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. 8383 _____ Федоров И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

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

Выполнение работы.

Были импортированы необходимые для работы класы, модули, функции, а также данные *imdb*.

```
import pandas
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import plot_model

#lr 6
from tensorflow.keras import layers
from tensorflow.keras.datasets import imdb    #импорт данных imdb
```

Была построена нейронная сеть, для прогноза успеха фильма по обзорам, на основе методических указаний. Ее точность на контрольных данных составила 89.06%. Общий объем данных составил 20 000 образцов, из-за нехватки ресурсов. Схема модели приведена на рис. 1. Число эпох было задано равным 2 (мет. указания). Из рисунка 2 можно убедиться, что переобучение действительно наступает после 2-й эпохи.

При уменьшении вектора представления с 10000 до 4000, то точность на тестовых данных снизилась до 88.0125%. Снизив вектор еще раз до 2000, точность упадет до 86.21%. Можно сделать вывод, что недостаточный объем представления негативно влияет на точность модели.

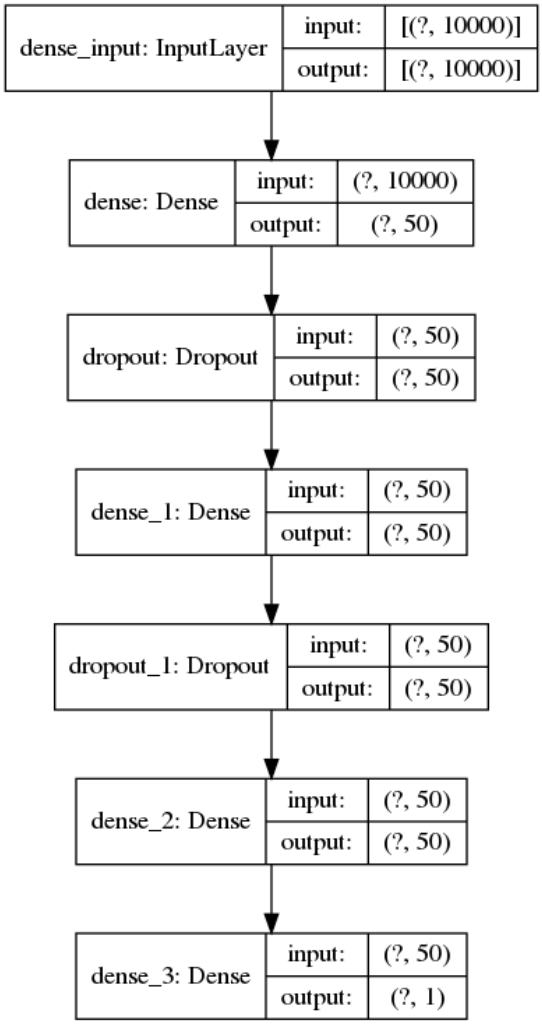


Рисунок 1 - Схема модели

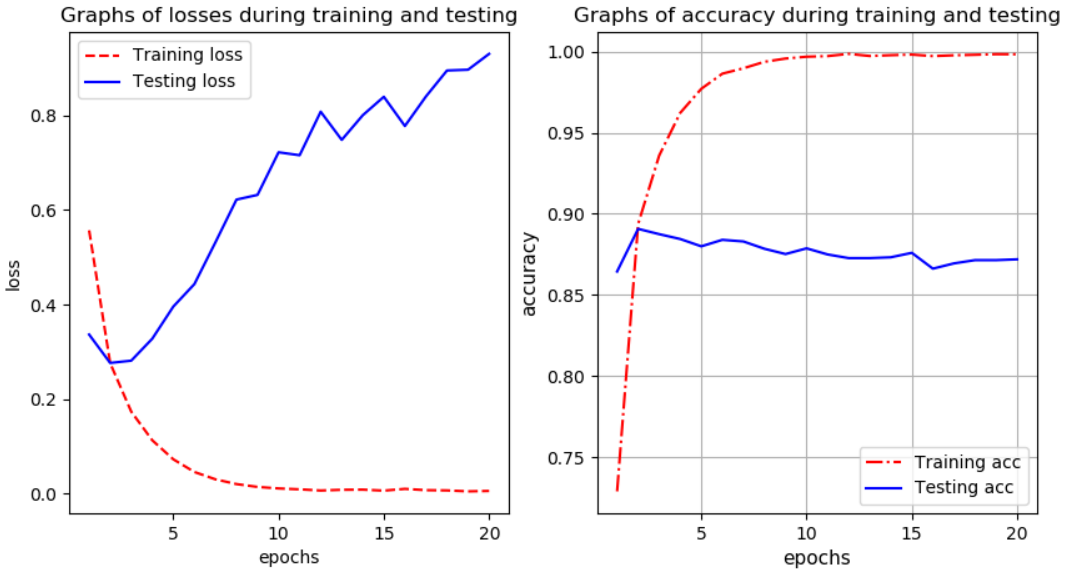
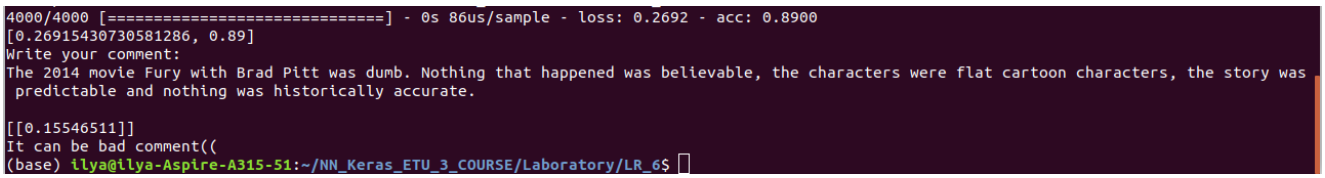


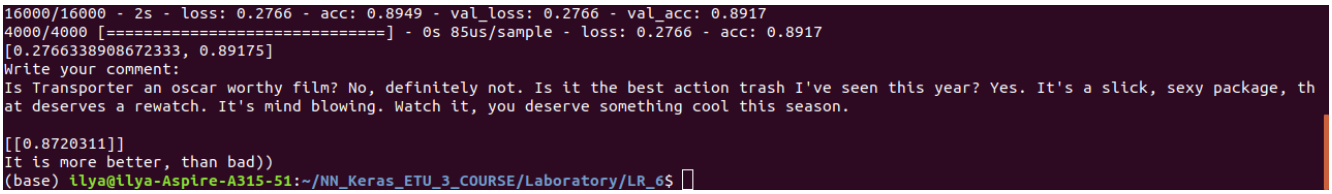
Рисунок 2

Была написана функция, позволяющая ввести пользовательский текст. Код представлен ниже. Функция `input_user_comment()` считывает строку, которую вводит пользователь и удаляет из нее лишние символы. Функция `vectorize_user_comment()` приводит к виду, удобному для ИНС.

