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. 8382 ____ Ефимова М.А

Преподаватель ____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

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

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

Порядок выполнения работы.

1. Ознакомиться с задачей регрессии;
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).

Ход работы.

Классификация – один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование класса), к которому относится данный объект.

IMDb – набор данных для оценки моделей машинного обучения по задаче классификации отзывов к фильмам на положительные и отрицательные, опираясь на текст отзывов.

Набор данных представляет собой множество из 50000 самых разных отзывов к кинолентам в интернет-базе фильмов (Internet Movie Database). Набор разбит на 25000 обучающих и 25000 контрольных отзывов, каждый набор на 50% состоит из отрицательных и на 50% из положительных отзывов.

Положительные отзывы помечаются, как единицы, а отрицательные – нули. Классификация отзывов к фильмам – это пример бинарной классификации.

Была построена нейронная сеть, разработанный код представлен в приложении А.

Архитектура сети: оптимизатор «Adam, batch_size = 500, nb_epoch = 2, функция потерь «binary_crossentropy». Модель нейронной сети представлена на рис. 1.

```
# Создание модели
model = Sequential()
model.add(Dense(50, activation="relu", input_shape=(10000,)))

model.add(Dropout(0.2, noise_shape=None, seed=None))
model.add(Dense(50, activation="linear", kernel_regularizer=regularizers.l2()))
model.add(Dropout(0.5, noise_shape=None, seed=None))
model.add(Dense(100, activation="relu", kernel_regularizer=regularizers.l2()))
model.add(Dense(1, activation="sigmoid"))

# Инициализация параметров обучения
model.compile(Adam(), loss='binary_crossentropy', metrics=['accuracy'])

# Обучение сети
hist = model.fit(train_x, train_y, batch_size=500, epochs=2, validation_data=(test_x, test_y))
```

Рисунок 1 – Модель нейронной сети.

Графики потери и точности обучения текущей архитектуры приведены на рис. 2 - 3 соответственно.

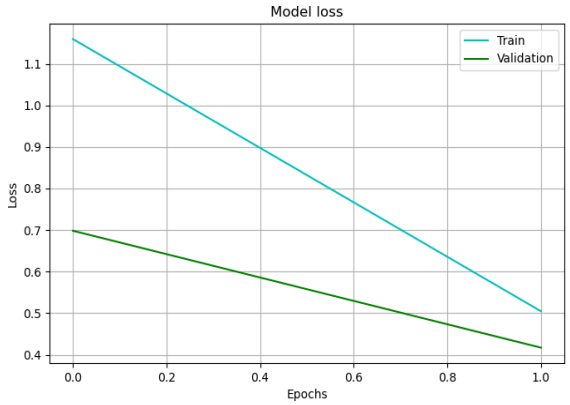

Рисунок 2 – График потери во время обучения модели.

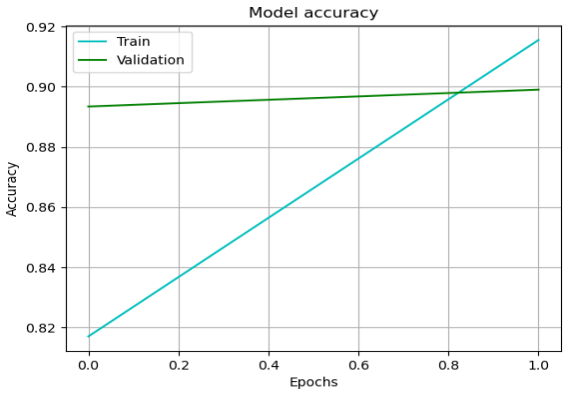

Рисунок 3 – График точности во время обучения модели.

Точность обучения модели равна 0.8963 при заданных параметрах архитектуры.

```
313/313 [==============================] - 1s 3ms/step - loss: 0.4240 - accuracy: 0.8963
----------------------------------------------------------------
Loss & Accuracy:  [0.4240337014198303, 0.8963000178337097]
```

Рисунок 4 – Точность обучения модели

Исследуем влияние длины вектора представления данных на результат обучения. Результаты показаны на рис. 5 - 10.

