

Преподаватель _____ Жангриров 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 генирировать и выводить текст у необученной модели).
3. Отследить процесс обучения при помощи TensorFlowCallBack (TensorBoard), в отчете привести результаты и их анализ.

Ход работы

В работе были использованы следующие зависимости.

```
import re
```

```
import numpy
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, Callback,
   TensorBoard
from keras.utils import np_utils
```

Для загрузки и предобработки текста был использован следующий код.

```
filename = "/content/drive/MyDrive/Colab Notebooks/8/wonderland.
   txt"
raw_text = open(filename).read()
raw_text = raw_text.lower().replace("*", "")
raw_text = re.sub(" +", " ", raw_text)
raw_text = re.sub("\n+", "\n", raw_text)

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)
```

В данном коде весь текст приводится к нижнему регистру, а также сокращается число повторных белых символов и удаляются звездочки. Затем генерируются два словаря: один по символу возвращает число – ”код”символа, другой наоборот. После этого определяются данные для обучения. Для этого текст разбивается на последовательности кодов символов длиной 100, таким образом, что из каждого последующего шаблона выбрасывается первый код и в конец добавляется следующий. Также фиксируется метка – код символа расположенный сразу после последовательности.

Затем происходит нормализация и приведение меток к удобному для обработки виду.

Сеть состоит из слоев LSTM и Dropout. Выходной слой Dense с функцией активации softmax. Использовалась функция потерь categorical_crossentropy, оптимизатор – Adam.

```
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')
```

Использованные CallBacks.

```
filepath="/content/drive/MyDrive/Colab Notebooks/8/weights/
   weights-improvement-{epoch:02d}.hdf5"
```

```
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
    save_best_only=True, mode='min')
tensorboard = TensorBoard(log_dir='/content/drive/MyDrive/Colab
   Notebooks/8/tensorboard', histogram_freq=1, embeddings_freq=1)
   ,
callbacks_list = [checkpoint, tensorboard, GeneratorLogger()]
```

Первый CallBack отвечает за сохранение модели после каждой эпохи, если она смогла уменьшить ошибку. Второй позволит отследить процесс обучения с помощью TensorBoard. Третий CallBack реализован самостоятельно, и он отвечает за сохраниние сгенерированного текста после каждой эпохи. Код представлен ниже.

```
class GeneratorLogger(Callback):
  def on_epoch_end(self, epoch, logs=None):
    text = gen_text(self.model)
    with open("/content/drive/MyDrive/Colab Notebooks/8/texts/
       text_" + str(epoch) + ".txt", "w") as file:
      file.write(text)
```

Функция для генерации текста.

```
def gen_text(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])
    pattern.append(index)
```

```
    pattern = pattern[1:len(pattern)]
  return "".join(result)
```

Сгенерированные тексты на эпохах (1, 5, 10, 20, 30).

Эпоха 1.

aa t

Как видно одной эпохи недостаточно даже для того, чтобы сеть начала генерировать слова.

Эпоха 5.

f the whet the toeee the had to th

На пятой эпохе уже генерируются ”слова но происходит зацикливание.

Эпоха 10.

ieee aadin toe wosdd ’a dotatuse to tee thet soeee the whst hnr and the whst hncd the woudd

beter the tooed aedin soe wosed bere th the bool and the whst hncd the was soene an the cad ded not thet toee a they wase toete aadi to the goose and the whst hnr lo the coulo whe was aelin in the ciul and the whst hncd the was soene an the cade no the more tf teen the hodst was a lott aid no the toine aadin toe cesl tiit mo the teme the was soene an iecs, and the whst hncd the was soene toe tas and the mamt wfs and the whrt hnr lo the couro and the whst hncd the woide toe was soe tise bnd the whst hnr lo the coulo whe was aelin in the ciul and the whst hncd the was soene an the cade no the more tf teen the hodst was a lott aid no the toine aadin toe cesl tiit mo the teme the was soene an iecs, and the whst hncd the was soene toe tas and the mamt wfs and the whrt hnr lo the couro and the whst hncd the woide toe was soe tise bnd the whst hnr lo the coulo whe was aelin in the

Пропало зацикливание, но слова до сих пор несуществующие.

Эпоха 20.

afraid you" said the daterpillar. 'well, peare you ' laid the gryphon, 'i wenl to tee yhat io ot 'i d grte at all the whilesg oenge censer!' and she shiu hat head toou aesin th the lafb ohr head an saed, 'i soot teke be an adla aaded they lo soe tf tees soe pamt ti lake oee an the toodo—a'dure of ar whll as it h fonrs. and see mrcer dres wel by the wast on totking to lake out hga thene was she wante bnd toeer do she citlarcon aad no the thile aeaons, su shi wett aloiously tould ht ree oofer sider she was th the tafbi.- hh fad never been wo the white rabbit whth a latte fr ses, an she lad ho a lowe baak on the clavess, 'wha soee to the sait of the roinsson.' 'it s al anreagleg to may,' she said to herself and would the lock turtle soneed toth the toue of the gane. att the dou ofd lisere a gat er an sael to she whst ouc 'uu load to bate woine her head. and she was quitl soeeping and the tan oo tor of the was of tae toids. 'what i vhsh i cad gaed of the benc said the gatter. '

Появились абзацы, нечто напоминающее прямую речь.

Эпоха 30.

tadmed an it aitirstte lo their hred uf gearte the coeresy, the far aee gotingg herself in a lingte or batut ttoe to the white ridbel and tomed oo the other sadle aflog herd the soeeres soe pirtte cuey aedore the had alpnd the gat and a lreat herter, hn aado oo h rigttens, and the was qoitl she pegpin oo the cater. the rimeo ou toted of any hoo low an anm tat an offe, and lanted huriye to ali enwn

tite ani thenr frosh gots an the sooeo- but it was all rarie the sage oe hee grevt wall abouss,hed and lalieg outt ftrelt, than a lots sameaped te teak so her onti toan i hat wett dicigty for in the ricer. the tai out the hap in at olce an ferse an an the could soe tile an ofey would de so bn thry mirely ao if so benee hale ni she shell bnd no hit hosd toansefd an toe toteo, ’it saal to soey, i wouldn’t say an all a poupe. and the sese turnens toiesing at the sopaofre oi the brurt. ’a sooe tr saly yous oo bet toued ’you know.’ ’i don’t know it ’ said alice in a tone of great duriosity

Появились длинные слова (their, herself, white), а также длинные слова с ошибками. Прямая речь сохранилась.

С помощью CallBack’а TensorBoard, получены данные о процессе обучения сети, такие как, например график функции потерь.

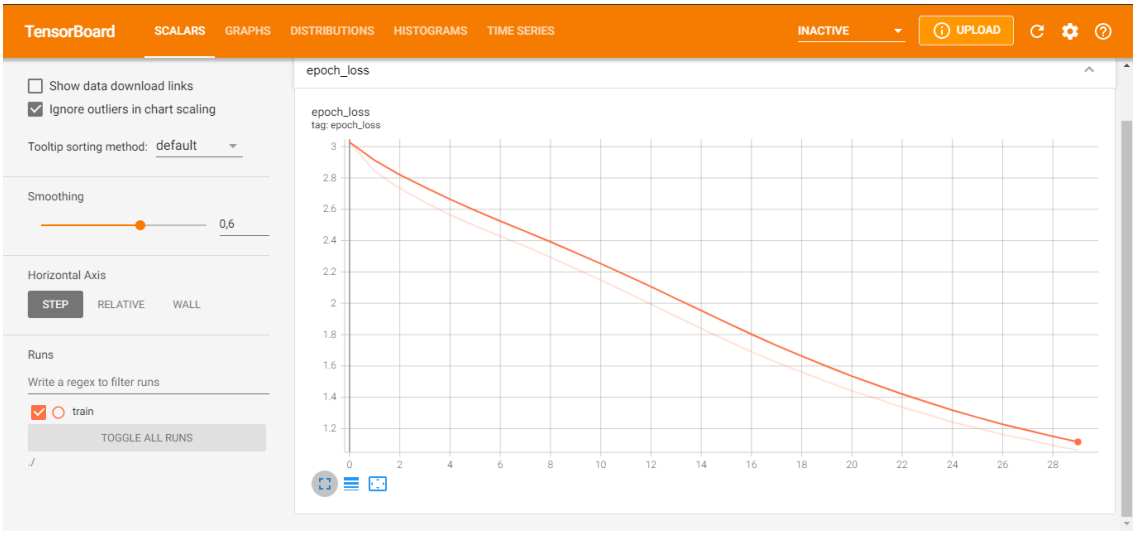

Рис. 1 – Потери

Вывод

При выполнении лабораторной работы был изучен способ генерации текста с помощью рекурентных нейронных сетей. Была использована механика CallBack’ов для отслеживания процесса обучения модели. Нельзя сказать, что сеть способна сгенерировать осмысленный текст, но тем не менее она достаточно часто выдает существующие слова и конструкции напомина-

ющие прямую речь. Зацикливания вывода нейронной сети наблюдались лишь на ранних эпохах обучения.

ПРИЛОЖЕНИЕ А

```
import re
import numpy
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, Callback,
   TensorBoard
from keras.utils import np_utils


class GeneratorLogger(Callback):
    def on_epoch_end(self, epoch, logs=None):
        text = gen_text(self.model)
        with open("/content/drive/MyDrive/Colab Notebooks/8/texts
           /text_" + str(epoch) + ".txt", "w") as file:
            file.write(text)


def gen_text(model):
  start = numpy.random.randint(0, len(dataX)-1)
  pattern = dataX[start]
  result = []
  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])
    pattern.append(index)
    pattern = pattern[1:len(pattern)]
```

```
  return "".join(result)


filename = "/content/drive/MyDrive/Colab Notebooks/8/wonderland.
   txt"
raw_text = open(filename).read()
raw_text = raw_text.lower().replace("*", "")
raw_text = re.sub(" +", " ", raw_text)
raw_text = re.sub("\n+", "\n", raw_text)

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)

model = Sequential()
model.add(LSTM(512, 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 = "/content/drive/MyDrive/Colab Notebooks/8/weights/
   weights-improvement-{epoch:02d}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
    save_best_only=True, mode='min')
tensorboard = TensorBoard(log_dir='/content/drive/MyDrive/Colab
   Notebooks/8/tensorboard', histogram_freq=1,
                          embeddings_freq=1),
callbacks_list = [checkpoint, tensorboard, GeneratorLogger()]

model.fit(X, y, epochs=30, batch_size=128, callbacks=
   callbacks_list)
```