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»

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цель работы.

Реализовать классификацию отзывов к фильмам по двум классам: положительные и отрицательные.

Задачи.

- Ознакомиться с задачей регрессии
- Изучить способы представления текста для передачи в ИНС

Требования.

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

Ход работы.

1. Была создана и обучена модель искусственной нейронной сети (код программы представлен в приложении А). Архитектура сети следующая: epochs = 2; batch_size = 500; loss = binary_crossentropy; optimizer = adam;
2. Исследуем влияние различных размеров вектора представления текста на результат обучения нейронной сети.

Первоначально размер вектора представления – 10 тыс. Результаты обучения сети приведены на рис. 1-2.

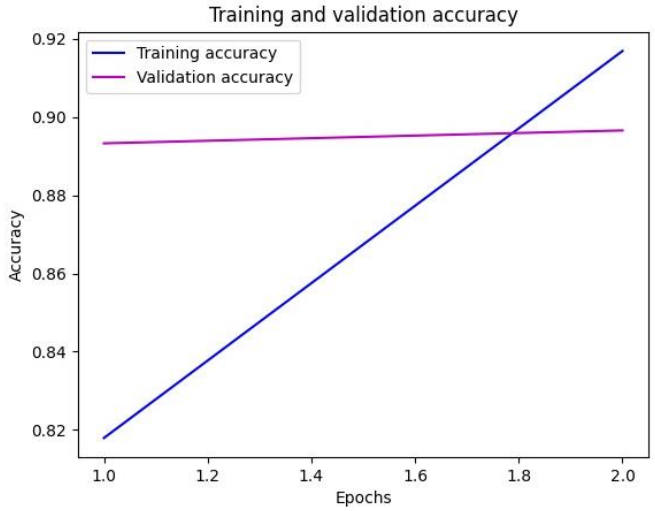

Рисунок 1 – График точности для вектора представления размером 10 тыс.

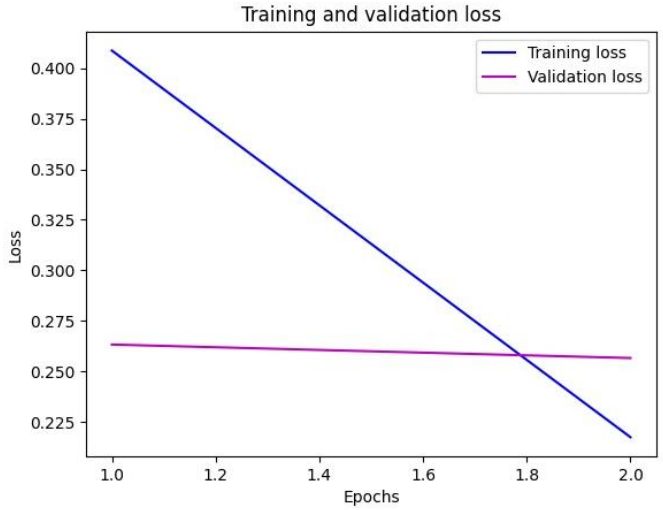

Рисунок 2 - График потерь для вектора представления размером 10 тыс.

Точность составила 0.8966, ошибка – 0.2566.

Далее была протестирована та же архитектура сети с размером вектора представления 2500. Результаты обучения сети представлены на рис. 3-4.

