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 успехов фильма по обзорам»

Студентка гр. 8383 Сырцова Е.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2021

Цель

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%

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)

Ход работы

Была построена искусственная нейронная сеть, обучающаяся на датасете IMDb, для решения поставленной задачи следующей архитектуры:

```
model.add(layers.Dense(50,activation="relu",input_shape=dmns, )))
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"))
model.add(layers.Dense(1, activation="sigmoid"))
```

Было проведено исследование зависимости результата от длины вектора представления текста. Для исследования были взяты длины: 1000, 5000, 10000, 50000. Далее, при увеличении длины вектора, точность работы сети менялась

не значительно. Результаты запуска представлены на графиках точности и ошибки на рис.1-4.

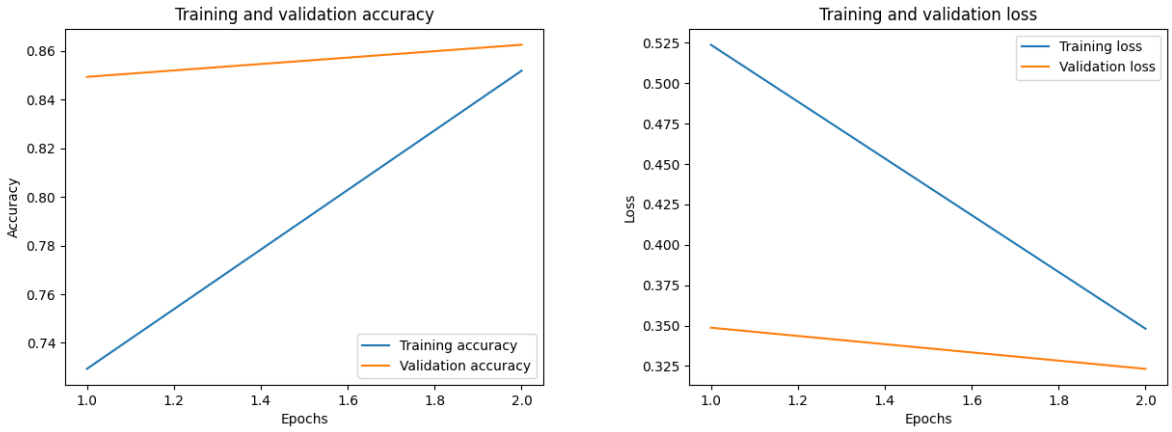

Рисунок 1 – Точность и ошибка при длине вектора 1000

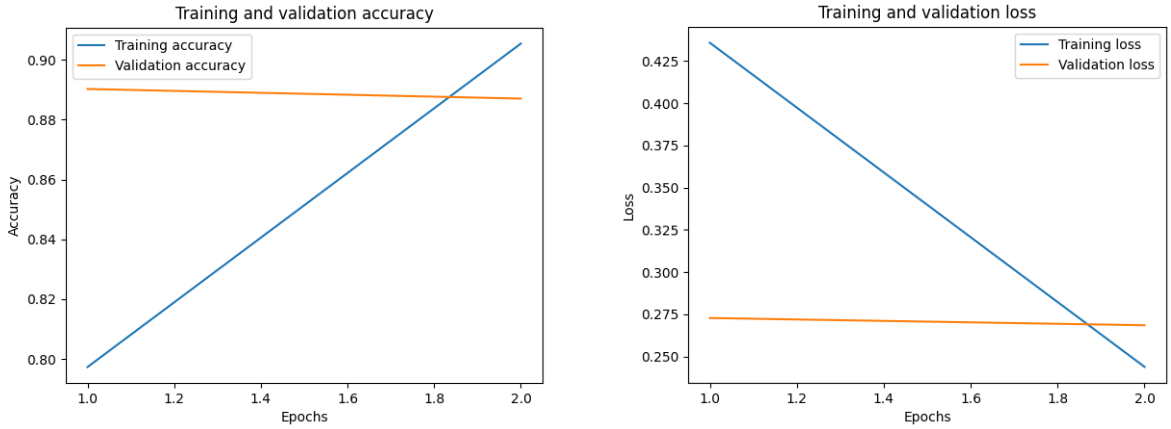

Рисунок 2 – Точность и ошибка при длине вектора 5000

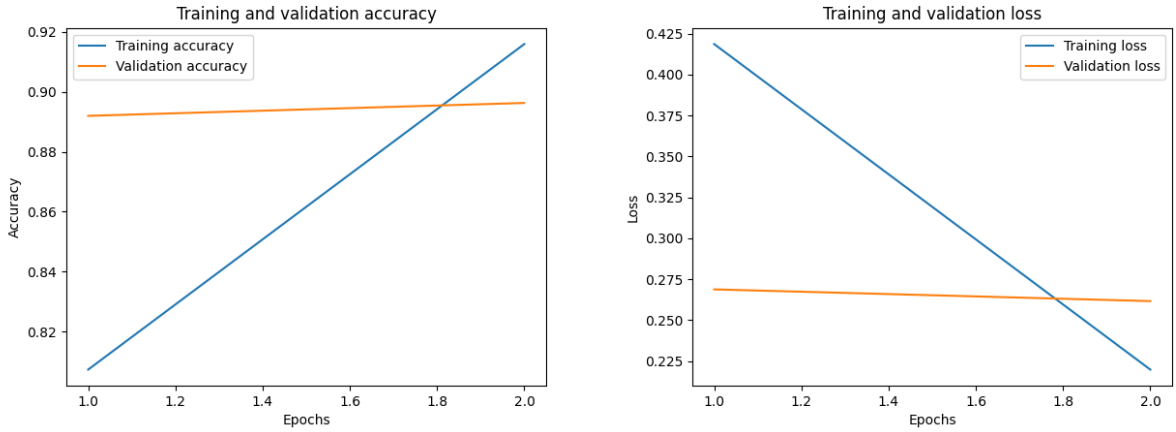

Рисунок 3 – Точность и ошибка при длине вектора 10000

Рисунок 4 – Точность и ошибка при длине вектора 50000

Проведенный эксперимент показал, что при увеличении длины ветора точность возрастает, однако примерно после значения длины вектора равным 5000 точность меняется незначительно. Наилучшая работа сети достигнута при значении 10000. Точность работы для разных размеров представлена в табл.1.

Таблица 1 – Точность

Размер вектора	1000	5000	10000
Точность в %	86,3	88,7	89,6

Для проверки корректной работы обученной сети была написана функция test_ review(), в которой оценивается тестовый отзыв.

review1 = ["Having already watched most of the series, I can make a personal conclusion that despite some shortcomings and bloopers that can be safely attributed to the artistry, and not the documentary authenticity of the film, the film adaptation came out very good. This is our first Russian film adaptation of the story about a freak and people."]

review2 = ["A story about two strange characters with whom you probably should have laughed, but it was not funny anywhere. The only funny scene is when Louis de Funes was in the frame, simply because he is funny in himself."]

Результатом работы сети при длине вектора 10000 было число 0,73 и 0,50, что соответствует содержаниям отзывов и характеризует один больше как положительный, а другой не очень хороший. Точность этого результата 90%.

