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 _____ Бергалиев М.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы: прогноз успеха фильмов по обзорам.

Порядок выполнения работы.

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

Ход работы.

0. Исходный код программы представлен в приложении.
1. Выберем модель сети. Возьмем три скрытых слоя с 50 нейронами с активацией relu, слой разреживания с вероятностью 0.3 между 1-ым и 2-ым скрытыми слоями, слой разреживания с вероятностью 0.2 между 2-ым и 3-им скрытыми слоями и на выходе слой с одним нейроном с функцией активации sigmoid. Результат обучения показан на рис. 1.

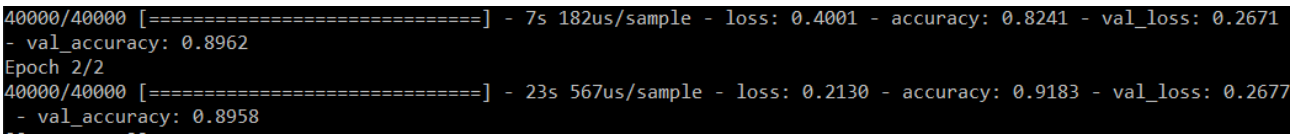

Рисунок 1 — Результат обучения модели

Как видно, модель имеет приличную точность в 89,5% на контрольных данных.

2. Исследуем влияние длины вектора представления данных на результат обучения. Результаты показаны на рис. 2-4.

