In [ ]:
```
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 = 128 # 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(64)) # прогон 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 [==============================] - 92s 4ms/step - loss: 0.4663 -
accuracy: 0.7741 - val_loss: 0.3724 - val_accuracy: 0.8378
Epoch 2/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.3016 -
accuracy: 0.8779 - val_loss: 0.3827 - val_accuracy: 0.8317
Epoch 3/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.2207 -
accuracy: 0.9166 - val_loss: 0.4526 - val_accuracy: 0.8130
Epoch 4/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.1631 -
accuracy: 0.9396 - val_loss: 0.5327 - val_accuracy: 0.8192
Epoch 5/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.1202 -
accuracy: 0.9567 - val_loss: 0.5669 - val_accuracy: 0.8128
Epoch 6/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.0949 -
accuracy: 0.9660 - val_loss: 0.6358 - val_accuracy: 0.8044
Epoch 7/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.0750 -
accuracy: 0.9737 - val_loss: 0.7960 - val_accuracy: 0.8085
Epoch 8/50
25000/25000 [==============================] - 87s 3ms/step - loss: 0.0577 -
accuracy: 0.9800 - val_loss: 0.7951 - val_accuracy: 0.7958
Epoch 9/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.0474 -
accuracy: 0.9833 - val_loss: 0.9094 - val_accuracy: 0.8051
Epoch 10/50
25000/25000 [==============================] - 87s 3ms/step - loss: 0.0429 -
accuracy: 0.9860 - val_loss: 0.8872 - val_accuracy: 0.8021
Epoch 11/50
25000/25000 [==============================] - 88s 4ms/step - loss: 0.0427 -
accuracy: 0.9853 - val_loss: 0.9698 - val_accuracy: 0.8026
Epoch 12/50
25000/25000 [==============================] - 89s 4ms/step - loss: 0.0331 -
accuracy: 0.9887 - val_loss: 1.0546 - val_accuracy: 0.8024
Epoch 13/50
25000/25000 [==============================] - 92s 4ms/step - loss: 0.0259 -
accuracy: 0.9912 - val_loss: 1.1057 - val_accuracy: 0.7977
Epoch 14/50
25000/25000 [==============================] - 92s 4ms/step - loss: 0.0219 -
accuracy: 0.9922 - val_loss: 1.1024 - val_accuracy: 0.7988
Epoch 15/50
25000/25000 [==============================] - 95s 4ms/step - loss: 0.0221 -
accuracy: 0.9930 - val_loss: 1.2072 - val_accuracy: 0.7980
Epoch 16/50
25000/25000 [==============================] - 90s 4ms/step - loss: 0.0265 -
accuracy: 0.9916 - val_loss: 1.0457 - val_accuracy: 0.7757
Epoch 17/50
25000/25000 [==============================] - 87s 3ms/step - loss: 0.0225 -
accuracy: 0.9928 - val_loss: 1.1661 - val_accuracy: 0.7974
Epoch 18/50
25000/25000 [==============================] - 89s 4ms/step - loss: 0.0166 -
accuracy: 0.9947 - val_loss: 1.2548 - val_accuracy: 0.7999
Epoch 19/50
25000/25000 [==============================] - 87s 3ms/step - loss: 0.0165 -
accuracy: 0.9939 - val_loss: 1.4214 - val_accuracy: 0.7969
Epoch 20/50
25000/25000 [==============================] - 87s 3ms/step - loss: 0.0184 -
accuracy: 0.9940 - val_loss: 1.3395 - val_accuracy: 0.7994
Epoch 21/50
25000/25000 [==============================] - 92s 4ms/step - loss: 0.0149 -
accuracy: 0.9946 - val_loss: 1.4685 - val_accuracy: 0.7815
Epoch 22/50
25000/25000 [==============================] - 90s 4ms/step - loss: 0.0144 -
accuracy: 0.9951 - val_loss: 1.4102 - val_accuracy: 0.7956
Epoch 23/50
```

In [ ]: