МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Искусственные нейронные сети»

Тема: «Генерация текста на основе “Алисы в стране чудес”»

Студент гр. 7381 _____ Тарасенко Е.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

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, в отчете привести результаты и их анализ.

Ход работы.

Сначала реализуем подготовку обучающих данных, построение и обучение модели, описанные в методических указаниях. Модель имеет следующую архитектуру:

```
model = tensorflow.keras.models.Sequential()
model.add(tensorflow.keras.layers.LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(tensorflow.keras.layers.Dropout(0.2))
model.add(tensorflow.keras.layers.Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Теперь необходимо организовать генерацию текста полученной нейронной сетью после определенных эпох для мониторинга улучшения результатов в зависимости от времени обучения. Для этого был разработан собственный Callback, вызывающий после определенных эпох обучения сети функцию генерации текста:

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

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

На момент 5-й эпохи генерируемый текст почти полностью состоит из повторяющихся частей. Количество сгенерированных реально существующих английских слов предельно мало (пока что только предлоги):

```
oete toet the woete toet aadin  'the woued to tee toet to tee toet to the woete to tee toete to the woete to the woete the woete the woete toe whe woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to the woeee th the woeee to
```

На момент 10-й эпохи ситуация улучшилась: текст уже не состоит из странных повторяющихся «фраз», а в написанных словах стали прослеживаться черты реального английского языка:

