**Отчет по лабораторной работе "Генерация последовательностей"**

Четвериков Виталий Анатольевич, М8О-107М-19

Номер в группе: 10, Вариант 04

**Цель работы**

Научиться генерировать тексты/последовательности с помощью рекуррентных нейронных сетей. Изучить архитектуры различных нейронных сетей пригодных для этой задачи.

**Используемые входные данные**

В качестве набора входных данных используется текст книги «Kachina Dolls: The Art of Hopi Carvers», источник «проект Гутенберг».

**Предварительная обработка входных данных**

Для тренировки нейросетевой модели данные(текст книги) необходимо предварительно подготовить. Для начала все символы преобразованы в нижний регистр, чтобы уменьшить словарный запас, который должна выучить сеть. Сложно моделировать символы напрямую, вместо этого преобразуем символы в целые числа. Для этого создадим набор всех отдельных символов в книге, а затем поставим в соответствие каждому символу уникальное целое число. Возможно некоторые символы, мы могли бы удалить для дальнейшей очистки набора данных, что уменьшило бы словарный запас и вероятно улучшило бы процесс моделирования, но я оставил как есть.

**Эксперимент 1: RNN**

**Архитектура сети:**

# define the SimpleRNN model

model = Sequential()

# The output of SimpleRNN will be a 2D tensor of shape (batch\_size, 128)

model.add(SimpleRNN(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')

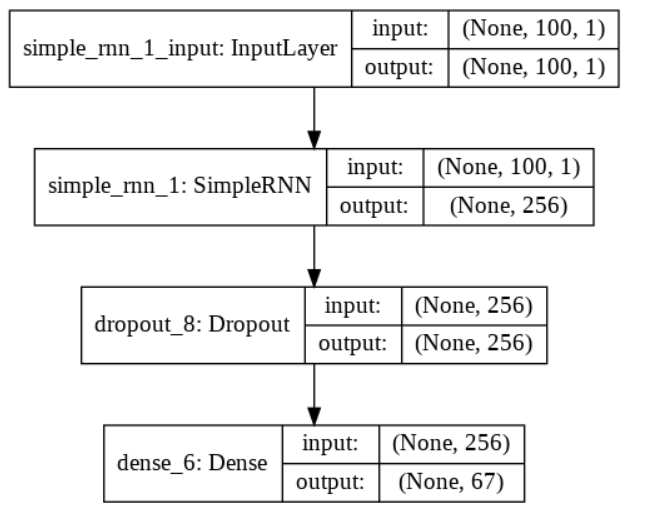


рис. 1

**Результат:**

В качестве результата по всем экспериментам привожу данные последних двух эпох обучения. С результатами других эпох можно ознакомиться в notebook приложенном к данному отчёту.

Epoch 00049: loss improved from 2.38914 to 2.38208, saving model to weights-improvement-49-2.3821-bigger.hdf5

Epoch 50/50 31102/31102 [==============================] - 58s 2ms/step - loss: 2.3709

Epoch 00050: loss improved from 2.38208 to 2.37094, saving model to weights-improvement-50-2.3709-bigger.hdf5

Результат генерации « e along these lines. You will probably get a lot of ideas when traveling, but here are a couple that»

**Вывод по данному эксперименту:**

В эксперименте мы достигли значения функции потерь 2.3709 за 50 эпох обучения.

**Эксперимент 2: Однослойная LSTM(**long short-term memory**)**

**Архитектура сети:**

# define the LSTM model

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')

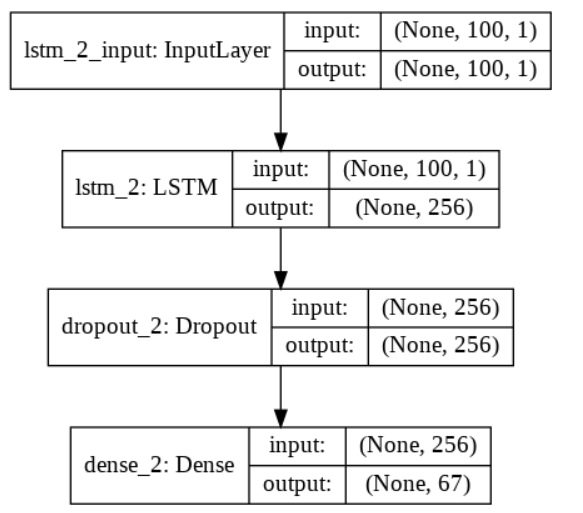


рис. 2

**Результат:**

Epoch 00049: loss improved from 1.86823 to 1.84832, saving model to weights-improvement-49-1.8483.hdf5

Epoch 50/50 31102/31102 [==============================] - 37s 1ms/step - loss: 1.8228

Epoch 00050: loss improved from 1.84832 to 1.82278, saving model to weights-improvement-50-1.8228.hdf5

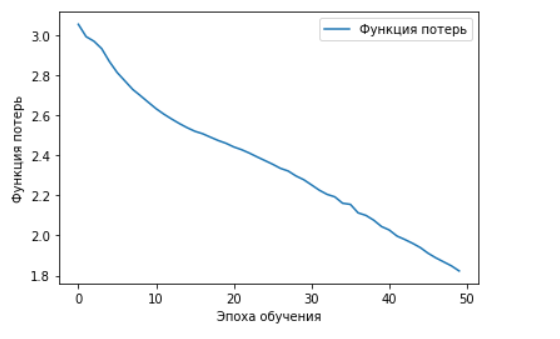


рис. 3

Результат генерации последовательности: «w. ben hunt milwaukee public museum pop».

