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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: Прогноз успеха фильмов по обзорам

Студент гр. 7383 Сычевский Р.А.

Преподаватель Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Порядок выполнения работы.

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

Ход работы.

Для исследования была разработана и использована программа. Код программы приведен в приложении А.

В ходе работы была получена модель, которая достигла точности 89%. Модель представлена на рисунке 1.

```
model = models.Sequential()
model.add(layers.Dense(50, activation="relu", input_shape=(DIMENSIONS,)))
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="linear"))
model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
model.add(layers.Dense(100, activation="relu"))
model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dense(1, activation="sigmoid"))
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
```

Рисунок 1 – Модель сети

Для исследования зависимости результата от различного размера вектора представления текста, протестируем нашу сеть при различных параметрах: 500, 1000, 5000, 10000. Р

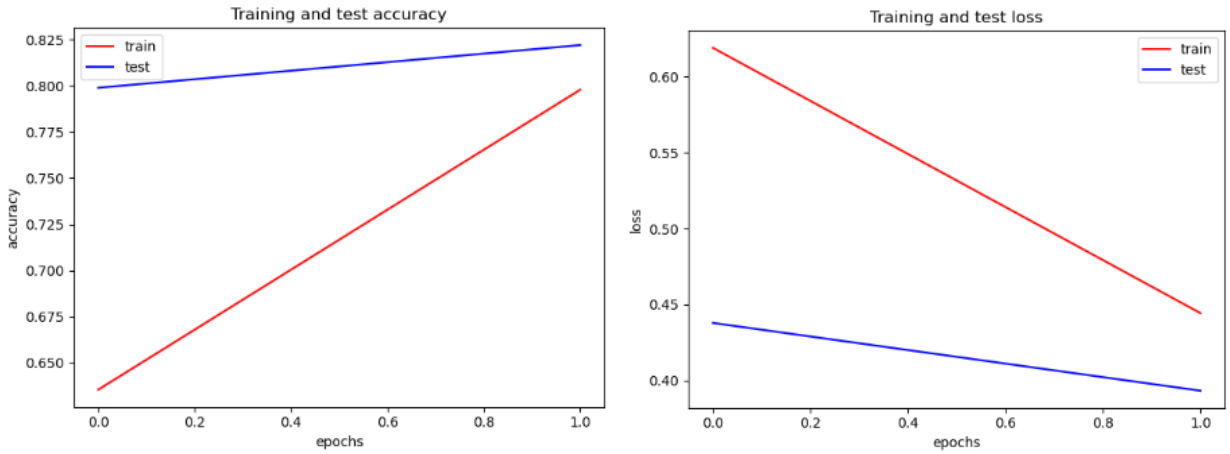

Рисунок 2 – Точность и потери сети при параметре 500

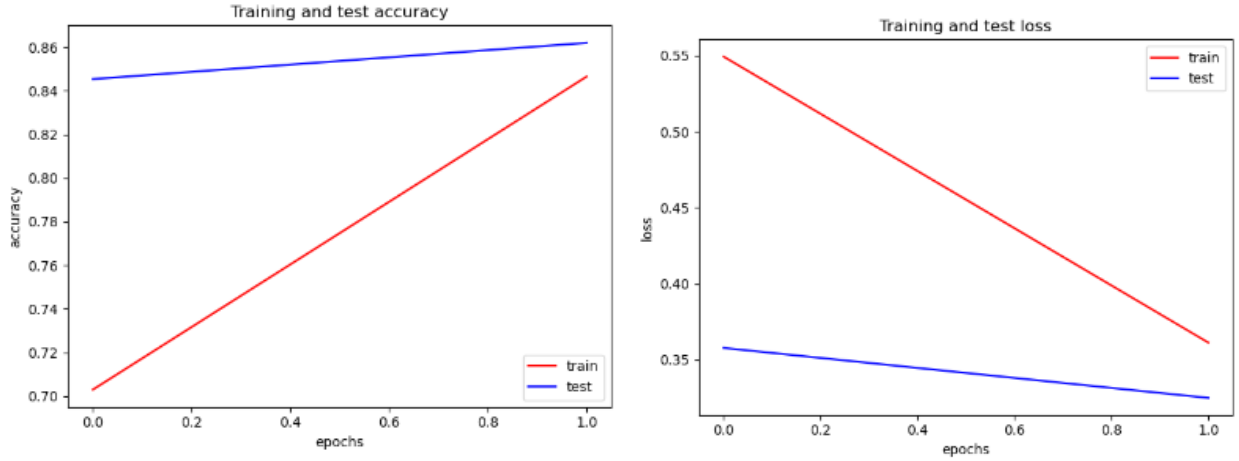

Рисунок 2 – Точность и потери сети при параметре 1000

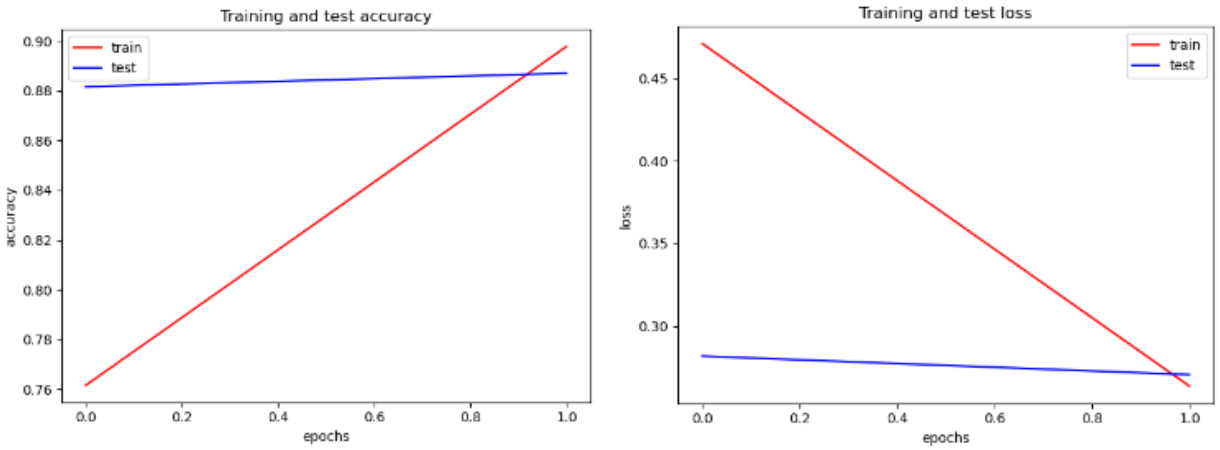

Рисунок 2 – Точность и потери сети при параметре 5000

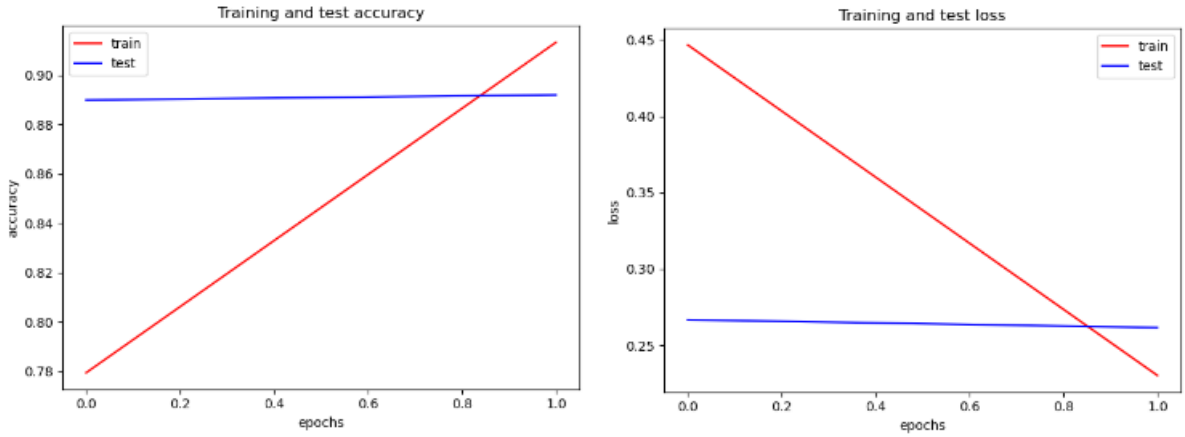

Рисунок 2 – Точность и потери сети при параметре 10000

Так же была написана функция, для загрузки обзора пользователя.

Выводы.

В ходе выполнения данной работы была создана сеть, которая может оценивать успех фильма по отзывам. Было исследовано влияние длины вектора представления текста на точность результата. Было выявлено, что при большем размере вектора, процесс обучения проходит дольше, но результаты точнее.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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


DIMENSIONS = 10000

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

def loadData():
    (training_data, training_targets), (testing_data, testing_targets)
