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»

Студент гр. 8383 _____ Муковский Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

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

- Построить и обучить нейронную сеть для обработки текста
- Исследовать результаты при различном размере вектора представления текста
- Написать функцию, которая позволяет ввести пользовательский текст

Ход работы

Была использована нейронная сеть, описанная в указаниях к работе.

Модель ИНС состоит из четырех полносвязных слоев, между которыми слои Dropout. Модель обучалась на 2 эпохах с batch_size = 500.

Была достигнута точность в 89% на валидационных данных.

```
_________________________________________________________________
Epoch 1/2
80/80 [==============================] - 2s 21ms/step - loss: 0.5438 - accuracy: 0.7140 - val_loss: 0.2634 - val_accuracy: 0.8927
Epoch 2/2
80/80 [==============================] - 1s 17ms/step - loss: 0.2286 - accuracy: 0.9154 - val_loss: 0.2590 - val_accuracy: 0.8951
0.8939000070095062
```

На рисунке 1 изображены графики потерь и точности.

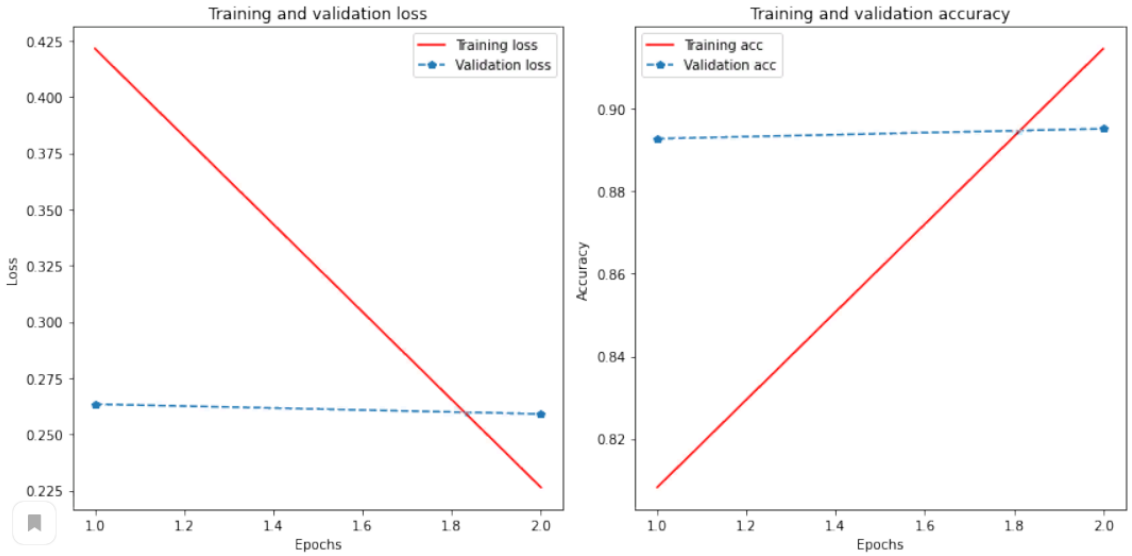

Рис. 1 - Графики потерь и точности

Уменьшим размер вектора представления текста до 2000.

```
Epoch 1/2
80/80 [==============================] - 1s 9ms/step - loss: 0.5941 - accuracy: 0.6545 - val_loss: 0.3080 - val_accuracy: 0.8708
Epoch 2/2
80/80 [==============================] - 1s 6ms/step - loss: 0.3099 - accuracy: 0.8736 - val_loss: 0.2861 - val_accuracy: 0.8760
0.8734000027179718
```

Результаты ухудшились, точность понизилась до 87 %.

Уменьшим вектора до 1000.

```
Epoch 1/2
80/80 [==============================] - 1s 7ms/step - loss: 0.6382 - accuracy: 0.6179 - val_loss: 0.3594 - val_accuracy: 0.8491
Epoch 2/2
80/80 [==============================] - 0s 5ms/step - loss: 0.3745 - accuracy: 0.8416 - val_loss: 0.3228 - val_accuracy: 0.8581
0.8535999953746796
```

Точность ухудшилась еще до 85%. Из чего можно сделать вывод, что при меньшем размере вектора результаты ИНС ухудшаются.

Функция ввода пользовательского текста (`read_text_from_input`) представлена в исходном коде в Приложении А. Функция получает словарь с ключами словами и их индексами, далее удаляем все не буквы и не цифры с помощью регулярки. Заменяем слова на их индексы и оставляем только 10000 самых частых слов. Векторизируем текст функцией vectorize и предсказываем. Результат работы представлен на рис. 2.

```
Mean Girls is one of the most iconic movies of all time. Every line is iconic and is quoted all the time.
As a person who prefers movies from the early 2000's I might be a bit bias when I say that Mean Girls definitely
makes my top 10 favorite movies list.
[[9, 31, 7, 4, 91, 5590, 102, 7, 32, 58, 347, 9, 5590, 5, 9, 32, 4, 58, 6, 415, 37, 8800, 102,
 39, 4, 402, 238, 30, 6, 227, 7492, 54, 135, 15, 407, 166, 61, 350, 158, 514, 102, 1029]]
[[0.94613844]]
```

Рис. 2 – Результат предсказания на отзыве

Выводы

В ходе выполнения лабораторной работы была создана нейронная сеть для прогнозирования успеха фильма по обзору. Была реализована функция для ввода пользовательского текста. Был изучен один из способов представления текста для передачи в нейронную сеть.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД

```
import re
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import gridspec
import numpy as np
from keras.utils import to_categorical
from keras.layers import Dense, Dropout
from keras.models import Sequential
from keras.datasets import imdb

dim = 10000

(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)

print("Categories:", np.unique(targets))
print("Number of unique words:", len(np.unique(np.hstack(data))))
length = [len(i) for i in data]
print("Average Review length:", np.mean(length))
print("Standard Deviation:", round(np.std(length)))

print("Label:", targets[0])
print(data[0])

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

print(decoded)
```

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

print(type(data))
data = vectorize(data)
targets = np.array(targets).astype("float16")

test_x = data[:10000]
test_y = targets[:10000]
train_x = data[10000:]
train_y = targets[10000:]

model = Sequential()
model.add(Dense(70, activation = "relu", input_shape=(dim, )))
model.add(Dropout(0.4))
model.add(Dense(40, activation = "relu"))
model.add(Dropout(0.3))
model.add(Dense(40, activation = "relu"))
model.add(Dense(1, activation = "sigmoid"))
model.summary()

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

def draws(H):
    loss = H.history['loss']
    val_loss = H.history['val_loss']
    acc = H.history['accuracy']
    val_acc = H.history['val_accuracy']
    epochs = range(1, len(loss) + 1)

    fig = plt.figure(figsize=(12, 6))
```

```
    gs = gridspec.GridSpec(1, 2, width_ratios=[3, 3])
    plt.subplot(gs[0])
    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'p--', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(gs[1])
    plt.plot(epochs, acc, 'r', label='Training acc')
    plt.plot(epochs, val_acc, 'p--', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.tight_layout()
    plt.show()

draws(results)

def read_text_from_input():
    dictionary = imdb.get_word_index()

    words = input()
    words = re.sub(r"[^\w']", " ", words).split(' ')

    valid = []
    for word in words:
        word = dictionary.get(word)
        if word in range(1, dim):
            valid.append(word+3)

    X = []
    X.append(valid)
    print(X)
```

```
    X = vectorize(X)
    result = model.predict(X)
    print(result)

read_text_from_input()
```