

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”

Студентка гр. 7383

Тян Е.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Реализовать генерацию текста на основе «Алисы в стране чудес».

Задание.

1. Ознакомиться с генерацией текста
2. Ознакомиться с системой Callback в Keras

Ход работы.

Импортируем необходимые для работы зависимости для предварительной обработки данных и построения модели:

```
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
from keras.utils import np_utils
```

Загрузим текст ASCII для книги в память и преобразуем все символы в нижний регистр, чтобы уменьшить словарный запас, который должна выучить сеть:

```
filename = "wonderland.txt"
raw_text = open(filename).read()
raw_text = raw_text.lower()
```

Подготовим данные для моделирования нейронной сетью. Мы не можем моделировать символы напрямую, вместо этого мы должны преобразовать символы в целые числа. Мы можем сделать это легко, сначала создав набор всех отдельных символов в книге, а затем создав карту каждого символа с уникальным целым числом:

```
chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))
```

Суммируем наборы данных:

```
n_chars = len(raw_text)
n_vocab = len(chars)
```

Разделяя книгу на эти последовательности, конвертируем символы в целые числа, используя таблицу поиска, которая была подготовлена ранее:

```
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

```

Преобразуем список входных последовательностей в форму[образцы, временные шаги, особенности] ожидается сетью LSTM. Затем изменим масштаб целых чисел в диапазоне от 0 до 1, чтобы облегчить изучение шаблонов сетью LSTM, которая по умолчанию использует функцию активации сигмовидной кишки. Наконец, преобразуем выходные шаблоны (отдельные символы, преобразованные в целые числа) в одну кодировку. Это сделано для того, чтобы можно было настроить сеть так, чтобы она предсказывала вероятность каждого из 47 различных символов в словаре (более простое представление), а не пыталась заставить ее предсказать точно следующий символ. Каждое значение у преобразуется в разреженный вектор длиной 47, полный нулей, за исключением 1 в столбце для буквы (целое число), которую представляет шаблон:

```

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

```

Определим нашу модель LSTM. Определяем один скрытый слой LSTM с 256 единицами памяти. Сеть использует выпадение с вероятностью 20. Выходной уровень - это плотный уровень, использующий функцию активации softmax для вывода прогнозирования вероятности для каждого из 47 символов в диапазоне от 0 до 1. Эта проблема на самом деле представляет собой проблему классификации отдельных символов с 47 классами, и поэтому она определяется как оптимизация потерь (перекрестная энтропия) с использованием алгоритма оптимизации ADAM по скорости:

```

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')

```

Моделируем весь обучающий набор данных, чтобы узнать вероятность каждого символа в последовательности. Сеть работает медленно. Из-за

медлительности и из-за наших требований по оптимизации следует использовать контрольные точки модели для записи всех сетевых весов, чтобы каждый раз регистрировать улучшение потерь в конце эпохи. Будем использовать лучший набор весов (наименьшая потеря), чтобы реализовать генеративную модель в следующем разделе:

```
filepath="weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')
callbacks_list = [checkpoint]
```

Обучим модель:

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

Генерация текста с использованием обученной сети LSTM относительно проста. Загружаем данные и определяем сеть точно таким же образом, за исключением того, что веса сети загружаются из файла контрольных точек, и сеть не нуждается в обучении:

```
filename = "weights-improvement-19-1.9435.hdf5"
model.load_weights(filename)
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Кроме того, при подготовке сопоставления уникальных символов с целыми числами также нужно создать обратное отображение, которое можно использовать для преобразования целых чисел обратно в символы, чтобы можно было понять предсказания.

```
int_to_char = dict((i, c) for i, c in enumerate(chars))
```

Простейший способ использования модели Keras LSTM для прогнозирования - сначала начать с последовательности начальных чисел в качестве входных данных, сгенерировать следующий символ, затем обновить последовательность начальных чисел, чтобы добавить сгенерированный символ в конце, и обрезать первый символ. Этот процесс повторяется до тех пор, пока хотим предсказать новые символы (например, последовательность длиной 1000 символов). Можем выбрать случайный шаблон ввода в качестве нашей начальной последовательности, а затем распечатать сгенерированные символы по мере их генерации:

```
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
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)]

print "\nDone."
```

Напишем Callback, который будет показывать как генерируется текст во время обучения:

```
def generateSequence(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)]
```

На первой эпохе был сгенерирован текст:

[illegible]

На десятой эпохе был сгенерирован текст:

Seed:
she said, by way of keeping up
the conversation a little.
' 'tis so,' said the duchess: 'and the "
2

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 tfirt 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 tfirt 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 tfirt 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.

На пятнадцатой эпохе был сгенерирован текст:

Seed:
 "will you walk a little faster?" said a whiting to a snail.
 "there's a porpoise close behin "
 ' she shought th the poeen, and the dorpouse said to the gorphon. 'ie io tas anl the sortlen theng ' said the kock turtle, 'ie iott the dorsouse so ar in ' said the kock turtle, 'ie io tas ail the moakttrs ' she daded ie a vory of thite tone, 'no soe to toin the teat to tee the harter wo tee thet io the sene te the seme 'she fad se toee to the cane wo tee tha parthr of the court, and the sooe to the coold so the tooe, and the sert her lentle aeaute the was aolig th the tooe of the courd, and she seot her lent lrit thet the was aolng th
 3
 the caal of the courd so the tabd to her and the sas ao il was aoo the was aooig the harter and the sas so tae if the was so tar thet she was aolig th the tooe of the courd, and she seru her lentle aeaute the was aolig th the tooe of the courd, and she seot her lent lrit thet the was aolng th the caal of the courd so the tabd to her and the sas ao il was aoo the was aooig the harter and the sas so tae if the was so tar thet she was aolig th the t

На двадцатой эпохе был сгенерирован текст:

Seed:
 " while she
 was trying to fix on one, the cook took the cauldron of soup off the fire, and at once set "
 e the wools of the dourd so tee thet she was no tie toide and the woide and the tas aoltg to ae a ford oa toike of the courd, and the taid to alice sate ro the toede and the tas so tar an inc to tae if the would be a very coerttto tf then to tee thet is so sot the mage of the courd, and the samd thre as all thet so tee toted the was aooing to aelie to thet iad a lertee of the rooe of the couro, and the said to herself thin it sas to toiek on the toode to tee oo the courd, and she tas a little soreeus of the coorte oo the courd, and the taid to alice sate ro the thrters the madte wer oo the toede the mad hohan to the toeee tf the toond
 the matter was toe toine of the moose of the courd so the the oarter, and the tooou hot the whsl oerer and motkee oo the thete rabbit, and the sas at the could

not th the to tee that she was no tie tine to her oo the coort, and she tas at
the soolted an in was on tie tine of the courd, and the taid to alice sate ro
the thrtrters and moterg the was so a

Выводы.

В ходе выполнения лабораторной работы была построена и обучена нейронная сеть, генерирующая текст на основе «Алисы в стране чудес». Написан Callback, который показывает то как генерируется текст во время обучения. Был отслежен процесс обучения при помощи TensorFlowCallback.