In [2]:
```
import numpy as np
from keras.layers import Dense, Activation
from keras.layers.recurrent import SimpleRNN, LSTM, GRU
from keras.models import Sequential


# построчное чтение из примера с текстом
with open("alice_in_wonderland.txt", 'rb') as _in:
    lines = []
    for line in _in:
        line = line.strip().lower().decode("ascii", "ignore")
        if len(line) == 0:
            continue
        lines.append(line)
text = " ".join(lines)
chars = set([c for c in text])
nb_chars = len(chars)


# создание индекса символов и reverse mapping чтобы передвигаться между знач
ениями numerical
# ID and a specific character. The numerical ID will correspond to a column
# ID и определенный символ. Numerical ID будет соответсвовать колонке
# число при использовании one-hot кодировки для представление входов символо
в
char2index = {c: i for i, c in enumerate(chars)}
index2char = {i: c for i, c in enumerate(chars)}

# для удобства выберете фиксированную длину последовательность 10 символов
SEQLEN, STEP = 10, 1
input_chars, label_chars = [], []

# конвертация data в серии разных SEQLEN-length субпоследовательностей
for i in range(0, len(text) - SEQLEN, STEP):
    input_chars.append(text[i: i + SEQLEN])
    label_chars.append(text[i + SEQLEN])


# Вычисление one-hot encoding входных последовательностей X и следующего сим
вола (the label) y

X = np.zeros((len(input_chars), SEQLEN, nb_chars), dtype=np.bool)
y = np.zeros((len(input_chars), nb_chars), dtype=np.bool)
for i, input_char in enumerate(input_chars):
    for j, ch in enumerate(input_char):
        X[i, j, char2index[ch]] = 1
    y[i, char2index[label_chars[i]]] = 1


# установка ряда метапамертров  для нейронной сети и процесса тренировки
BATCH_SIZE, HIDDEN_SIZE = 128, 128
NUM_ITERATIONS = 700 # 25 должно быть достаточно
NUM_EPOCHS_PER_ITERATION = 1
NUM_PREDS_PER_EPOCH = 100


# Create a super simple recurrent neural network. There is one recurrent
# layer that produces an embedding of size HIDDEN_SIZE from the one-hot
# encoded input layer. This is followed by a Dense fully-connected layer
# across the set of possible next characters, which is converted to a
# probability score via a standard softmax activation with a multi-class
# cross-entropy loss function linking the prediction to the one-hot
# encoding character label.

'''
Создание очень простой рекуррентной нейронной сети. В ней будет один реккуре
```

```
==================================================
Итерация #: 0
Epoch 1/1
158773/158773 [==============================] - 37s 236us/step - loss: 2.313
1
Генерация из посева: g those be
g those beat the said the said the said the said the said the said the said t
he said the said the said the sai==========================================
======
Итерация #: 1
Epoch 1/1
158773/158773 [==============================] - 37s 233us/step - loss: 1.913
3
Генерация из посева: in complia
in compliag to her and the gront the hat has the hast the hat has the hast th
e hat has the hast the hat has th==========================================
======
Итерация #: 2
Epoch 1/1
158773/158773 [==============================] - 52s 328us/step - loss: 1.760
6
Генерация из посева: h in parag
h in parage on the mouse to the gryphon a little the mouse to the gryphon a l
ittle the mouse to the gryphon a ==========================================
======
Итерация #: 3
Epoch 1/1
158773/158773 [==============================] - 63s 398us/step - loss: 1.659
6
Генерация из посева: en i get i
en i get in the morked the morked the morked the morked the morked the morked
the morked the morked the morked===========================================
=====
Итерация #: 4
Epoch 1/1
158773/158773 [==============================] - 61s 386us/step - loss: 1.584
5
Генерация из посева: s room for
s room for the project gutenberg to see the project gutenberg to see the proj
ect gutenberg to see the project ==========================================
======
Итерация #: 5
Epoch 1/1
158773/158773 [==============================] - 61s 384us/step - loss: 1.522
8
Генерация из посева: uppress hi
uppress his the way of the works and the way of the works and the way of the
works and the way of the works an==========================================
======
Итерация #: 6
Epoch 1/1
158773/158773 [==============================] - 62s 391us/step - loss: 1.470
5
Генерация из посева: such a tri
such a tried to the mouse the mock turtle sook the mock turtle sook the mock
turtle sook the mock turtle sook ==========================================
======
Итерация #: 7
Epoch 1/1
158773/158773 [==============================] - 62s 391us/step - loss: 1.427
1
Генерация из посева: rtunity fo
rtunity for the hatter without was i to herself the hatter without was i to h
erself the hatter without was i t==========================================
======
Итерация #: 8
Epoch 1/1
158773/158773 [==============================] - 55s 349us/step - loss: 1.389
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-2-eab24678006f> in <module>
     84     print("=" * 50)
     85     print("Итерация #: %d" % (iteration))
---> 86     model.fit(X, y, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS_PER_ITER
ATION)
     87
     88     # Select a random example input sequence.

~/anaconda3/lib/python3.7/site-packages/keras/engine/training.py in fit(self,
x, y, batch_size, epochs, verbose, callbacks, validation_split, validation_da
ta, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, val
idation_steps, validation_freq, max_queue_size, workers, use_multiprocessing,
**kwargs)
   1237                                         steps_per_epoch=steps_per_epo
ch,
   1238                                         validation_steps=validation_s
teps,
-> 1239                                         validation_freq=validation_fr
eq)
   1240
   1241     def evaluate(self,

~/anaconda3/lib/python3.7/site-packages/keras/engine/training_arrays.py in fi
t_loop(model, fit_function, fit_inputs, out_labels, batch_size, epochs, verbo
se, callbacks, val_function, val_inputs, shuffle, initial_epoch, steps_per_ep
och, validation_steps, validation_freq)
    194                     ins_batch[i] = ins_batch[i].toarray()
    195
--> 196                 outs = fit_function(ins_batch)
    197                 outs = to_list(outs)
    198                 for l, o in zip(out_labels, outs):

~/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/keras/backend.
py in __call__(self, inputs)
   3725         value = math_ops.cast(value, tensor.dtype)
   3726       converted_inputs.append(value)
-> 3727     outputs = self._graph_fn(*converted_inputs)
   3728
   3729     # EagerTensor.numpy() will often make a copy to ensure memory saf
ety.

~/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/eager/functio
n.py in __call__(self, *args, **kwargs)
   1549       TypeError: For invalid positional/keyword argument combination
s.
   1550     """
-> 1551     return self._call_impl(args, kwargs)
   1552
   1553   def _call_impl(self, args, kwargs, cancellation_manager=None):

~/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/eager/functio
n.py in _call_impl(self, args, kwargs, cancellation_manager)
   1589       raise TypeError("Keyword arguments {} unknown. Expected {}.".fo
rmat(
   1590           list(kwargs.keys()), list(self._arg_keywords)))
-> 1591     return self._call_flat(args, self.captured_inputs, cancellation_m
anager)
   1592
   1593   def _filtered_call(self, args, kwargs):

~/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/eager/functio
n.py in _call_flat(self, args, captured_inputs, cancellation_manager)
   1690       # No tape is watching; skip to running the function.
   1691       return self._build_call_outputs(self._inference_function.call(
-> 1692           ctx, args, cancellation_manager=cancellation_manager))
   1693     forward_backward = self._select_forward_and_backward_functions(
```

In [ ]: