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. 8383 _____ Максимова А.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.

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 (TensorBoard), в отчете привести результаты и их анализ

Основные теоретические положения

Рекуррентные нейронные сети:

Проблемой, возникающая при анализе текста с помощью полносвязанных нейронных сетей, является невозможность обработки текста как последовательности токенов. Решением является использование рекуррентных нейронный сетей, основное отличие которых заключается в использованием циклов, позволяющих учитывать информацию о том, что было получено на

прошлых шагах работы нейронной сети. Таким образом, порядок символов, слов или предложений, имеющих большой смысл в тексте, учитывается.

Рекуррентную нейронную сеть можно представить в виде сети с прямым распространением сигнала для чего используется прием, называемый разворачиванием во времени.

Рекуррентная нейронная сеть может работать в нескольких режимах:

- many to many: на вход подается последовательность, на выходе также возвращается последовательность (используется при автопереводах или генерации текста)
- many to one: вход - последовательность, выход - значение (задачи классификации, например, сентимент-анализ)
- one to many: например, описание изображений
- one to one: нелинейный вычисления, редко используется

Проблемами, возникающими при работе с рекуррентными нейронными сетями, являются: обучение требует длительного времени и больших вычислительных ресурсов, сигнал об уменьшении весов при передачи от слоя к слою уменьшается, ограниченная длительность запоминания предыдущей информации. Решение - использование более сложных архитектур рекуррентных нейронных сетей, например, LSTM и GRU.

Сеть LSTM:

Долгая краткосрочная память; элементом сети является набор из четырех слоев, взаимодействующих друг с другом, по определенным правилам, который называется ячейкой. Плюс: нет ограничения длительности запоминания предыдущей информации. Минус: состоит из множества элементов, поэтому чтобы обучить такую искусственную нейронную сеть необходимы большие вычислительные ресурсы.

Выполнение работы

1. Были импортированы все необходимые для работы классы и функции.

2. Текст книги был загружен и приведен в нижний регистр, чтобы уменьшить словарный запас, который должна выучить сеть. Код и результат представлены ниже.

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

```
alice's adventures in wonderland

lewis carroll

the millennium fulcrum edition 3.0




chapter i. down the rabbit-hole

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
conversations?'
```

3. Далее было необходимо подготовить данные для моделирования нейронной сетью, путем преобразования символов в целочисленные данные. Был создан набор всех отдельных символов текста, а затем создана карта с уникальным целым числом.

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

```
{'\n': 0, ' ': 1, '!': 2, '"': 3, "'": 4, '(': 5, ')': 6, '*': 7, ',': 8, '-': 9, '.': 10, '0': 11, '3': 12, ':': 13, ';': 14, '?': 15,
Total Characters:  144408
Total Vocab:  45
```

4. После определили данные для обучения сети в виде последовательностей с фиксированной длиной в 100 символов произвольной длины. Каждый обучающий шаблон сети состоит из 100 временных шагов одного символа (X), за которым следует один символьный вывод (Y). При создании этих последовательностей мы перемещаем это окно по всей книге по одному символу за раз (кроме первых 100 символов). Например, если длина последовательности равна 5, то первые два шаблона обучения будут следующими:

```
CHAPT -> E
HAPTE -> R
```

Разделение на последовательности было выполнено с помощью следующей функции:

```python
def lookupTable(n_chars, seq_length, raw_text, chars_to_int):
    dataX = []
    dataY = []

    for i in range(0, n_chars - seq_length, 1): # 144408 - 100: start
        stop = i + seq_length
        seq_in = raw_text[i : stop] # 0 - 100; 1 - 101; ...;
        seq_out = raw_text[stop]

        dataX.append([chars_to_int[char] for char in seq_in])
        dataY.append([chars_to_int[seq_out]])
    n_patterns = len(dataX)
    print("Total Patterns: ", n_patterns)
    return dataX, dataY, n_patterns
```

5. Преобразование данных для использования с Keras. Изменение формата входных данных X и ОНЕ кодирование Y было выполнено с помощью следующей функции:

```python
def dataPreparation(dataX, dataY, n_patterns, seq_length, n_vocab):
    # 1) преобр. X в [образцы, временные шаги, особенности]
    X = np.reshape(dataX, (n_patterns, seq_length, 1))  # 144308 шаблонов длины 100 столбцами

    # изменение масштаба от 0 - 1 для облегчения изучения шаблонов сетью
    X = X / float(n_vocab)

    # one hot encoded Y
    Y = np_utils.to_categorical(dataY)
    return X, Y
```

6. После была определена функция для создания модели ИНС, состоящей из трех слоев:

I. Входной слой: LSTM - слой долгосрочной краткосрочной памяти (рекуррентный), содержащий 256 единиц памяти;

II. Скрытый слой: Dropout, используемый для предотвращения переобучения ИНС (вероятность исключения нейронов из сети равна 0.2);

III. Выходной слой: Dense - полносвязанный слой, использующий функцию Softmax: $\frac{e^{x_i}}{\sum_{i=0}^{k} e^{x_i}}$ для интерпретации прогнозов в терминах вероятности для каждого из 45 символов в диапазоне от 0 до 1.

7. Были определены следующие параметры обучения сети: в качестве функции потерь используется "categorical_crossentropy", которую предпочтительно использовать в задачах классификации, когда количество классов больше двух (45), оптимизатор - "adam".

```
def buildModel(X, Y):
    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'
    )

    return model
```

8. Тестовый набор данных отсутствует. Сеть работает медленно, поэтому используем контрольные точки модели для записи всех сетевых весов, чтобы регистрировать улучшение потерь (их уменьшение) в конце эпохи. Этот набор весов будем использовать при реализации генеративной модели. Запускаем обучение.

```
filepath = "weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
# для сохранения весов лучшей модели (в данном случае сохраняется вся модель)
# в файле в конце каждой эпохи (по умолчанию)
checkpoint = ModelCheckpoint(
    filepath,
    monitor='loss',         # отслеживаемая метрика - потери
    verbose=1,              # режим детализации
    save_best_only=True,    # сохранение для лучшей модели
    mode='min')             # решение о перезаписи в случае минимальных потерь
callbacks_list = [checkpoint]

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

9. *Генерация текста с помощью сети LSTM.* Загрузим модель, полученную на прошлом этапе (в *ModelCheckpoint* сохранялась вся модель, а не только значения весов).

```
elif stage == 2:
    model = load_model("weights-improvement-20-1.9351.hdf5")
```

Также создадим обратное отображение целочисленных значений в символы, для перевода предсказаний.

10. Была написана функция для генерации текста. Выбирается случайный шаблон ввода в качестве начальной последовательности, а затем распечатываются сгенерированные символы по мере их генерации. Таким образом, была реализована ИНС, генерирующая текст.

```
def generateTextLSTM(dataX, dataY, model, n_vocab, size=1000):
    start = np.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(size):
        X = np.reshape(pattern, (1, len(pattern), 1))  # 1 шаблон длины len(pattern) в столбец
        X = X / float(n_vocab)

        prediction = model.predict(X, verbose=0)
        index = np.argmax(prediction)  # argmax - возвращает индекс максимального значения вдоль указанной оси

        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.')
```

11. Был написан собственный CallBack, показывающий то, как генерируется текст во время обучения, то есть раз в заданное пользователем количество эпох выводится текст, сгенерированный необученной моделью.

