**Жуковский Павел, 3 курс, 12 группа, кафедра КТС**

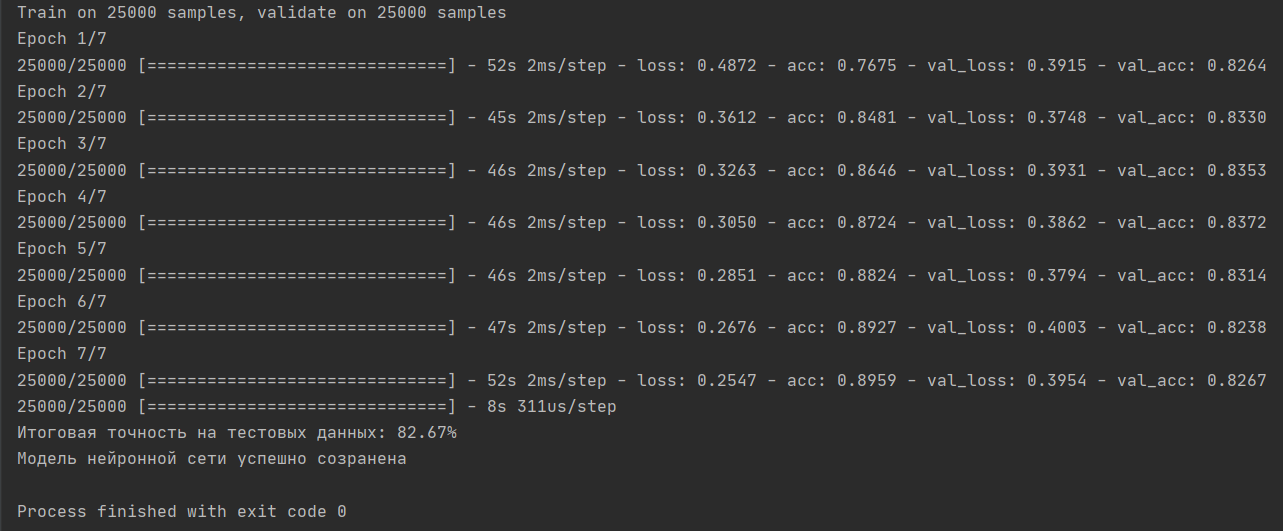
**Задание 1.**

Обучите рекуррентную нейронную сеть распознаванию тональности отзывов на тестовых данных открытого набора данных imdb.

Код для обучения сети (task\_1\_network\_training.py):

import numpy as np  
from keras.layers import Dense, Embedding, LSTM  
from keras.models import Sequential  
from keras.preprocessing import sequence  
from keras.datasets import imdb  
  
# Устанавливаем seed для повторяемости результатов  
np.random.seed(42)  
  
# Максимальное количество слов (по частоте использования)  
max\_features = 5000  
  
# save np.load  
np\_load\_old = np.load  
  
# modify the default parameters of np.load  
np.load = lambda \*a, \*\*k: np\_load\_old(\*a, \*\*k, allow\_pickle=True)  
  
# Загружаем данные  
(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(nb\_words=max\_features)  
  
# restore np.load for future normal usage  
np.load = np\_load\_old  
  
# Максимальная длина рецензии в словах  
maxlen = 80  
  
# Заполняем или обрезаем рецензии  
X\_train = sequence.pad\_sequences(X\_train, maxlen=maxlen)  
X\_test = sequence.pad\_sequences(X\_test, maxlen=maxlen)  
  
# Создаем сеть  
model = Sequential()  
  
# Слой для векторного представления слов  
model.add(Embedding(max\_features, 32, dropout=0.2))  
  
# Слой долго-краткосрочной памяти  
model.add(LSTM(100, dropout\_W=0.2, dropout\_U=0.2))  
  
# Полносвязный слой для классификации  
model.add(Dense(1, activation="sigmoid"))  
  
# Компилируем модель  
model.compile(loss='binary\_crossentropy',  
 optimizer='adam',  
 metrics=['accuracy'])  
  
# Обучаем сеть  
model.fit(X\_train, y\_train,  
 batch\_size=64,  
 nb\_epoch=7,  
 validation\_data=(X\_test, y\_test),  
 verbose=1)  
  
# Проверяем качество обучения на тестовых данных  
scores = model.evaluate(X\_test, y\_test, batch\_size=64)  
print("Итоговая точность на тестовых данных: %.2f%%" % (scores[1]\*100))  
  
# Генерируем описание модели в формате json  
model\_json = model.to\_json()  
  
# Записываем модель в файл  
json\_file = open("rnn\_model\_test.json", "w")  
json\_file.write(model\_json)  
json\_file.close()  
  
model.save\_weights("rnn\_model\_test.h5")  
  
print("Модель нейронной сети успешно сохранена!")

Вывод:



Таким образом, мы обучили нашу сеть и сохранили её в json-файле.

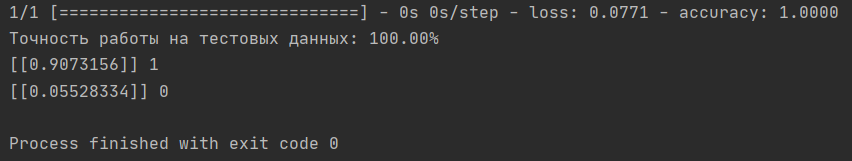
**Задание 2**

Теперь я использовал данную сеть для распознавания тональности отзывов на тестовых данных открытого набора данных imdb.

Код для использования обученной сети (task\_2\_recognition.py):

import numpy as np  
from tensorflow.keras.models import model\_from\_json  
from keras.preprocessing.sequence import pad\_sequences  
from keras.preprocessing.text import hashing\_trick  
  
max\_words = 10000  
  
# np\_load\_old = np.load  
# np.load = lambda \*a,\*\*k: np\_load\_old(\*a, allow\_pickle=True, \*\*k)  
  
# (x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=max\_words)  
  
# np.load = np\_load\_old  
  
maxlen = 200  
  
# x\_train = pad\_sequences(x\_train, maxlen=maxlen)  
# x\_test = pad\_sequences(x\_test, maxlen=maxlen)  
  
json\_file = open("rnn\_model.json", "r")  
loaded\_model\_json = json\_file.read()  
json\_file.close()  
loaded\_model = model\_from\_json(loaded\_model\_json)  
loaded\_model.load\_weights("rnn\_model.h5")  
  
loaded\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
  
  
file1=open("review1.txt")  
review1=file1.read()  
x1=hashing\_trick(review1, maxlen)  
  
file2=open("review2.txt")  
review2=file2.read()  
x2=hashing\_trick(review2, maxlen)  
  
arr=np.array([x1, x2])  
arr = pad\_sequences(arr, maxlen=maxlen)  
y=np.array([1, 0])  
  
scores = loaded\_model.evaluate(arr, y, verbose=1)  
print("Точность работы на тестовых данных: %.2f%%" % (scores[1]\*100))  
  
print(loaded\_model.predict(arr[:1]), y[0])  
print(loaded\_model.predict(arr[1:]), y[1])

Вывод:

Рекуррентные нейронные сети отлично справились с задачей.