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
from keras.layers import Conv1D
from keras.layers import MaxPooling1D
from keras.layers import Dropout

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(Conv1D(filters=32, kernel_size=3, padding='same', activation='rel
u'))
#model.add(MaxPooling1D(pool_size=2))
# model.add(keras.layers.Dropout(0.3))

model.add(Embedding(max_features, 128))
model.add(LSTM(256, 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='SGD', # optimizer='adam' прогон 4
              metrics=['accuracy']) # 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 [==============================] - 229s 9ms/step - loss: 0.6930 -
accuracy: 0.5161 - val_loss: 0.6929 - val_accuracy: 0.5290
Epoch 2/50
25000/25000 [==============================] - 205s 8ms/step - loss: 0.6929 -
accuracy: 0.5162 - val_loss: 0.6929 - val_accuracy: 0.5341
Epoch 3/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6929 -
accuracy: 0.5148 - val_loss: 0.6929 - val_accuracy: 0.5338
Epoch 4/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6929 -
accuracy: 0.5244 - val_loss: 0.6928 - val_accuracy: 0.5406
Epoch 5/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6928 -
accuracy: 0.5335 - val_loss: 0.6928 - val_accuracy: 0.5411
Epoch 6/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6928 -
accuracy: 0.5229 - val_loss: 0.6928 - val_accuracy: 0.5386
Epoch 7/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6928 -
accuracy: 0.5334 - val_loss: 0.6927 - val_accuracy: 0.5402
Epoch 8/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6927 -
accuracy: 0.5350 - val_loss: 0.6927 - val_accuracy: 0.5466
Epoch 9/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6927 -
accuracy: 0.5346 - val_loss: 0.6927 - val_accuracy: 0.5507
Epoch 10/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6927 -
accuracy: 0.5350 - val_loss: 0.6926 - val_accuracy: 0.5452
Epoch 11/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6926 -
accuracy: 0.5332 - val_loss: 0.6926 - val_accuracy: 0.5478
Epoch 12/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6926 -
accuracy: 0.5411 - val_loss: 0.6925 - val_accuracy: 0.5580
Epoch 13/50
25000/25000 [==============================] - 200s 8ms/step - loss: 0.6925 -
accuracy: 0.5434 - val_loss: 0.6925 - val_accuracy: 0.5592
Epoch 14/50
25000/25000 [==============================] - 203s 8ms/step - loss: 0.6925 -
accuracy: 0.5437 - val_loss: 0.6925 - val_accuracy: 0.5534
Epoch 15/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6924 -
accuracy: 0.5521 - val_loss: 0.6924 - val_accuracy: 0.5558
Epoch 16/50
25000/25000 [==============================] - 202s 8ms/step - loss: 0.6924 -
accuracy: 0.5523 - val_loss: 0.6924 - val_accuracy: 0.5589
Epoch 17/50
25000/25000 [==============================] - 202s 8ms/step - loss: 0.6924 -
accuracy: 0.5518 - val_loss: 0.6923 - val_accuracy: 0.5606
Epoch 18/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6923 -
accuracy: 0.5574 - val_loss: 0.6923 - val_accuracy: 0.5612
Epoch 19/50
25000/25000 [==============================] - 204s 8ms/step - loss: 0.6922 -
accuracy: 0.5575 - val_loss: 0.6922 - val_accuracy: 0.5492
Epoch 20/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6922 -
accuracy: 0.5566 - val_loss: 0.6922 - val_accuracy: 0.5576
Epoch 21/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6922 -
accuracy: 0.5440 - val_loss: 0.6921 - val_accuracy: 0.5727
Epoch 22/50
25000/25000 [==============================] - 201s 8ms/step - loss: 0.6921 -
accuracy: 0.5637 - val_loss: 0.6921 - val_accuracy: 0.5591
Epoch 23/50
```

In [ ]: