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

ОТЧЕТ

по лабораторной работе №6

"Прогноз успеха фильмов по обзорам"

по дисциплине «Искусственные нейронные сети»

Студент гр. 8383 _____ Бабенко Н.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель.

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

Задание.

1. Ознакомиться с задачей классификации
2. Изучить способы представления текста для передачи в ИНС
3. 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)

Выполнение работы.

Листинг модели представлен ниже:

```
model = models.Sequential()
model.add(layers.Dense(50, activation="relu", input_shape=(dimension,)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(50, 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))
```

Тестирование будет проводиться с числом входных параметров 2000, 4000, 6000, 8000, 10000.

Точность и ошибки при длине 2000 представлены ниже:

```
Epoch 1/2
loss: 0.6287 - accuracy: 0.6229 - val_loss: 0.3262 - val_accuracy: 0.8622
Epoch 2/2
loss: 0.3458 - accuracy: 0.8560 - val_loss: 0.2927 - val_accuracy: 0.8761
```

Точность и ошибки при длине 4000 представлены ниже:

```
Epoch 1/2
loss: 0.6478 - accuracy: 0.5875 - val_loss: 0.2991 - val_accuracy: 0.8757
Epoch 2/2
loss: 0.3155 - accuracy: 0.8768 - val_loss: 0.2701 - val_accuracy: 0.8910
```

Точность и ошибки при длине 6000 представлены ниже:

```
Epoch 1/2
loss: 0.5990 - accuracy: 0.6548 - val_loss: 0.2784 - val_accuracy: 0.8844
Epoch 2/2
loss: 0.2838 - accuracy: 0.8898 - val_loss: 0.2657 - val_accuracy: 0.8927
```

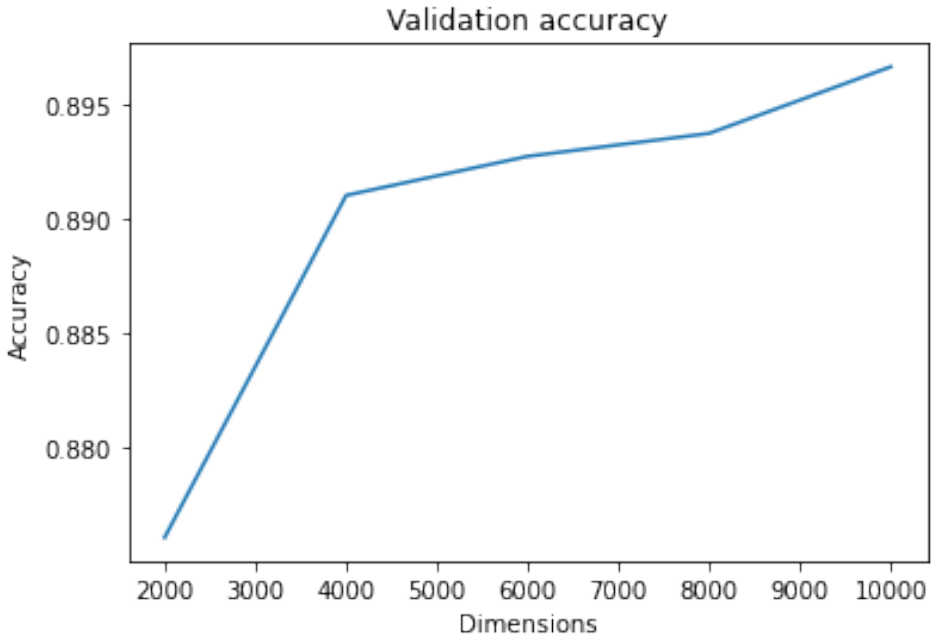
Точность и ошибки при длине 8000 представлены ниже:

```
Epoch 1/2
loss: 0.5902 - accuracy: 0.6661 - val_loss: 0.2722 - val_accuracy: 0.8883
Epoch 2/2
loss: 0.2630 - accuracy: 0.8989 - val_loss: 0.2613 - val_accuracy: 0.8937
```

Точность и ошибки при длине 10000 представлены ниже:

```
Epoch 1/2
loss: 0.5985 - accuracy: 0.6604 - val_loss: 0.2710 - val_accuracy: 0.8891
Epoch 2/2
loss: 0.2652 - accuracy: 0.8981 - val_loss: 0.2591 - val_accuracy: 0.8966
```

Представим график зависимости точности от числа параметров:



Из графика видно, что увеличением числа параметров точность возрастает.

Листинг функции, которая загружает пользовательский текст из массива, подготавливает его к передаче в нейронную сеть, выводит результат нейронной сети:

```
def text_load():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(answers)
    for string in strings:
        words  =  string.replace(',',  '  ').replace('.',  '  ').replace('?',  '
').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = dictionary.get(word)
            if word is not None and word < 10000:
                num_words.append(word)
        test_x.append(num_words)
    test_x = [vectorize(test_x)]
    model = build_model(10000)
    (train_x, train_y), (s1, s2) = prepare_data(10000)
    model.fit(train_x, train_y, epochs=2, batch_size=500)
    predictions = model.predict(test_x)
    print(predictions)
```

Пользовательский текст:

In the film, a demonstration of outright stupidity, humor below the waist, toilet scenes with diarrhea, funny chases for robbers at a funeral. Agitation for easy money, since working is not an option: 'work for slaves'. Open neglect and insult of the older generation, although in the Caucasus it is just the opposite. The theme of the film is, in general, friendship, and the most disgusting thing is how the main character treats his best friend.

Результат нейронной сети:

0.2393991, то есть отзыв отрицательный, что верно.

Выводы.

В ходе выполнения лабораторной работы было изучено решение задачи классификации отзывов на фильмы как положительные или отрицательные.

Было изучено, как передавать нейронной сети текстовые данные, а также, как влияет количество входных параметров.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл lr6.py

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

strings = ["In the film, a demonstration of outright stupidity, humor below the waist, toilet scenes with diarrhea, funny chases for robbers at a funeral. Agitation for easy money, since working is not an option: 'work for slaves'. Open neglect and insult of the older generation, although in the Caucasus it is just the opposite. The theme of the film is, in general, friendship, and the most disgusting thing is how the main character treats his best friend."]


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


def prepare_data(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(dimension):
    model = models.Sequential()
    model.add(layers.Dense(50, activation="relu", input_shape=(dimension,)))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(50, activation="relu"))
    model.add(layers.Dropout(0.5))
    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 model_fit(train_x, train_y, test_x, test_y, dimension):
    model = build_model(dimension)
    H     =     model.fit(train_x,     train_y,     epochs=2,     batch_size=500,
validation_data=(test_x, test_y))
    draw_plot(H)
    return H


def test_dim(dimension):
    (train_x, train_y), (test_x, test_y) = prepare_data(dimension)
    H = model_fit(train_x, train_y, test_x, test_y, dimension)
    return H.history['val_accuracy'][-1]


def test_dimensions():
    dimensions = [2000, 4000, 6000, 8000, 10000]
    val_accs = []
    for dim in dimensions:
        val_accs.append(test_dim(dim))
    plt.plot(dimensions, val_accs)
    plt.title('Validation accuracy')
    plt.xlabel('Dimensions')
    plt.ylabel('Accuracy')
    plt.show()


def text_load():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(answers)
    for string in strings:
        words  =  string.replace(',',  '  ').replace('.',  '  ').replace('?',  '
').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = dictionary.get(word)
            if word is not None and word < 10000:
                num_words.append(word)
        test_x.append(num_words)
    test_x = [vectorize(test_x)]
    model = build_model(10000)
    (train_x, train_y), (s1, s2) = prepare_data(10000)
```

```
    model.fit(train_x, train_y, epochs=2, batch_size=500)
    predictions = model.predict(test_x)
    print(predictions)


test_dimensions()
text_load()
```