МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: Прогноз успеха фильмов по обзорам

Студент гр. 7383 _____ Александров Р.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы.

Прогноз успеха фильмов по обзорам (Predict Sentiment From Movie Reviews).

Постановка задачи.

1. Ознакомиться с задачей регрессии
2. Изучить способы представления текста для передачи в ИНС
3. Достигнуть точность прогноза не менее 95%

Требования.

1. Построить и обучить нейронную сеть для обработки текста
2. Исследовать результаты при различном размере вектора представления текста
3. Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте)

Выполнение работы.

В ходе работы была создана и обучена модель нейронной сети, весь код представлен в приложении А. Первоначальные параметры: количество эпох равно 2, размер батча равен 500. Архитектура сети состоит из 6 слоев: 4 dense-слоя и 2 droupout-слоя.

Исследования проводились на следующих размерах вектора представления текста (*L)*: 2000, 4000, 6000, 8000 и 10000.

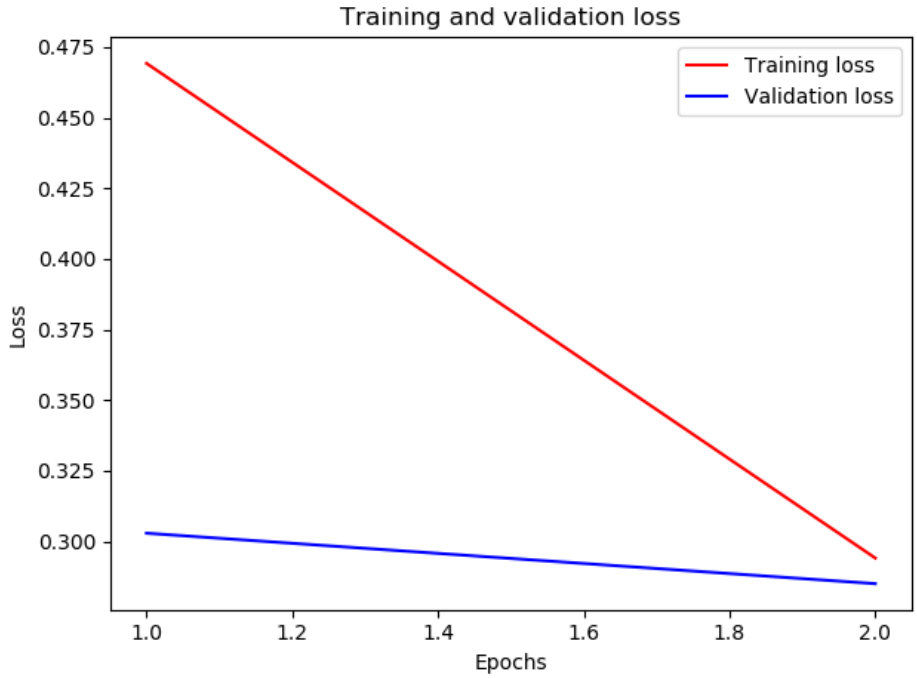

Рисунок 1 – Ошибки при L = 2000

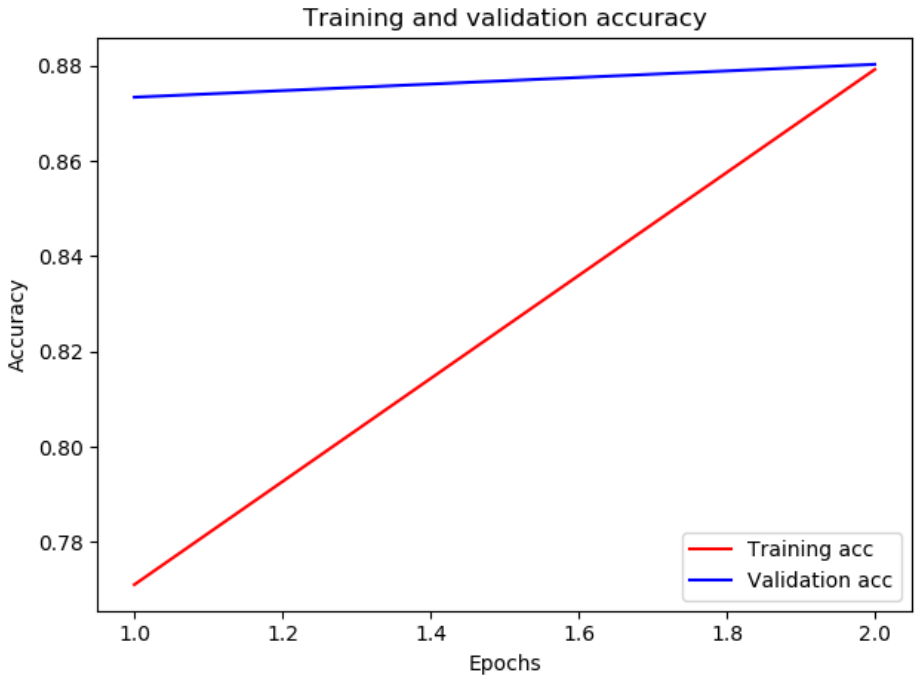

Рисунок 2 – Точность при L = 2000

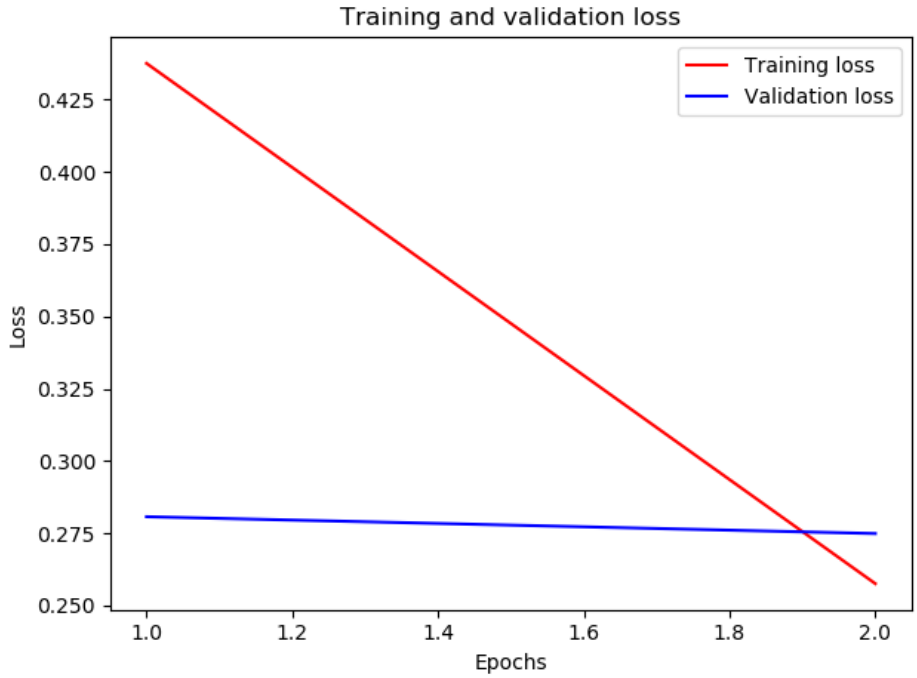

Рисунок 3 – Ошибки при L = 4000

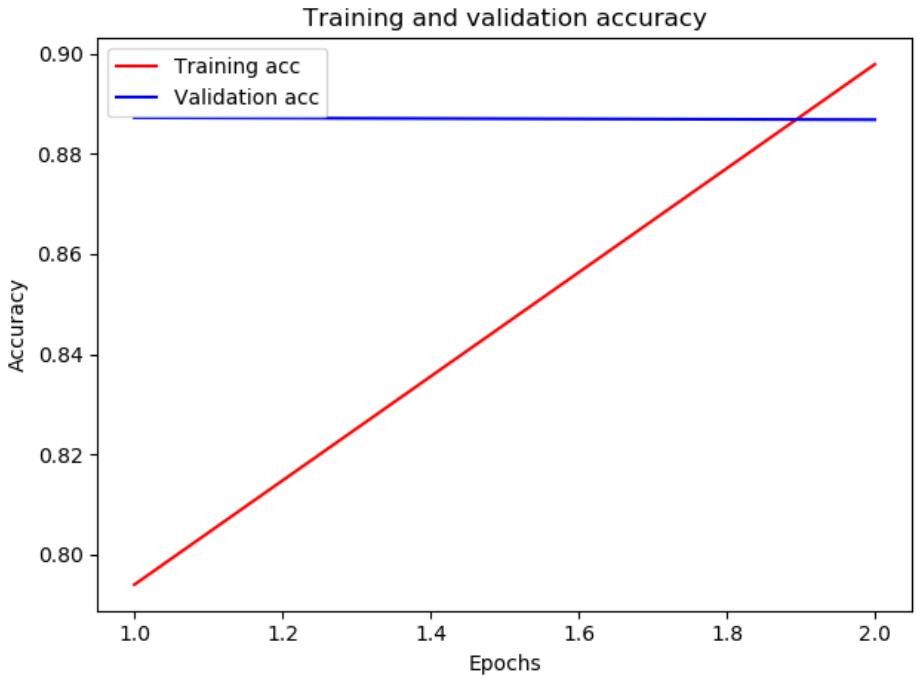

Рисунок 4 – Точность при L = 4000

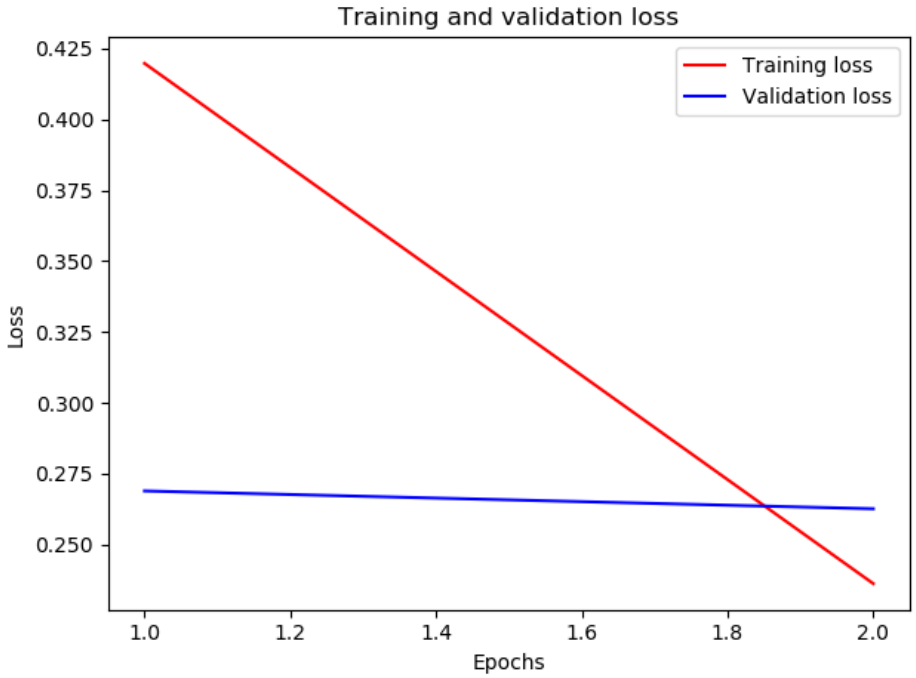

Рисунок 5 – Ошибки при L = 6000

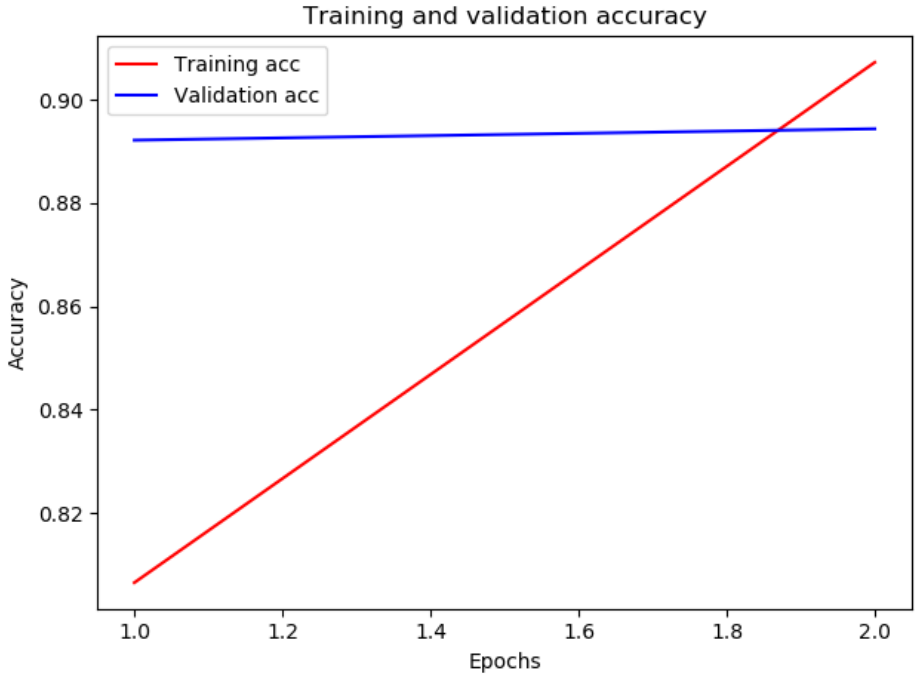

Рисунок 6 – Точность при L = 6000

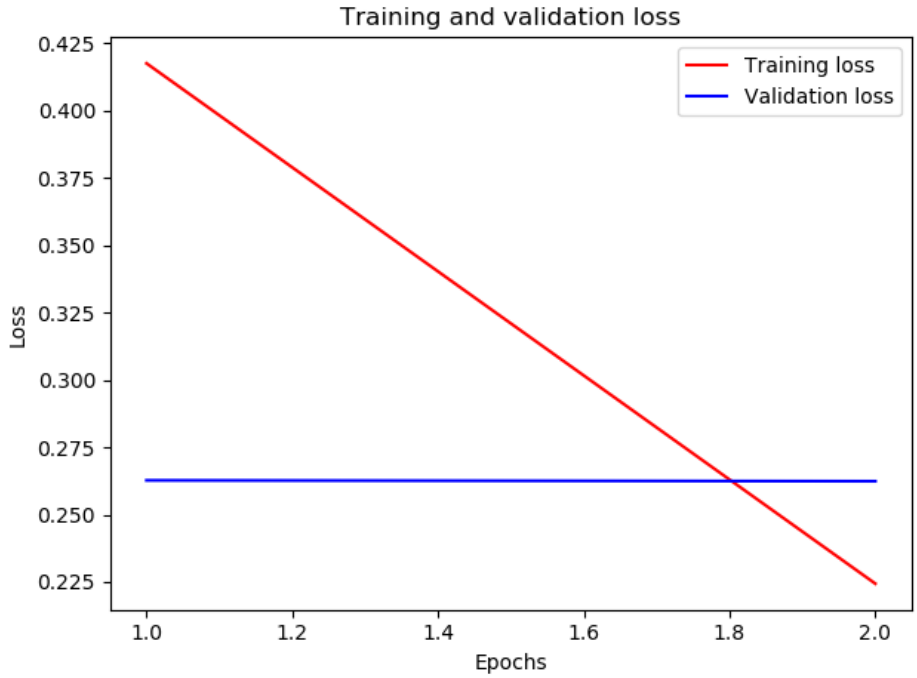

Рисунок 7– Ошибки при L = 8000

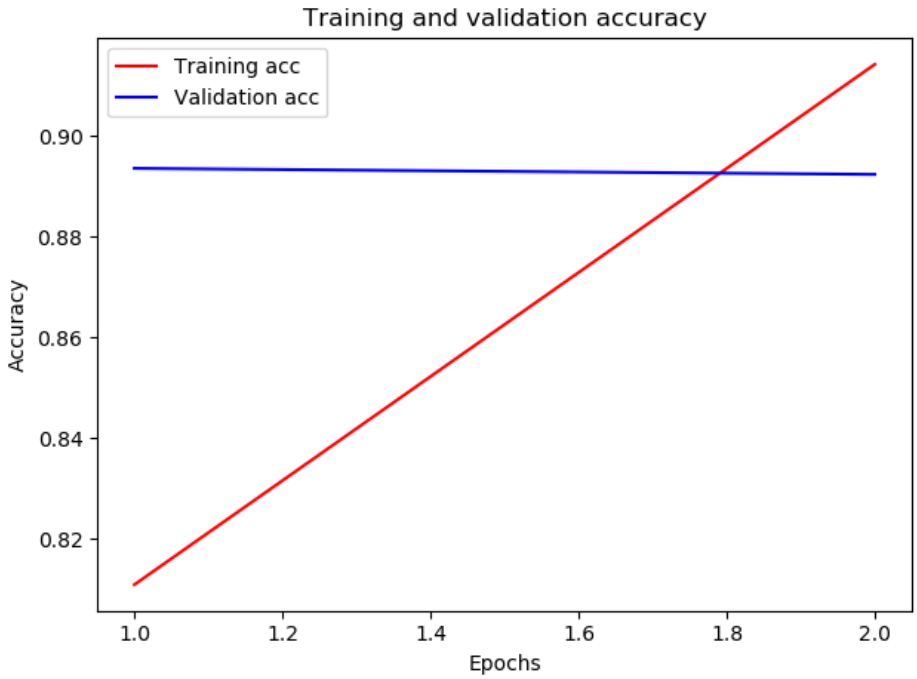

Рисунок 8 – Точность при L = 8000

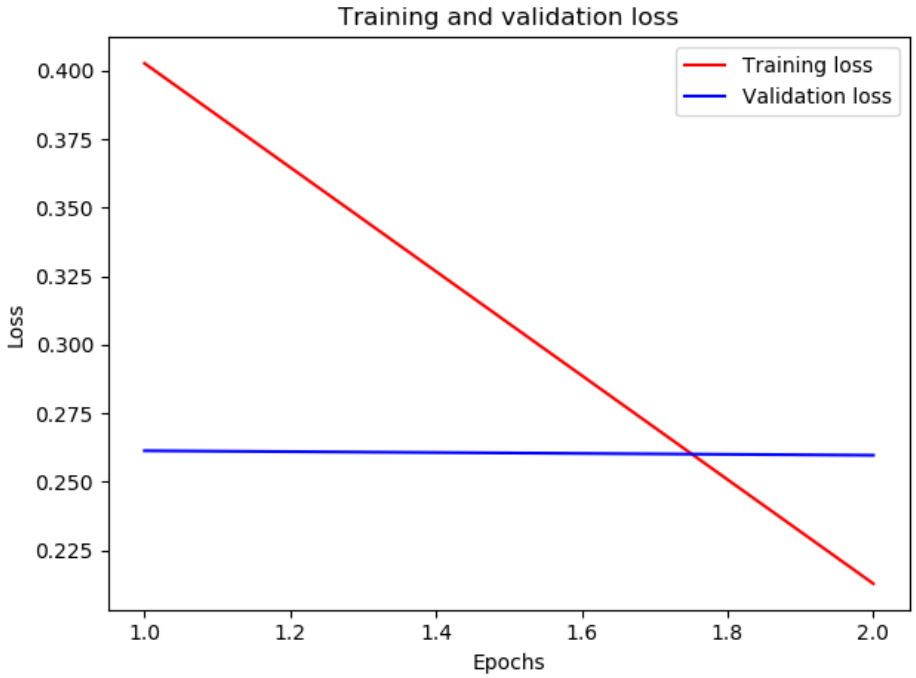

Рисунок 9 – Ошибки при L = 10000

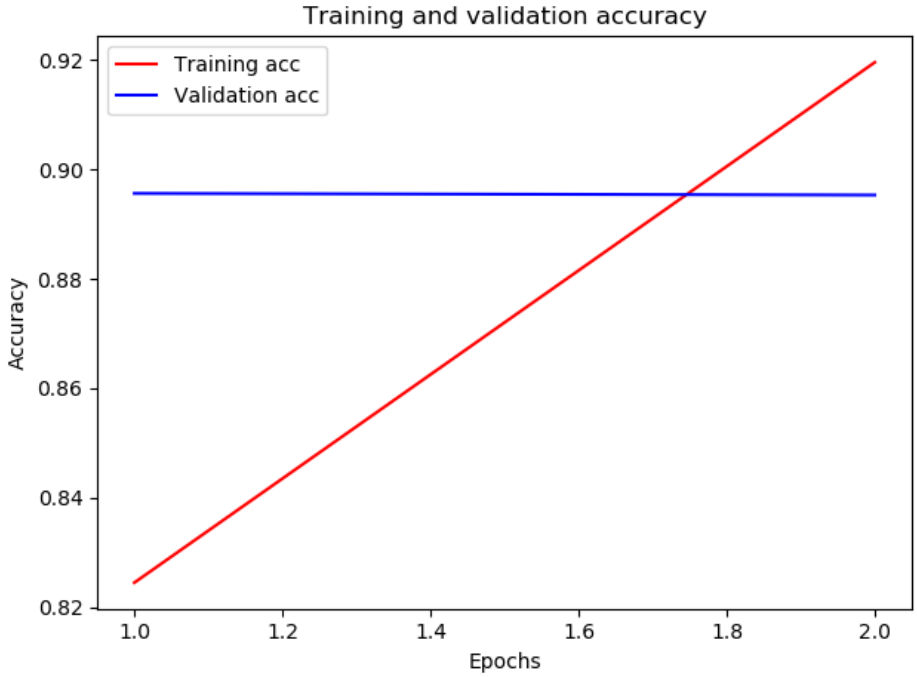

Рисунок 10 – Точность при L = 10000