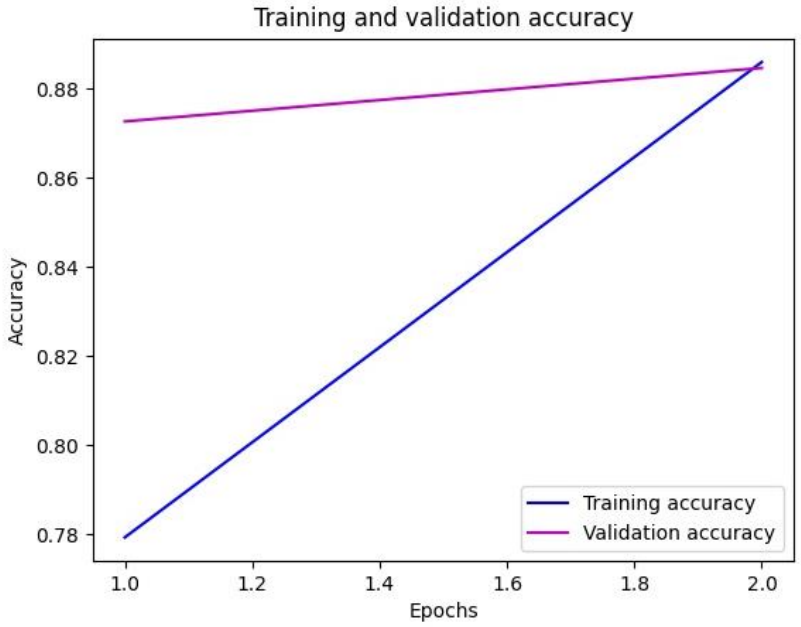

Рисунок 3 - График точности для вектора представления размером 2500

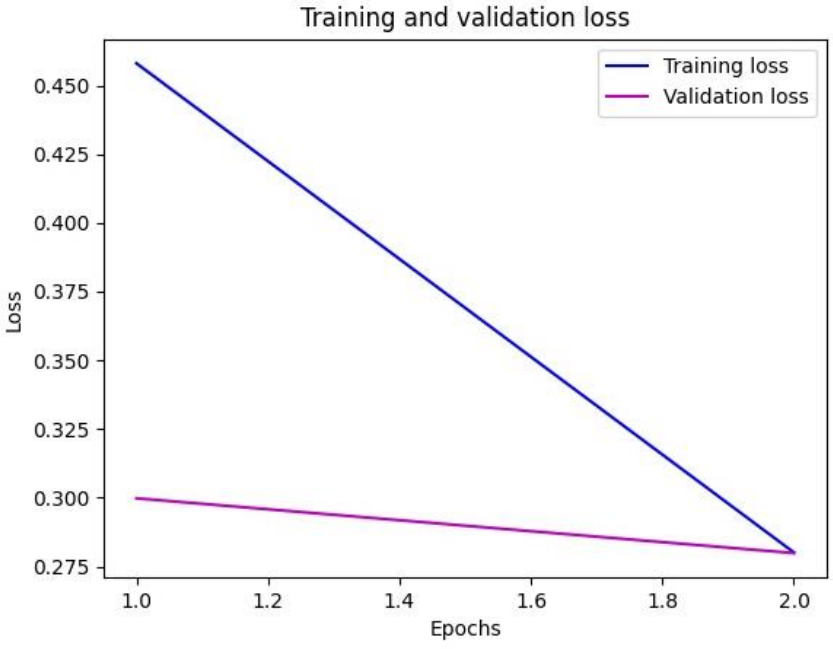

Рисунок 4 - График потерь для вектора представления размером 2500

Точность составила 0.8845, ошибка – 0.2798.

Затем, была исследована работа сети с размером вектора представления 500. Результаты обучения продемонстрированы на рис. 5 – 8.

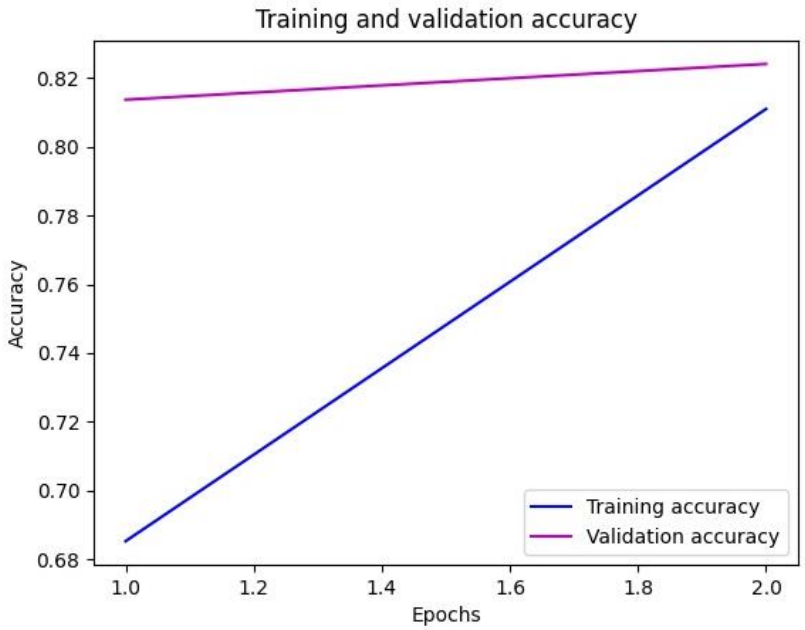

Рисунок 5 – График точности для вектора представления размером 500

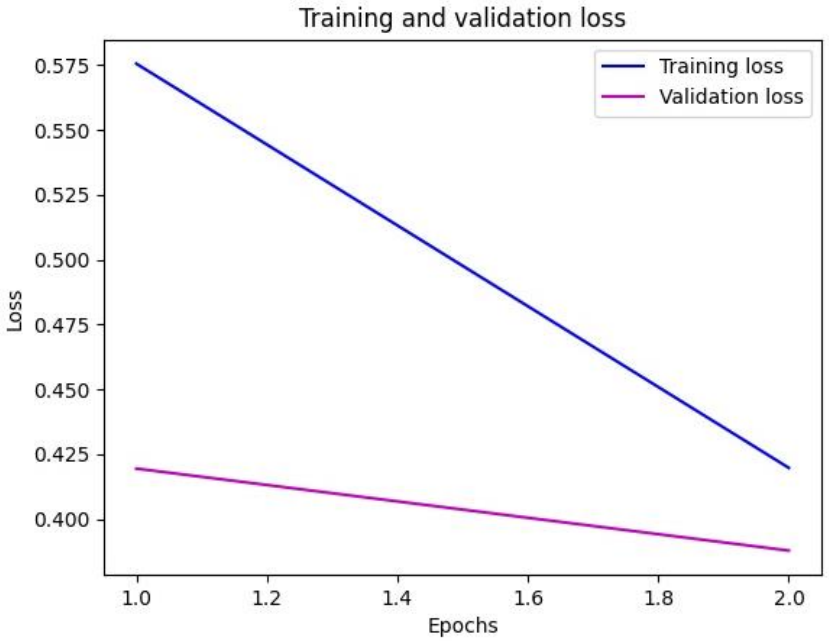

Рисунок 6 – График потерь для вектора представления размером 500

Точность составила 0.8240, ошибка – 0.3878.

По результатам видно, что с уменьшением размера вектора представления текста уменьшается точность и увеличивается ошибка. Это может быть связано с уменьшением в обзоре количества слов, характеризующих положительное или отрицательное отношение.

3. Написав функцию считывания пользовательского текста из файла,

протестируем работу программы на примере отзыва: «Wonderful! This is the best movie that I've seen so far! I loved just everything! Amazing job!». Результат – 0.7038, что соответствует положительному настроению отзыва.

Вывод.

В ходе выполнения лабораторной работы была построена модель искусственной нейронной сети для обработки текста. Было изучено влияние различных размеров вектора представления текста на результат обучения нейронной сети. Максимальная точность 0.8966 и минимальная ошибка 0.2566 соответствовали максимальному размеру вектора – 10 тыс. С уменьшением размера вектора ухудшалась точность и росла ошибка. Была также написана функция, позволяющая считывать из файла пользовательский текст.

ПРИЛОЖЕНИЕ А

```
import matplotlib.pyplot as plt

import numpy as np

from keras import Sequential

from keras import layers

from keras.utils import to_categorical

from keras import models

from keras.datasets import imdb


dim = 500

filename = 'text.txt'


def vectorize(sequences, dimension = dim):

    results = np.zeros((len(sequences), dimension))

    for i, sequence in enumerate(sequences):

        results[i, sequence] = 1

    return results


def load_text(filename):

    text = []

    with open(filename, 'r') as file:

        for line in file.readlines():

            text += [s.strip(''.join(['.', ',', ':', ';', '!', '?',
'(', ')'])).lower() for s in line.strip().split()]

    print(text)
```

```
        index = imdb.get_word_index()
        words = []
        for s in text:
            if s in index and index[s] < 10000:
                words.append(index[s])
        test_text(np.array(words))

    def test_text(text):
        text = vectorize([text])
        print(text)
        model = test_model()
        prediction = model.predict(text)
        print(prediction)

    def test_model():
        (training_data, training_targets), (testing_data, testing_targets) = imdb.load_data(num_words=dim)
        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 = Sequential()
        model.add(layers.Dense(50, activation = "relu", input_shape=(dim,
)))
        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, 'b', label='Training loss')
        plt.plot(epochs, val_loss, 'm', 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, 'b', label='Training accuracy')
    plt.plot(epochs, val_acc, 'm', 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

#load_text(filename)
test_model()
```