```
def input_user_comment():

    def clean_str(str):
        str = "".join(c for c in str if c.isalpha() or
c.isspace()).expandtabs(tabsize=1).strip().lower()
        str = ' '.join(str.split())
        return str

    print('Write your comment: ')
    result_str = ''
    while True:
        str = input()
        if str:
            result_str += str + " "
        else:
            break

    res_vect = clean_str(result_str).split(' ')
    return res_vect


def vectorize_user_comment(vector_len=10000, need_decoded=False):

    def words2index(words, words_index, vector_len=10000):
        result = []
        for elem in words:
            ind = words_index.get(elem)
            if ind is not None and ind < vector_len:
                result.append(ind+3)
        return result


    words_index = imdb.get_word_index()
    usr_comment = input_user_comment()

    index_vec = words2index(usr_comment, words_index, vector_len)
#print(index_vec)

    if need_decoded:
        reverse_index = dict([(value, key) for (key, value) in
words_index.items()])
        decoded = " ".join( [reverse_index.get(i - 3, "#") for i in
index_vec] )
        print(decoded)


    res = vectorize(np.asarray([index_vec]))

    return res
```

Ниже показаны примеры использования:





Выводы.

В ходе выполнения данной работы была реализована ИНС для прогнозирования успеха фильма по обзорам. Была реализована функция, для ввода пользовательского текста. Был изучен один из способов представления текста для передачи в ИНС.