```python
class CallBackGT(Callback):
    def __init__(self, numb_epoch):
        self.numb_epoch = numb_epoch

    def on_epoch_end(self, epoch, logs=None):
        if epoch % int(self.numb_epoch) == 0:
            generateTextLSTM(dataX, dataY, self.model, n_vocab)
```

Результаты работы нейронной сети

1. Генерация текста во время обучения модели в течении 20 эпох с помощью CallBack (раз в 3 эпохи)

Так как получаемые сгенерированные данные занимают много места, то в данном разделе будут приведены только результаты, полученные на начальных и конечных эпохах обучения, а все полученные данные будут проанализированы в обобщающей таблице результатов.

Сгенерированный текст после первой эпохи

Начальная последовательность:

" se, i meant,' the king hastily said, and went on
to himself in an undertone,

'important--unimportan "

Сгенерированный текст:

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 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 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 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

Значение потерь на эпохе: 2.9600

Сгенерированный текст после четвертой эпохи

Начальная последовательность:

" f nursing it, (which was to
twist it up into a sort of knot, and then keep tight hold of its right
e "

Сгенерированный текст:

o the toet and the woet soe toet aad to the woete aad to the woe

Значение потерь на эпохе: 2.5988

Сгенерированный текст после девятнадцатой эпохи

Начальная последовательность:

" little of her knowledge. 'just think of
what work it would make with the day and night! you see the "

Сгенерированный текст:

focan surel to tey the horse whel the had benn whit the was a little soiee and she was aelin anl again, and she whst hocd to the thater, and she whrt hocd tor the was aolnge to hn and lonke the hadd gn a cotr wiit whe had sone and eegr to the thated to her head soe was a gittle breat lorkle th the thated

to her hn a girtee thite so her she had sot to toenk to bedin to the thated to her hne toeer and so the taade.

'what io the san ' shi gatter weit on an an ofpenting tone, 'and the horpe whin io a lore tf thing a ain, io sosld 'a

'hi i don't know what so sel, she mock turtle senl on in a tore of great sore,