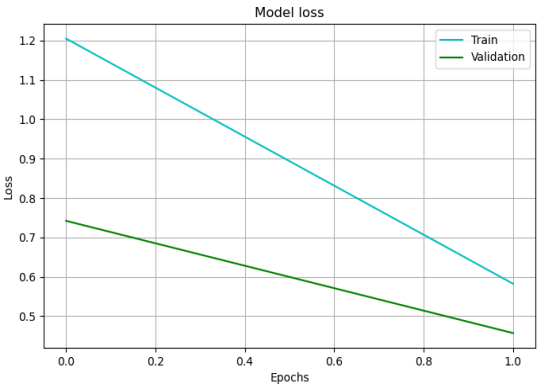

Рисунок 5 – График потери при длине вектора 1000.

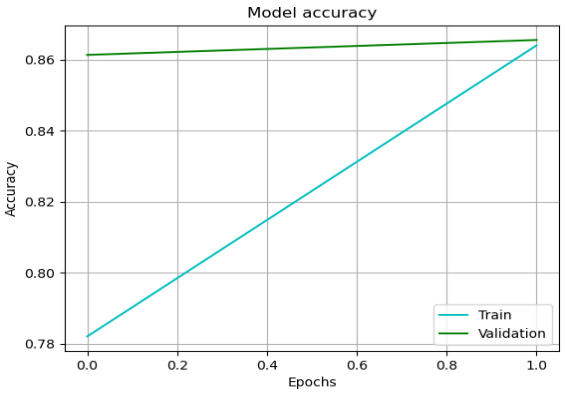

Рисунок 6 – График точности длине вектора 1000.

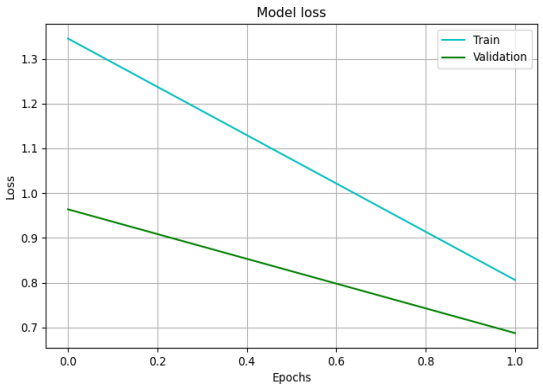

Рисунок 7 – График потери при длине вектора 100.

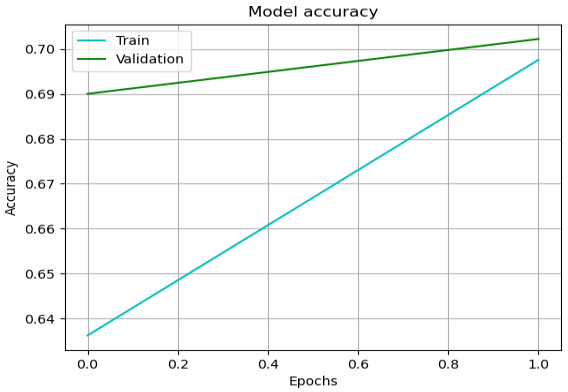

Рисунок 8 – График точности при длине вектора 100.

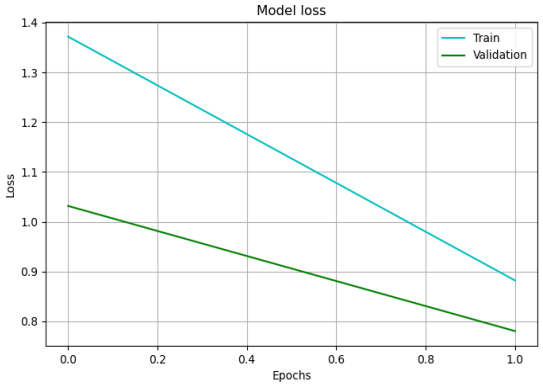

Рисунок 9 – График потери при длине вектора 10.

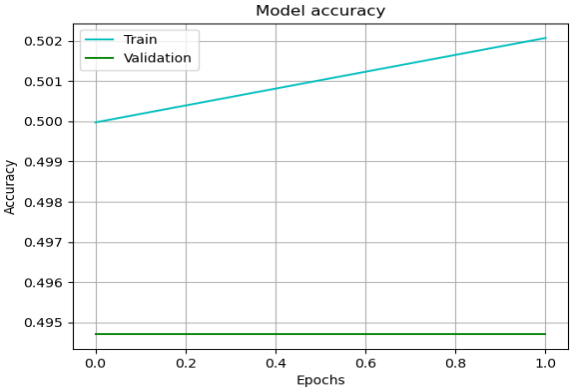

Рисунок 10 – График точности придлине вектора 10.