= imdb.load_data(num_words=DIMENSIONS)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets),
axis=0)
    data = vectorize(data, DIMENSIONS)
    targets = np.array(targets).astype("float32")
    return data[10000:], targets[10000:], data[:10000],
targets[:10000]

def buildModel():
    model = models.Sequential()
    model.add(layers.Dense(50, activation="relu",
input_shape=(DIMENSIONS,)))
    model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation="linear"))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(100, activation="relu"))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation="relu"))
    model.add(layers.Dense(1, activation="sigmoid"))
    model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])
    return model

def plots(history):
    plt.title('Training and test accuracy')
    plt.plot(history.history['accuracy'], 'r', label='train')
    plt.plot(history.history['val_accuracy'], 'b', label='test')
    plt.xlabel("epochs")
```

```
    plt.ylabel("accuracy")
    plt.legend()
    plt.show()

    plt.title('Training and test loss')
    plt.plot(history.history['loss'], 'r', label='train')
    plt.plot(history.history['val_loss'], 'b', label='test')
    plt.xlabel("epochs")
    plt.ylabel("loss")
    plt.legend()
    plt.show()

def predict(review, model, dimensions=DIMENSIONS):
    punctuation = str.maketrans(dict.fromkeys(string.punctuation))
    review = review.lower().translate(punctuation).split(" ")
    indexes = imdb.get_word_index()
    encoded = []
    for w in review:
        if w in indexes and indexes[w] < dimensions:
            encoded.append(indexes[w])
    review = vectorize([np.array(encoded)], dimensions)
    return model.predict(review)[0][0]

train_x, train_y, test_x, test_y = loadData()
model = buildModel()
history = model.fit(train_x, train_y, epochs=2, batch_size=500,
validation_data=(test_x, test_y))
print(model.evaluate(test_x, test_y, verbose=1))
plots(history)

review1 = "It is a fantastic movie! This year's best film!"
review2 = "It is a usual film for one evening."
review3 = "it was very very bad!"
result = predict(review1, model)
print(result)
```