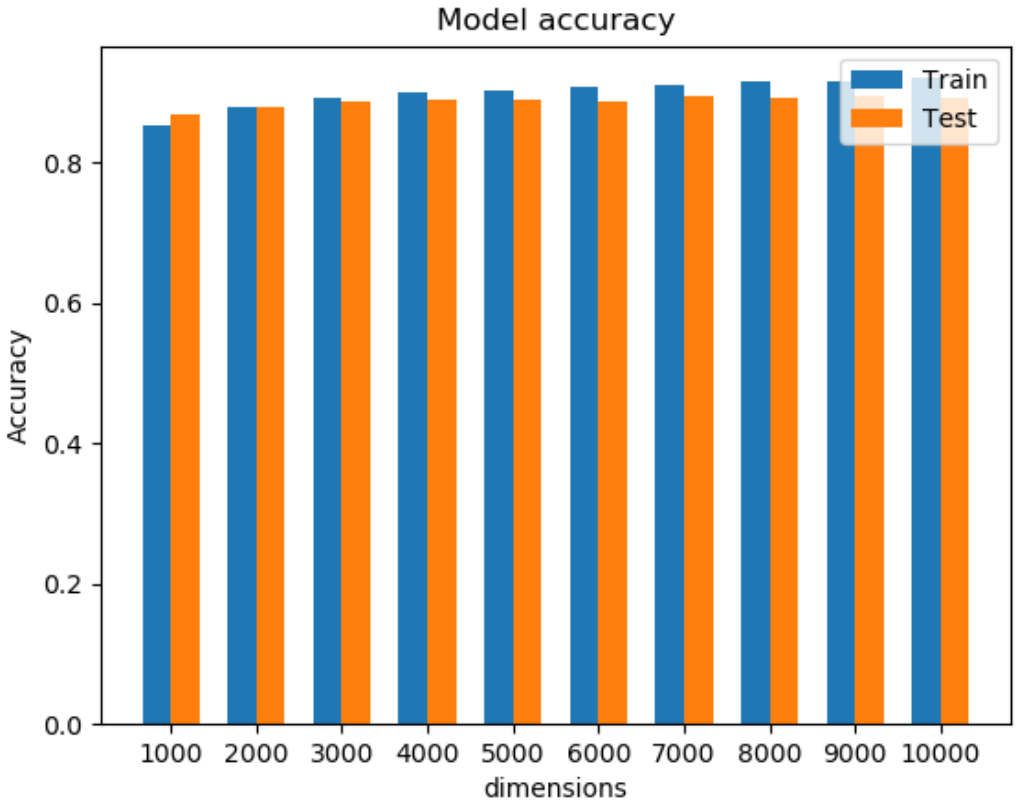

Рисунок 2 — Точность при изменении длины вектора от 1000 до 10000

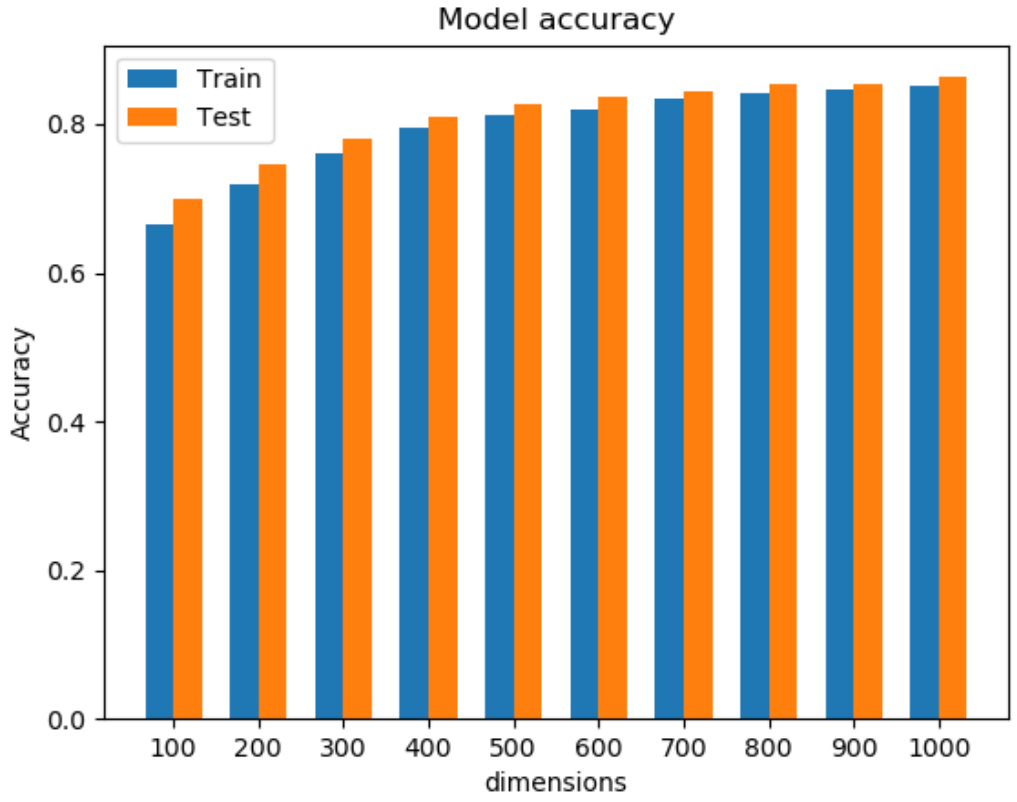

Рисунок 3 — Точность при изменении длины вектора от 100 до 1000

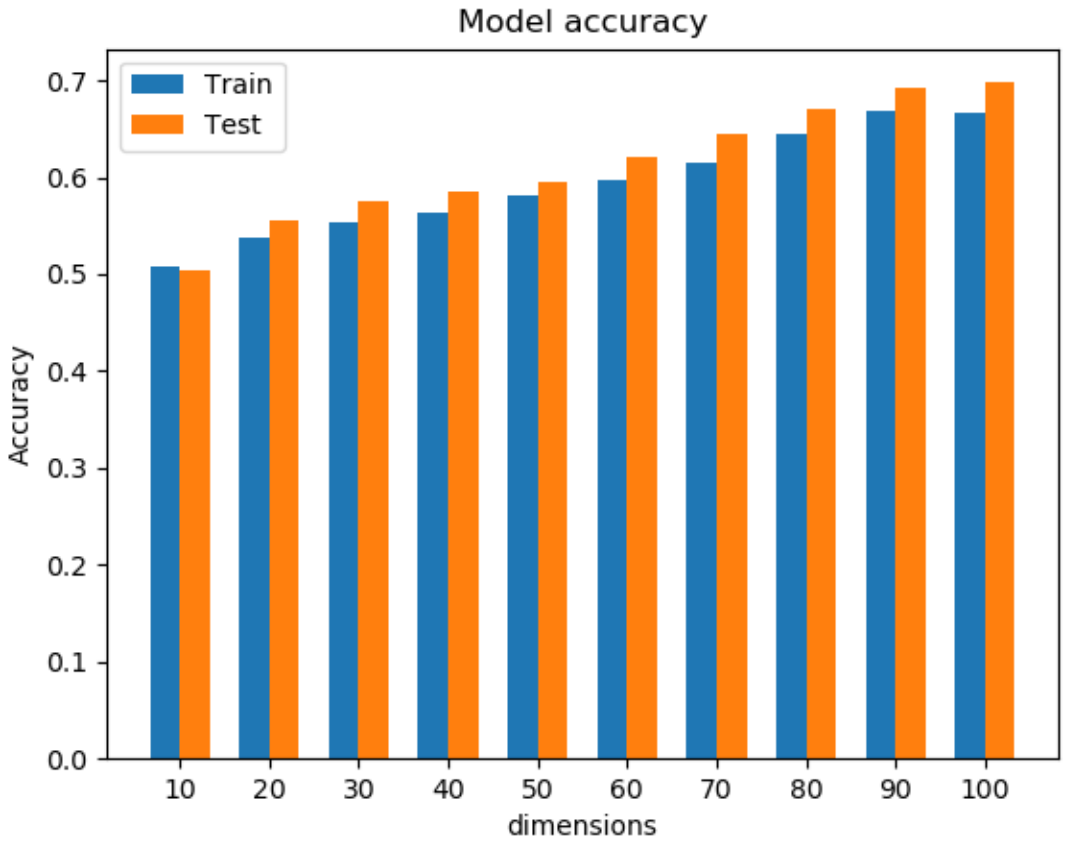

Рисунок 4 —Точность при изменении длины вектора от 10 до 100

Как видно, при уменьшении длины вектора точность падает. Однако даже при длине 500 сеть дает 80% точность, что вполне неплохо. При длине 10 сеть работает не лучше монетки. По всей видимости, в 10 наиболее встречающихся словах отсутствует какая-либо полезная информация об окраске обзора. При длине 20 уже заметно, что какие-то полезные данные сеть смогла выявить.

3. Была написана функция загрузки собственного текста `loadText`. Проверим работу модели на следующем тексте:

Scenario of this movie is really DUMP! But actors played very well, so it saved this film.

Сеть в качестве ответа выдала значение 0.7802197. Сеть отнесла обзор больше как положительный, чем отрицательный. Действительно, хоть сценарий фильма оценивается очень негативно в обзоре, все же в нем говорится, что актерская игра спасла фильм.

Выводы: Было изучено влияние длины вектора представления данных. При относительно небольшой длине, равной 500, достигается точность в 80%. Была написана функция, считывающая текст из файла. На примере собственного текста была показана работа модели.

ПРИЛОЖЕНИЕ

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

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


def loadData(dimension):
                (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:]
    return (train_x, train_y, test_x, test_y)

def build_model():
    model = models.Sequential()
    model.add(layers.Dense(50, activation = "relu"))
    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"])
    return model

def plot_bars(res_train, res_test, dimensions):
```

```