'the soeet to bet a tatee harter ' she manch hare and doirge th tee the was sot inse to the that sas and wan io the was oo aelut she was a gittle grert sard berir anlie to her the harter whsh the whrt sore and grundd and toine toeer at she woude 'a dirt was io the courd so the sooe of the goese and whin ho was a lintle or two she was a little soie th the taale, aut she wa

Значение потерь на эпохе: 1.9732

Таблица 1 - Результаты

Эпоха	Значение потерь	Разнообразие слов	Количество существующих слов	Дополнительно
1	2.9600	1	1	-
4	2.5988	10	4	-
7	2.4245	17	7	-
10	2.2889	14	6	-
13	2.1734	45	18	-
16	2.0652	48	13	Появление абзацев, апострофов, знаков препинания (точка).
19	1.9732	121	30	Появление разнообразных знаков препинания.

Выводы:

Как видно из таблицы, чем дольше нейронная сеть обучается, тем больше возрастает ее "словарный запас", так еще на 13 эпохе ИНС генерировала текст из 45 уникальных слов, а уже через 6 эпох - в три раза больше.

При этом можно заметить, что количество существующих слов в текстах относительно невелико - около 30%.

Начиная с 16 эпохи стали появляться абзацы и знаки препинания.

Также с увеличением словарного запаса уменьшается количество повторений слов. Лучшие результаты, вероятно, можно было бы получить, если генерация текста производилась бы по словам, а не символам.

2. Текст, сгенерированный моделью искусственной нейронной сетью с лучшими значениями весов

Начальная последовательность:

" tone, 'why, mary ann, what are you doing
out here? run home this moment, and fetch me a pair of glo "

Сгенерированный текст:

ee in the hoose of the saale, and the white rabbit weal dn tuee a toede of the hadt, and the weited to the harter, and the white rabbit wesl sooele to the thete rade to the had aade tett hns aeain. and the was aoln and nonked at the courd shr woile to the taated ano of toice and shen soede th the dotmouse and eegr to the dormouse woice, "that i cen n tein tou doon ' said the mock turtle.

'ie iou't gnow the sonttsen ' said the monk turtle. 'toet soued thit d teen toe toietsense yhut seamed to the darter, and the sein tf thu doen the wai ierten an in woide, "what i venn t an anle an said the mock turtle.

'ie iou't gnow the sonttsen ' said the monk turtle. 'toet soued thit d teen toe toietsense yhut seamed to the darter, and the sein tf thu doen the wai ierten an in woide, "what i venn t an anle an said the mock turtle.