Вывод

В процессе выполнения лабораторной работы была изучена зависимость размера вектора представления текста на результат работы сети. Таким образом, при увеличении размера точность увеличивается, но достигнуть точности 95% не представляется возможным.

Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А

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


review1 = ["Having already watched most of the series, I can make a personal
conclusion that despite some shortcomings and bloopers that can be safely
attributed to the artistry, and not the documentary authenticity of the film,
the film adaptation came out very good. This is our first Russian film
adaptation of the story about a freak and people."]
review2 = ["A story about two strange characters with whom you probably
should have laughed, but it was not funny anywhere. The only funny scene is
when Louis de Funes was in the frame, simply because he is funny in
himself."]
dmns = 10000


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


def build_model():
    (training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=dmns)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets), axis=0)

    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:]

    model = models.Sequential()
    model.add(layers.Dense(50, activation="relu", input_shape=(dmns, )))
    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"))
    model.add(layers.Dense(1, activation="sigmoid"))
```

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

    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(loss) + 1)
    plt.plot(epochs, loss, label='Training loss')
    plt.plot(epochs, val_loss, label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()

    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    plt.plot(epochs, acc, label='Training accuracy')
    plt.plot(epochs, val_acc, label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()
    results = model.evaluate(test_x, test_y)
    print(results)
    return model


def test_review():
    index = imdb.get_word_index()
    test_x = []
    words = []
    for line in review1:
        lines = line.translate(str.maketrans('', '', ',.?!:;()')).lower()
    for chars in lines:
        chars = index.get(chars)
    if chars is not None and chars < 10000:
        words.append(chars)
    test_x.append(words)
    test_x = vectorize(test_x)
    model = build_model()
    predict = model.predict(test_x)
    print(predict)


test_review()
```