```
g the care and the cruld  and the woole whi woold whu doen the woole oo toe thet the would bed toene the would  and the woold whu doond the woole oo toe thet the would  and the woold th the woile to ter toe woiee oo the would  and the woold th the woile to tel toe woile oo the woile oo the woide  and the woold whi woold whi had hot the woile oo toe the would th the toone the would  and the woold whi woold whi had hot
```

the woile oo toe thet the woued bed toene the would and the woold whu doond the woole oo toe toene to the would and the woold th the woile to tel toe woile oo the woile oo the woide and the woold whi woold whi had hot the woile oo toe the would th the toone the would and the woold whi woold whi had hot the woile oo toe thet the woued bed toene the would and the woold whu doond the woole oo toe toene to the would and the woold th the woile to tel toe woile oo the woile oo the woide and the woold whi woold whi had hot the woile oo toe the would th the toone the would

Перед началом 15-й эпохи программа смогла сгенерировать уже более разнообразный текст, наполненный либо реальными словами, либо словами, которые очень сильно на таковые похожи (с отличиями в одну или две буквы):

the monke
in sou doold to the tooes.'

'i don't know it ' said the monke, and the gorphon sar so the pore, 'the mort of thin soeee to toink the sooe of the woued hareen at the cateer and aldcete the had hot doen the care and the cateer aadut the was so the kint and the pabbit sate the was soteng the rabbit and then sae i rany doon the toins on the tooed to tee that she was not in the cane and the career, and the was soteng to the tiie at the cad fot io the was so thek it was the winte thit har soe tine thi had seeen her hend thet she was not in the cane and the career, and the was soteng to the tiie at the cad fot io the was so thek it was the winte thit har soe tine thi had seeen her hend thet she was not in the cane and the career, and the was soteng to the tiie at the cad fot io the was so thek it was the winte thit har soe tine thi had seeen her hend thet she was not in the cane and the career, and the was soteng to the tiie at the cad fot io the was so thek it was the winte thit

Начало 20-й эпохи. Программа стала явно «исправляться» и допускать все меньше «опечаток», хотя результат по-прежнему не такой хороший. Однако положительной чертой процесса обучения может являться появление некоторых словосочетаний, явно намекающих на обучающий материал, например, «…white rabbit was so…»:

to the gorhoong that she was sot aadin th the had hen herd and the woudd hot and noee th the was so ani then she had been whit hl whsh a sary oo the was to toek in the wood, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt

of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was soe madt of the tore, and she white rabbit was s

Обучение в течение 20-ти эпох заняло несколько часов, поэтому дальнейшие эксперименты не проводились, однако на данном этапе можно сделать вывод, что продолжительность обучения положительно сказывается на качестве генерируемого текста. Таким образом, лучшие результаты (меньшие потери) достигались по истечение именно последней, 20-й, эпохи:

toen the was so the kiakte oo the so be a coore the ragt of the tabbit so baald to tietk toat the pabt of the tabbit sar sh the tooe,
and the sooe oater aadun tee thet she was to the wiite tar to tee that she was to the white tar to tee that she was to the kintle goose tf the had feveen the rage of the garter and the carer and the waite iar badut the whitg tas to teek to her hort an inr si the wai oo the was to toenk to the woide
the faree hareen wery socer woice, and the tas goon the couro sf the oabe tf the tabbit soeee of the was to aeiin thet sam the rabbit sare the was so the wiitg theeg the had been woin the had beoee woine to the whitg tab it aalir the aadk in airiess touh oh the tooe,
and the sooe har aelin in a mortee oo the hag feveen the riget world her head to teye the had hoo not al anleoss the white tar to the white table, and the white rabbit was soe cint an in sotee an socerion of the tabli of the care and the carerpillar sooe the whilg she was gown the dan and the

В тексте, сгенерированном обученной моделью, присутствуют в достаточном количестве реальные слова и даже их сочетания (that she was, the white rabbit was, her head и т. д.). Помимо этого, присутствует немалое количество словосочетаний, сильно напоминающих человеческую речь, за исключением грамматических ошибок (wery soccer woice). Все это на данном этапе слабо напоминает осознанный и связный рассказ, однако все же, однозначно, не может не удивлять.

Выводы.

В ходе выполнения данной лабораторной работы была разработана и обучена модель, которая специализируется на самостоятельной генерации текста на основе обучающего материала, в качестве которого выступило произведение Льюиса Кэрролла «Приключения Алисы в стране чудес». В ходе эксперимента выяснилось, что по ходу обучения, генерируемый текст приобретает все больше и больше реально существующих в английском языке слов, или, по крайней мере, включает в себя таковые с небольшими грамматическими ошибками. Однозначным минусом этой сети остается слишком большое время обучения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import numpy
import tensorflow


def generating(model=None, epoch='best'):
    # load ascii text and covert to lowercase
    filename = "wonderland.txt"
    raw_text = open(filename).read()
    raw_text = raw_text.lower()

    # create mapping of unique chars to integers, and a reverse mapping
    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))

    # summarize the loaded data
    n_chars = len(raw_text)
    n_vocab = len(chars)
    print("Total Characters: ", n_chars)
    print("Total Vocab: ", n_vocab)

    # prepare the dataset of input to output pairs encoded as integers
    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 = tensorflow.keras.utils.to_categorical(dataY)

    if not model:
        epoch = 'best'
        # define the LSTM model
        model = tensorflow.keras.models.Sequential()
        model.add(tensorflow.keras.layers.LSTM(256, input_shape=(X.shape[1],
X.shape[2])))
        model.add(tensorflow.keras.layers.Dropout(0.2))
        model.add(tensorflow.keras.layers.Dense(y.shape[1], activation='softmax'))

        # load the network weights
        model.load_weights("weights-improvement-20-1.9073.hdf5")
        model.compile(loss='categorical_crossentropy', optimizer='adam')

    # pick a random seed
    start = numpy.random.randint(0, len(dataX)-1)
```

```
    pattern = dataX[start]
    #print("Seed:")
    #print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")

    # generate characters
    predicted_text = ''
    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)
        predicted_text += str(int_to_char[index])
        seq_in = [int_to_char[value] for value in pattern]
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
    with open(epoch + '.txt', 'w') as textfile:
        textfile.write(predicted_text)
    print("\nPredictions completed!")


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

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


def preparation():
    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))

    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 = tensorflow.keras.utils.to_categorical(dataY)

    model = tensorflow.keras.models.Sequential()
    model.add(tensorflow.keras.layers.LSTM(256, input_shape=(X.shape[1],
X.shape[2])))
    model.add(tensorflow.keras.layers.Dropout(0.2))
    model.add(tensorflow.keras.layers.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 = tensorflow.keras.callbacks.ModelCheckpoint(filepath, monitor='loss',
verbose=1, save_best_only=True, mode='min')
    callbacks_list = [checkpoint, GetText([4, 9, 14, 19])]

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


if __name__ == '__main__':
    preparation()
    generating()
```