**Вывод по данному эксперименту**

В данном эксперименте мы достигли значения функции потерь 1.82278, за 50 эпох обучения.

**Эксперимент 3: Двухслойная LSTM**

**Архитектура сети:**

model = Sequential()

model.add(LSTM(256, input\_shape=(X.shape[1], X.shape[2]), return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(256))

model.add(Dropout(0.2))

model.add(Dense(y.shape[1], activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam')

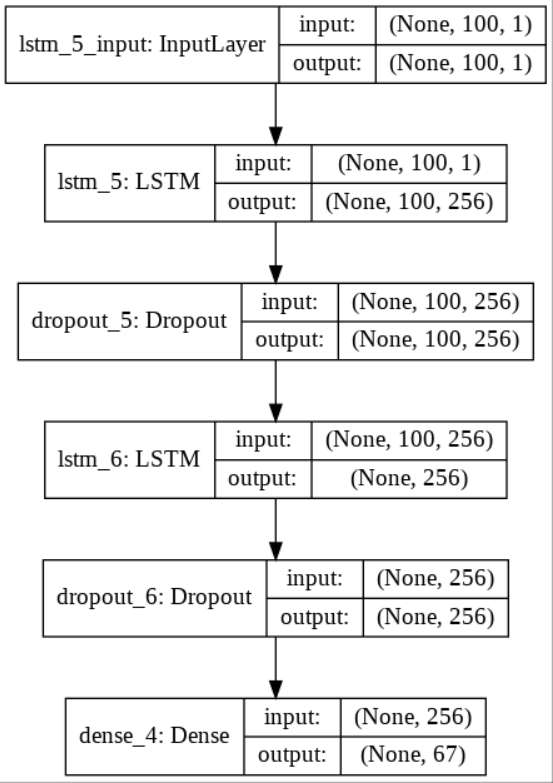


рис. 4

**Результат:**

Epoch 00049: loss improved from 0.94066 to 0.91674, saving model to weights-improvement-49-0.9167-bigger.hdf5

Epoch 50/50

31102/31102 [==============================] - 231s 7ms/step - loss: 0.8987

Epoch 00050: loss improved from 0.91674 to 0.89866, saving model to weights-improvement-50-0.8987-bigger.hdf5

Результат генерации последовательности: « n it was

discovered that tourists prized them. and like everything else, someone

saw a chance to ear».

**Вывод по данному эксперименту:**

В данном эксперименте мы достигли значения функции потерь 0.89866, что существенно лучше результатов предыдущих экспериментов.

**Эксперимент 4: Однослойный GRU(gated recurrent units)**

**Архитектура сети:**

# define the GRU model

model = Sequential()

model.add(GRU(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')

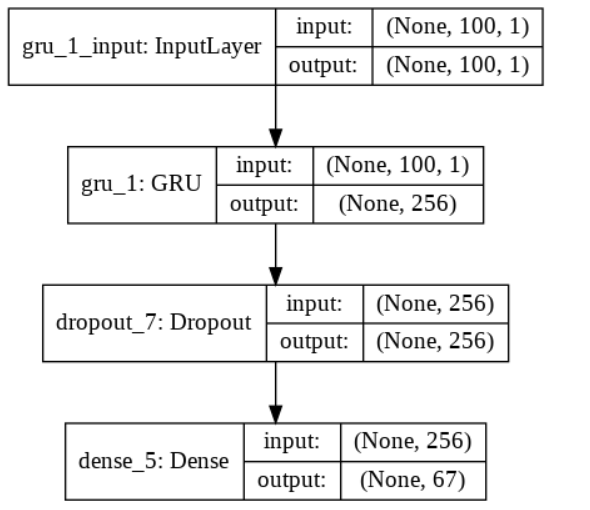


рис. 4

**Результат:**

Epoch 00049: loss improved from 1.03352 to 1.00786, saving model to weights-improvement-49-1.0079-bigger.hdf5

Epoch 50/50

31102/31102 [==============================] - 152s 5ms/step - loss: 0.9810

Epoch 00050: loss improved from 1.00786 to 0.98096, saving model to weights-improvement-50-0.9810-bigger.hdf5

Результат генерации последовательности: «cements dry

quickly, it is better to use a glue that water color will adhere to.

regular hide glue o ».

**Вывод по данному эксперименту:**

В данном эксперименте мы достигли значения функции потерь 0.98096, немного хуже чем в предыдущем эксперименте, но лучше чем в эксперименте с однослойной LSTM и простой RNN.

**Выводы**

Наиболее эффективно с задачей генерации текста справилась двухслойная сеть LSTM. Изучены возможности создания архитектур различных рекуррентных нейронных сетей пригодных для генерации текста, создание моделей а также методы их обучения и методы отображения результатов.

**Источники**

* <https://github.com/Arfua-zz/Text-generation-with-GRU/blob/master/generator.py>
* <https://www.gutenberg.org/files/62286/62286-0.txt>
* <https://towardsdatascience.com/text-generation-using-rnns-fdb03a010b9f>
* <https://www.tensorflow.org/tutorials/text/text_generation>
* <https://habr.com/ru/company/ods/blog/325432/>
* <https://buomsoo-kim.github.io/colab/2018/04/16/Importing-files-from-Google-Drive-in-Google-Colab.md/>