

**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. 7381

\_\_\_\_\_

Трушников А.П.

Преподаватель

\_\_\_\_\_

Жукова Н.А..

Санкт-Петербург

2020

## Цель работы.

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

## Задачи.

- Ознакомиться с задачей регрессии
- Изучить способы представления текста для передачи в ИНС
- Достигнуть точность прогноза не менее 95%

## Ход работы.

1. Построить и обучить нейронную сеть для обработки текста

Была найдена архитектура, которая даёт точность  $\approx 90\%$ , параметры которой представлены в таблице 1–2.

Таблица 1

Оптимизатор	Функция потерь	Метрика качества обучения сети	Число эпох	batch_size
Adam	binary_crossentropy	accuracy	2	500

Таблица 2

Номер слоя	Описание слоя
1	<code>model.add(layers.Dense(50, activation="relu", input_shape=(10000,)))</code>
2	<code>model.add(layers.Dropout(0.4, noise_shape=None, seed=None))</code>
3	<code>model.add(layers.Dense(30, activation="relu"))</code>
4	<code>model.add(layers.Dropout(0.4, noise_shape=None, seed=None))</code>
5	<code>model.add(layers.Dense(30, activation="relu"))</code>
6	<code>model.add(layers.Dense(1, activation="sigmoid"))</code>

Графики точности и ошибки представлены на рис.1–2.

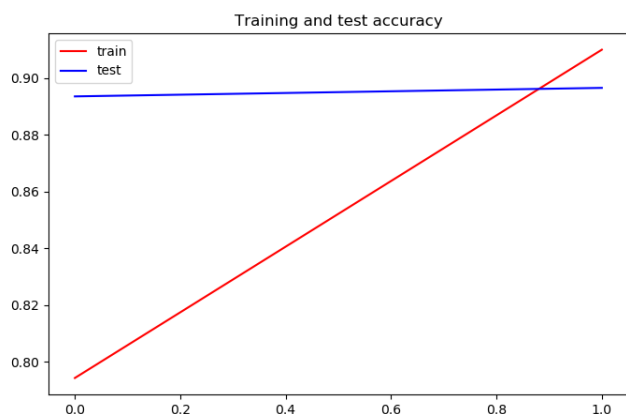


Рисунок 1 – График точности построенной сети

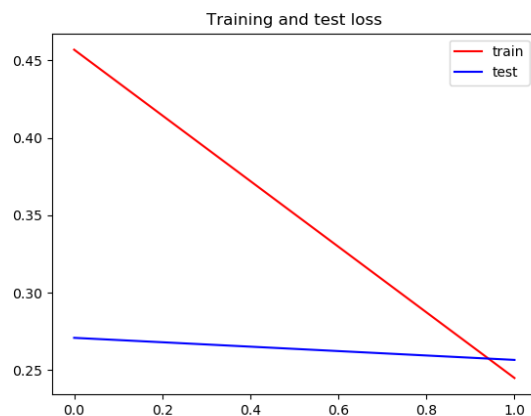


Рисунок 2 – График ошибок построенной сети

2. Исследовать результаты при различном размере вектора представления текста.

Графики точности представлены на рис.3–9, значения точности представлены в таблице 3.

Таблица 3.

<i>dimension</i> (размер вектора)	Значение точности
1000	0.8623
5000	0.8936
10000	0.8971
20000	0.8972
30000	0.8978
40000	0.898
50000	0.8988

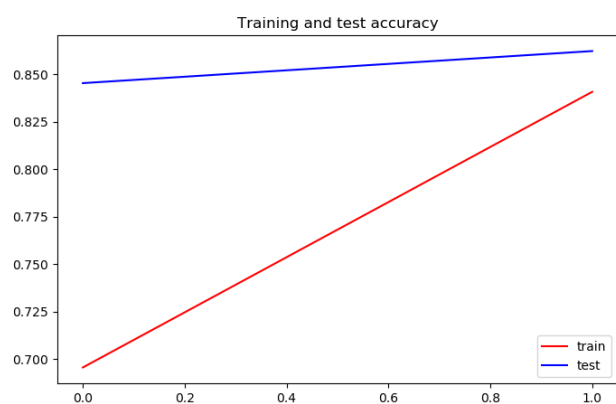


Рисунок 3 – График точности при значении *dimension* = 1000

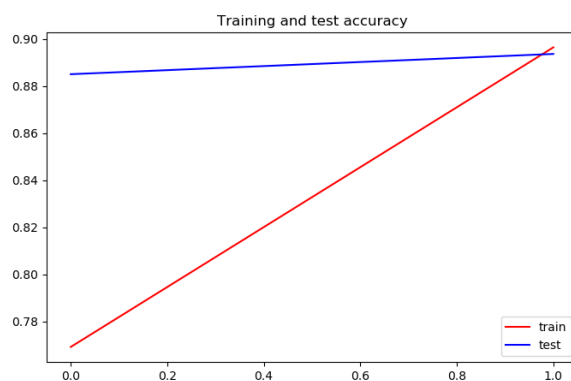


Рисунок 4 – График точности при значении *dimension* = 5000

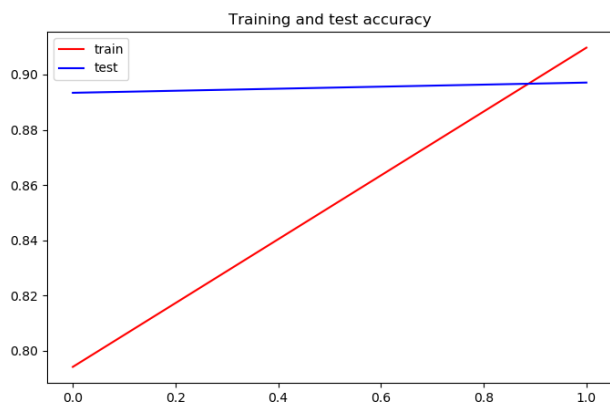


Рисунок 5 – График точности при значении  $dimension = 10000$

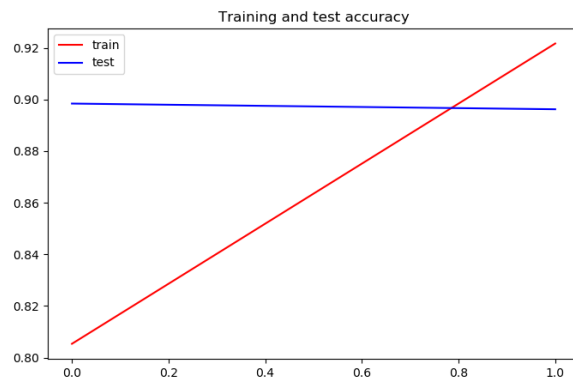


Рисунок 6 – График точности при значении  $dimension = 20000$

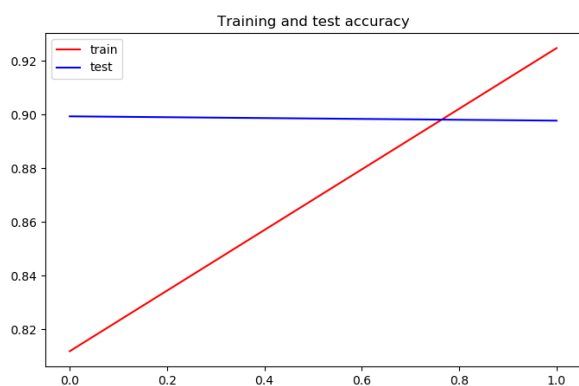


Рисунок 7 – График точности при значении  $dimension = 30000$

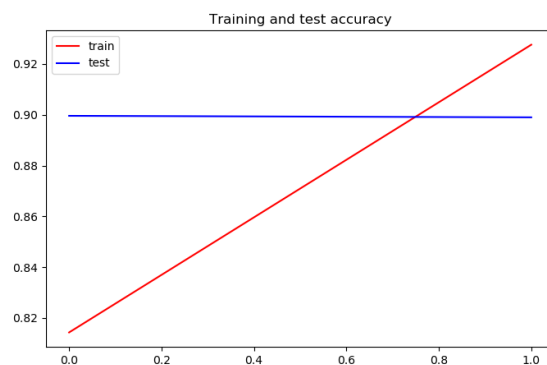


Рисунок 8 – График точности при значении  $dimension = 40000$



Рисунок 9 – График точности при значении  $dimension = 50000$

Как видно из таблицы 3 точность увеличивается при увеличении значения  $dimension$ .

3. Написать функцию, которая позволяет ввести пользовательский текст.