Из рис. 1-10 видим, что при увеличении размера вектора представления текста потери уменьшаются, а точность возрастает.

Теперь реализуем функцию, которая позволяет ввести пользовательский текст, она представлена в приложении Б.

Проверим нейронную сеть на двух отзывах:

- “The film has a bad actor playing and the worst plot”;
- “Best of the best”.

Результат представлен на рис. 11.

```
Review "The film has a bad actor playing and the worst plot"
 has a 0.37988815 accuracy to be positive
Review "Best of the best"
 has a 0.61099637 accuracy to be positive
```

Рисунок 11 – Результат работы нейронной сети на пользовательских отзывах

Выводы.

В ходе работы были освоена передача пользовательского текста в нейронную сеть, изучена задача регрессии, исследованы результаты при различном размере вектора представления текста. Было установлено, что увеличении размера вектора представления текста потери уменьшаются, а точность возрастает.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import matplotlib.pyplot as plt
import numpy as np
from keras import models, Sequential
from keras import layers

from keras.datasets import imdb

dimension = 10000

(training_data, training_targets), (testing_data, testing_targets)
= imdb.load_data(num_words=dimension)

data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets),
axis=0)

def vectorize(sequences, dim=dimension):
    results = np.zeros((len(sequences), dim))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1

    return results

data = vectorize(data)
targets = np.array(targets).astype("float32")

test_x = data[:10000]
test_y = targets[:10000]

train_x = data[10000:]
train_y = targets[10000:]

def build_model():
    model = Sequential()
    # Input - Layer
    model.add(layers.Dense(50, activation="relu",
input_shape=(dimension,)))
    # Hidden - Layers
    model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation="relu"))
    model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation="relu"))
    # Output- Layer
```

```
    model.add(layers.Dense(1, activation="sigmoid"))
    model.summary()

    return model

model = build_model()
model.compile(optimizer = "adam", loss = "binary_crossentropy",
metrics = ["accuracy"])

results = model.fit(train_x, train_y, epochs= 2, batch_size = 500,
validation_data = (test_x, test_y))
print(np.mean(results.history["val_accuracy"]))

history_dict = results.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']
epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'r', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ФУНКЦИИ ДЛЯ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА

```
def input_user_review():
    model = build_model()
    model.compile(optimizer="adam",     loss="binary_crossentropy",
metrics=["accuracy"])
    model.fit(train_x,    train_y,    epochs=2,    batch_size=500,
validation_data=(test_x, test_y))

    text = input('Enter user text')
    reviews = [text]
    word_index = imdb.get_word_index()
    number_reviews = []
    for review in reviews:
        single_number_review = []
        for w in review:
            if w in word_index and word_index[w] < dimension:
                single_number_review.append(word_index[w])
        number_reviews.append(single_number_review)
    for i in range(len(number_reviews)):
        vectorized_review = vectorize([number_reviews[i]])
        accuracy = model.predict(vectorized_review)[0][0]
        print('Review  "'  +  reviews[i]  +  '"\n'  +  '  has  a  '  +
str(accuracy) + ' accuracy to be positive')
input_user_review()
```