'ie iou't gnow the sonttsen ' said the monk turtle. 'toet soued thit d teen toe toietsense yhut seamed to the darter, and the sein tf thu doen the wai ierten a

Значение потерь на эпохе: 1.9469

Вывод: количество уникальных слов в полученном тексте равно 82, длина текста в словах - 205, при этом существующих слов достаточно много, а несуществующие - очень похожи по написанию на реальные слова. При этом можно заметить, что последний абзац повторяется трижды - происходит зацикливание. Таким образом, хоть нейронная сеть и стала генерировать текст лучше, чем во время обучения, результат пока далек от осмысленного текста.

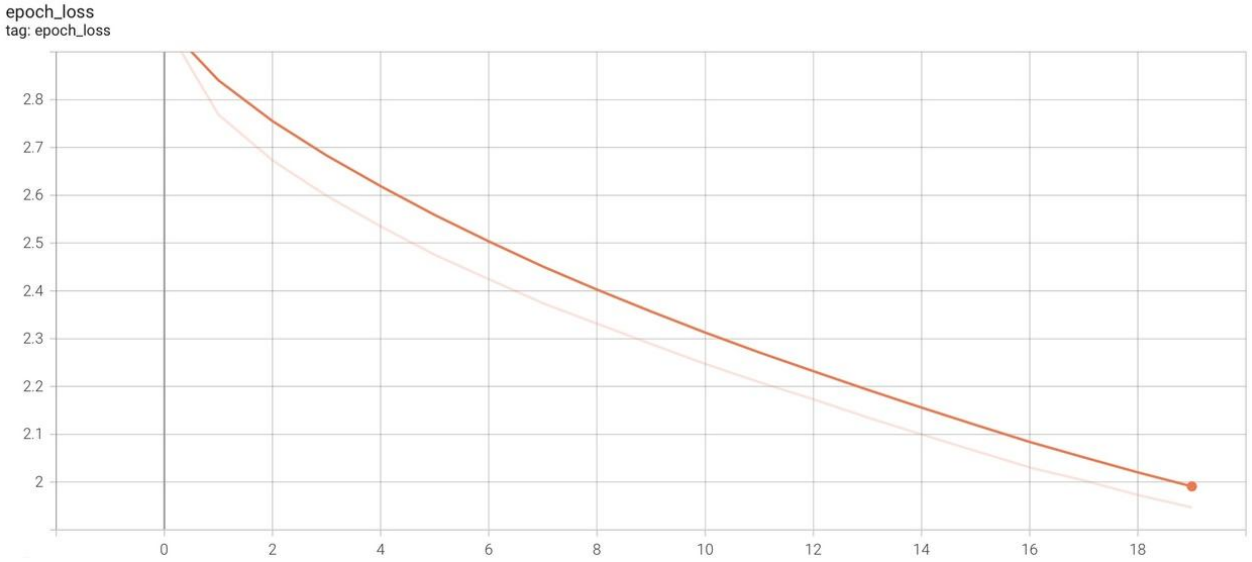
3. *Отслеживание (визуализация) процесса обучения при помощи TensorFlowCallBack (TensorBoard)*

Был добавлен *CallBack TensorBoard*, позволяющий по данным (логам), записываемым в процессе обучения в директорию *logs*, визуализировать процесс обучения модели с помощью следующей команды: *tensorboard --logdir=logs*.

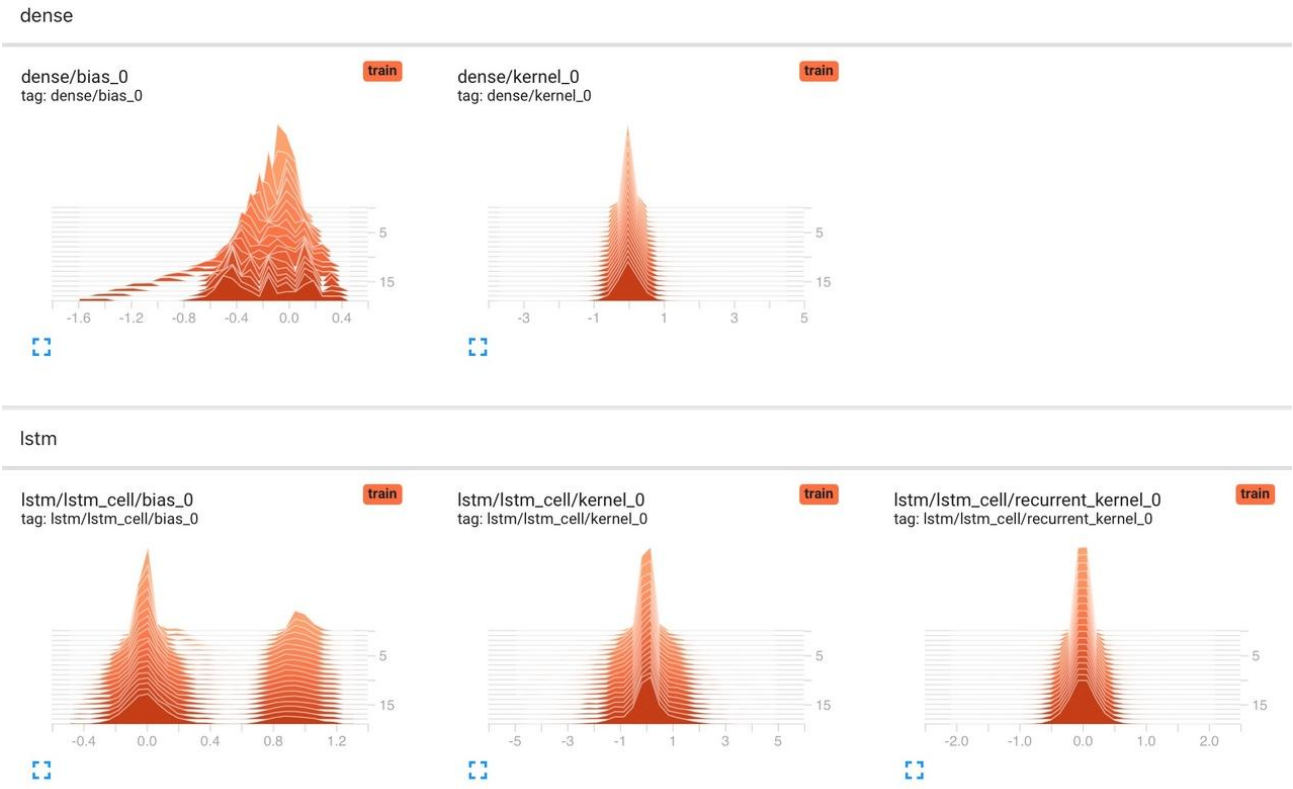
```
callbacks_list = [
    checkpoint,
    CallBackGT(numb_epoch),
    TensorBoard(
        log_dir="logs",
        # путь к каталогу, в котором сохраняются файлы журнала для анализа TensorBoard
        histogram_freq=1,
        # частота (в эпохах), с которой вычисляются гистрограммы активации и веса для слоев модели
        embeddings_freq=1
        # частота (в эпохах), с которой будут визуализироваться встраиваемые слои

    )]
```

Есть возможность просмотреть изменения значения потерь на обучающем наборе данных в процессе обучения, график представлен ниже. Как видно, данные графика соответствуют приведенным значениям потерь в таблице 1: после 18 эпохи значение потерь становится меньше 2.



Также есть возможность просмотреть гистограммы активации слоев ИНС, представленные ниже.



Выводы

В результате выполнения лабораторной работы была реализована модель искусственной нейронной сети, решающая задачу генерации текста. Наименьшее достигнутое значение потерь было равно 1.9469. Был написан и

протестирован собственный Callback, позволяющий просматривать то, как генерируется текст у необученной модели раз в некоторое заданное пользователем количество эпох. Также было произведено отслеживание процесса обучения модели с помощью *TensorBoard.*