    plt.bar(np.arange(len(dimensions)) * 3, res_train, width=1)
    plt.bar(np.arange(len(dimensions)) * 3 + 1, res_test, width=1)
    plt.xticks([3*i + 0.5 for i in range(0,len(dimensions))],
               labels=[dimension for dimension in dimensions])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('dimensions')
    plt.legend(['Train', 'Test'])
    plt.savefig('bars.png')
    plt.clf()

def test_model(dimension):
    train_x, train_y, test_x, test_y = loadData(dimension)
    model = build_model()
     results = model.fit(train_x, train_y, epochs= 2, batch_size =
500,
                         validation_data = (test_x, test_y))
                        return      (results.history['accuracy'][-1],
results.history['val_accuracy'][-1])

def test_dimensions(dimensions):
    res_train = []
    res_test = []
    for dimension in dimensions:
        train, test = test_model(dimension)
        res_train.append(train)
        res_test.append(test)
    plot_bars(res_train, res_test, dimensions)

def testOwnText(text):
    train_x, train_y, test_x, test_y = loadData(10000)
    model = build_model()
     results = model.fit(train_x, train_y, epochs= 2, batch_size =
500,
                         validation_data = (test_x, test_y))
    text = vectorize([text])
    res = model.predict(text)
    print(res)

def loadText(filename):
    punctuation = ['.',',',':',';','!','?','(',')']
    text = []
    with open(filename, 'r') as f:
        for line in f.readlines():
              text += [s.strip(''.join(punctuation)).lower() for s
in line.strip().split()]
```

```
    print(text)
    indexes = imdb.get_word_index()
    encoded = []
    for w in text:
        if w in indexes and indexes[w] < 10000:
            encoded.append(indexes[w])
    print(encoded)
    return np.array(encoded)

if __name__ == '__main__':
    filename = 'text.txt'
    text=loadText(filename)
    testOwnText(text)
```