```
In [2]: from __future__ import print_function

        from keras.preprocessing import sequence
        from keras.models import Sequential
        from keras.layers import Dense, Embedding
        from keras.layers import LSTM
        from keras.datasets import imdb

        max_features = 20000

        # обрезание текстов после данного количества слов (среди top max_features на
        иболее используемые слова)
        maxlen = 80
        batch_size = 50 # увеличьте значение для ускорения обучения

        print('Загрузка данных...')
        (x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_feature
        s)
        print(len(x_train), 'тренировочные последовательности')
        print(len(x_test), 'тестовые последовательности')

        print('Pad последовательности (примеров в x единицу времени)')
        x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
        x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
        print('x_train shape:', x_train.shape)
        print('x_test shape:', x_test.shape)

        print('Построение модели...')
        model = Sequential()
        model.add(Embedding(max_features, 128))
        model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
        model.add(Dense(1, activation='sigmoid'))

        # стоит попробовать использовать другие оптимайзер и другие конфигурации опт
        имайзеров
        model.compile(loss='binary_crossentropy',
                      optimizer='adam',
                      metrics=['accuracy'])

        print('Процесс обучения...')
        model.fit(x_train, y_train,
                  batch_size=batch_size,
                  epochs=50, # увеличьте при необходимости
                  validation_data=(x_test, y_test))
        score, acc = model.evaluate(x_test, y_test,
                                    batch_size=batch_size)
        print('Результат при тестировании:', score)
        print('Тестовая точность:', acc)
```

```
Загрузка данных...
25000 тренировочные последовательности
25000 тестовые последовательности
Pad последовательности (примеров в x единицу времени)
x_train shape: (25000, 80)
x_test shape: (25000, 80)
Построение модели...
Процесс обучения...
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/fram
ework/indexed_slices.py:433: UserWarning: Converting sparse IndexedSlices to
a dense Tensor of unknown shape. This may consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/50
25000/25000 [==============================] - 189s 8ms/step - loss: 0.4576 -
accuracy: 0.7828 - val_loss: 0.3798 - val_accuracy: 0.8365
Epoch 2/50
25000/25000 [==============================] - 187s 7ms/step - loss: 0.3020 -
accuracy: 0.8774 - val_loss: 0.4354 - val_accuracy: 0.7941
Epoch 3/50
25000/25000 [==============================] - 181s 7ms/step - loss: 0.2302 -
accuracy: 0.9108 - val_loss: 0.4155 - val_accuracy: 0.8324
Epoch 4/50
25000/25000 [==============================] - 182s 7ms/step - loss: 0.1691 -
accuracy: 0.9353 - val_loss: 0.4609 - val_accuracy: 0.8157
Epoch 5/50
25000/25000 [==============================] - 189s 8ms/step - loss: 0.1250 -
accuracy: 0.9530 - val_loss: 0.5141 - val_accuracy: 0.8217
Epoch 6/50
25000/25000 [==============================] - 186s 7ms/step - loss: 0.0922 -
accuracy: 0.9670 - val_loss: 0.5965 - val_accuracy: 0.8120
Epoch 7/50
25000/25000 [==============================] - 185s 7ms/step - loss: 0.0735 -
accuracy: 0.9746 - val_loss: 0.7028 - val_accuracy: 0.8164
Epoch 8/50
25000/25000 [==============================] - 176s 7ms/step - loss: 0.0605 -
accuracy: 0.9789 - val_loss: 0.7117 - val_accuracy: 0.7994
Epoch 9/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0533 -
accuracy: 0.9821 - val_loss: 0.8305 - val_accuracy: 0.8116
Epoch 10/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0346 -
accuracy: 0.9888 - val_loss: 0.8718 - val_accuracy: 0.8148
Epoch 11/50
25000/25000 [==============================] - 173s 7ms/step - loss: 0.0321 -
accuracy: 0.9895 - val_loss: 0.8627 - val_accuracy: 0.8024
Epoch 12/50
25000/25000 [==============================] - 177s 7ms/step - loss: 0.0243 -
accuracy: 0.9922 - val_loss: 0.8783 - val_accuracy: 0.8056
Epoch 13/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0177 -
accuracy: 0.9946 - val_loss: 0.9066 - val_accuracy: 0.8039
Epoch 14/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0180 -
accuracy: 0.9944 - val_loss: 1.0196 - val_accuracy: 0.8051
Epoch 15/50
25000/25000 [==============================] - 175s 7ms/step - loss: 0.0149 -
accuracy: 0.9952 - val_loss: 1.0589 - val_accuracy: 0.8045
Epoch 16/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0133 -
accuracy: 0.9959 - val_loss: 1.1555 - val_accuracy: 0.8076
Epoch 17/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0141 -
accuracy: 0.9958 - val_loss: 1.2122 - val_accuracy: 0.8084
Epoch 18/50
25000/25000 [==============================] - 174s 7ms/step - loss: 0.0127 -
accuracy: 0.9958 - val_loss: 1.2041 - val_accuracy: 0.8078
Epoch 19/50
25000/25000 [==============================] - 176s 7ms/step - loss: 0.0061 -
accuracy: 0.9981 - val_loss: 1.2940 - val_accuracy: 0.8072
Epoch 20/50
25000/25000 [==============================] - 176s 7ms/step - loss: 0.0088 -
accuracy: 0.9974 - val_loss: 1.1823 - val_accuracy: 0.8046
Epoch 21/50
25000/25000 [==============================] - 176s 7ms/step - loss: 0.0072 -
accuracy: 0.9980 - val_loss: 1.2773 - val_accuracy: 0.8045
Epoch 22/50
25000/25000 [==============================] - 180s 7ms/step - loss: 0.0034 -
accuracy: 0.9990 - val_loss: 1.3458 - val_accuracy: 0.8089
Epoch 23/50
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-2-7036a194faf3> in <module>
     39           batch_size=batch_size,
     40           epochs=50, # увеличьте при необходимости
---> 41           validation_data=(x_test, y_test))
     42 score, acc = model.evaluate(x_test, y_test,
     43                             batch_size=batch_size)

~/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(
   1694         args,
In [ ]:
```