Была написана функция *prepInputText(text,target)*, которая представлена на рис. 10.

```
def prepInputText(text, target):
    def genNum(data, dic):
        data = data.translate(str.maketrans(dict.fromkeys(string.punctuation))).split()
        for i in range(len(data)):
            num = dic.get(data[i])
            if (num == None):
                data[i] = 0
            else:
                data[i] = num
        return data

    dic = dict(datasets.imdb.get_word_index())
    test_x = []
    test_y = np.array(target).astype("float32")
    for i in range(0, len(text)):
        test_x.append(genNum(text[i], dic))
    test_x = vectorize(test_x)
    return test_x, test_y
```

Рисунок 9 – код функции

При работе сети на пользовательском тексте точность составила  $\approx 60\%$ , текст представлен на рис.10. Результаты прогона текста через сеть представлены на рис.11.

```
x = [
    "This film is very interesting",
    "Actors played great in this film",
    "Uninteresting film",
    "Cracking. Keeps attention from start to finish",
    "In my opinion the movie is bad"
]
y = [1, 1, 0, 1, 0]
```

Рисунок 10 – текст

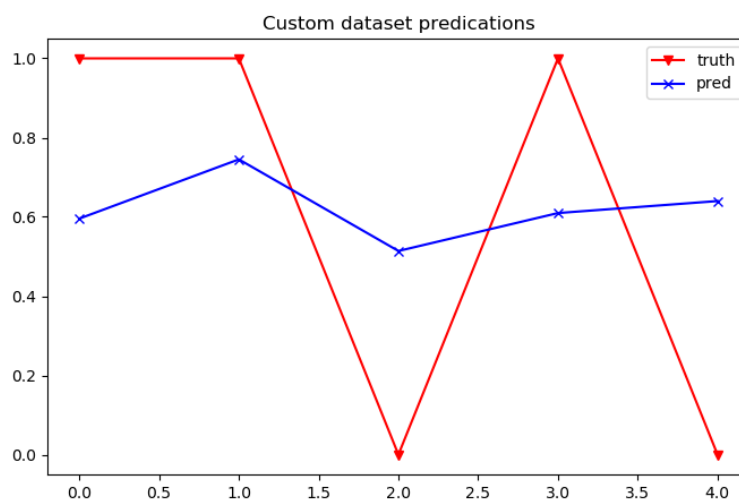


Рисунок 10 – результат работы программы

### **Выводы.**

Была изучена задача классификации обзоров из датасета IMDB. Также было изучено влияние длины вектора представления данных. Подобрана архитектура, дающая точность  $\approx 90\%$ . На собственных обзорах точность составила  $\approx 60\%$ .

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД

```
import string
import matplotlib.pyplot as plt
import numpy as np
from keras import models
from keras import layers
from keras.datasets import imdb
from tensorflow.keras import datasets

def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results

def prepInputText(text, target):
    def genNum(data, dic):
        data =
    data.translate(str.maketrans(dict.fromkeys(string.punctuation))).split()
    for i in range(len(data)):
        num = dic.get(data[i])
        if (num == None):
            data[i] = 0
        else:
            data[i] = num
    return data

    dic = dict(datasets.imdb.get_word_index())
    test_x = []
    test_y = np.array(target).astype("float32")
    for i in range(0, len(text)):
        test_x.append(genNum(text[i], dic))
    test_x = vectorize(test_x)
    return test_x, test_y

x = [
    "This film is very interesting",
    "Actors played great in this film",
    "Uninteresting film",
    "Cracking. Keeps attention from start to finish",
    "In my opinion the movie is bad"
]
y = [1, 1, 0, 1, 0]

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

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

model = models.Sequential()

model.add(layers.Dense(50, activation="relu", input_shape=(dimension,)))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(30, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(30, activation="relu"))
model.add(layers.Dense(1, activation="sigmoid"))

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

test_x, test_y = prepInputText(x, y)
custom_loss, custom_acc = model.evaluate(test_x, test_y)
print('custom_acc:', custom_acc)
preds = model.predict(test_x)
plt.figure(3, figsize=(8,5))
plt.title("Custom dataset predications")
plt.plot(test_y, 'r', marker='v', label='truth')
plt.plot(preds, 'b', marker='x', label='pred')
plt.legend()
plt.show()
plt.clf()

plt.figure(1, figsize=(8, 5))
plt.title("Training and test accuracy")
plt.plot(h.history['acc'], 'r', label='train')
plt.plot(h.history['val_acc'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()

plt.figure(1, figsize=(8, 5))
plt.title("Training and test loss")
plt.plot(h.history['loss'], 'r', label='train')
plt.plot(h.history['val_loss'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()

```