Приложение

```
import pandas
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import plot_model

#lr 6
from tensorflow.keras import layers
from tensorflow.keras.datasets import imdb   #импорт данных imdb


bool_is_check_data = False
need_decode = False

bool_need_compress = False
view_vector_len = 10000
data_size = 20000
epochs_num = 2


(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=view_vector_len)    # 10000 наиболее уникальных слов
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)



def input_user_comment():

    def clean_str(str):
        str = "".join(c for c in str if c.isalpha() or
c.isspace()).expandtabs(tabsize=1).strip().lower()
        str = ' '.join(str.split())
        return str

    print('Write your comment: ')
    result_str = ''
    while True:
        str = input()
        if str:
            result_str += str + " "
        else:
            break

    res_vect = clean_str(result_str).split(' ')
    return res_vect


def vectorize_user_comment(vector_len=10000, need_decoded=False):

    def words2index(words, words_index, vector_len=10000):
        result = []
        for elem in words:
            ind = words_index.get(elem)
            if ind is not None and ind < vector_len:
                result.append(ind+3)
        return result
```

```
    words_index = imdb.get_word_index()
    usr_comment = input_user_comment()

    index_vec = words2index(usr_comment, words_index, vector_len)
#print(index_vec)

    if need_decoded:
        reverse_index = dict([(value, key) for (key, value) in
words_index.items()])
        decoded = " ".join( [reverse_index.get(i - 3, "#") for i in
index_vec] )
        print(decoded)


    res = vectorize(np.asarray([index_vec]))

    return res



# графики потерь и точности при обучении и тестирования
def plot_model_loss_and_accuracy(history, figsize_=(10,5)):
    plt.figure(figsize=figsize_)
    train_loss = history.history['loss']
    test_loss = history.history['val_loss']
    train_acc = history.history['acc']
    test_acc = history.history['val_acc']
    epochs = range(1, len(train_loss) + 1)

    plt.subplot(121)
    plt.plot(epochs, train_loss, 'r--', label='Training loss')
    plt.plot(epochs, test_loss, 'b-', label='Testing loss')
    plt.title('Graphs of losses during training and testing')
    plt.xlabel('epochs')
    plt.ylabel('loss')
    plt.legend()
    plt.subplot(122)
    plt.plot(epochs, train_acc, 'r-.', label='Training acc')
    plt.plot(epochs, test_acc, 'b-', label='Testing acc')
    plt.title('Graphs of accuracy during training and testing')
    plt.xlabel('epochs', fontsize=11, color='black')
    plt.ylabel('accuracy', fontsize=11, color='black')
    plt.legend()
    plt.grid(True)

    plt.show()



if need_decode:
    index = imdb.get_word_index()
    reverse_index = dict([(value, key) for (key, value) in index.items()])
    decoded = " ".join( [reverse_index.get(i - 3, "#") for i in data[0]] )
    print(decoded)


# функция векторизации данных
def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    if bool_need_compress:
        results = results.astype(np.float32)
```

```
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results

#  векторизируем (one-hot) данные и делим в пропорции 80:20
data = data[:data_size]

targets = targets[:data_size]
data_size = int(data_size / 5)

targets = np.array(targets).astype("float32")
test_x = data[:data_size]
test_y = targets[:data_size]
train_x = data[data_size:]
train_y = targets[data_size:]


test_x = vectorize(test_x, view_vector_len)
train_x = vectorize(train_x, view_vector_len)


# создаем и обучаем модель
model = Sequential()
model.add(layers.Dense(50, activation="relu",
input_shape=(view_vector_len,)))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.2))
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=epochs_num,
    batch_size=500,
    validation_data=(test_x, test_y),
    verbose=2
)

if epochs_num > 4:
    plot_model_loss_and_accuracy(history)

test_score = model.evaluate(test_x,test_y)
print(test_score)

vec = vectorize_user_comment(need_decoded=False)
res = model.predict(vec)
print(res)

if res[0] > 0.5:
    print('It is more better, than bad))')
else:
    print('It can be bad comment((')
```