Из графиков видно, что при уменьшении размера вектора точность падает (0. 8610, 0.7011 и 0.4923 соответственно).

Это связано с тем, что мы убираем часть слов из рецензии и оставляем только наиболее часто встречающиеся. Из-за этого становится сложнее определить настроение обзора, поэтому нейронная сеть не может их точно классифицировать.

Была написана функция загрузки пользовательского текста `text_load`. Код представлен на рис. 11.

```
def text_load():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    for string in reviews:
        words = string.replace(',', ' ').replace('.', ' ').replace('?', ' ').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = word.lower()
            word = dictionary.get(word)
            num_words.append(word)
        test_x.append(num_words)
    #print(test_x)
    return test_x

    index = imdb.get_word_index()
    reverse_index = dict([(value, key) for (key, value) in index.items()])
    decoded = " ".join([reverse_index.get(i, "#") for i in test_x[1]])
    print(decoded)

reviews = text_load()
for j, i in enumerate(reviews):
    for index, value in enumerate(i):
        if value is None or value > 10000:
            reviews[j][index] = 0
```

Рисунок 11 – Функция загрузки пользовательского текста.

Функция `text_load` возвращает преобразованные строки, каждая строка-обзор представляет собой массив индексов слов в IMDb. Пользовательский датасет показан на рис. 12.

```
reviews = [
    "The best movie I have ever seen!!!",
    "I like this movie! Nice plot.",
    "Nice but very boring movie. I wouldn't like to watch it one more time.",
    "It really is horribly inert, and every time Downey opens his mouth to say something unintelligible, the film dies a bit more.",
    "I do not advise watching this film, it is disgusting."
]
```

Рисунок 12 – Пользовательские обзоры.

Была проверена работа функции загрузки пользовательского текста. Точность прогнозирования по комментарию «The best movie I have ever seen!!!» составляет 0.5017196, что говорит о том, что отзыв положительный. А настроение комментария «I do not advise watching this film, it is disgusting.» сеть оценила на 0.23739597, что соответствует отрицательному окрасу обзора.

График точности оценки фильма при прогоне через пользовательский датасет из пяти отзывов представлен на рис. 13. Точность составила приблизительно 70%, то есть нейронная сеть правильно классифицировала четыре из пяти отзыва.

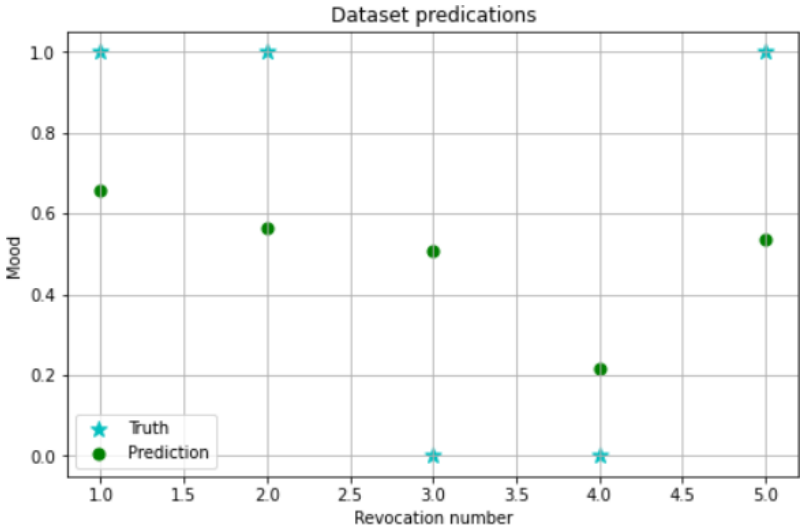

Рисунок 13 – График точности классификации отзывов к фильмам.

Выводы.

В ходе выполнения лабораторной работы была построена сеть, прогнозирующая оценку фильма по обзорам. Было рассмотрено преобразование текста к виду, с которым может работать нейронная сеть. Также было выявлено, что при уменьшении размера вектора представления текста точность падает, это связано с тем, что мы убираем часть слов из словаря. Наибольшей точностью (0.8963) обладает сеть с размером вектора 10000. Была написана функции загрузки пользовательского текста.

ПРИЛОЖЕНИЕ А

```
# Подключение модулей

import numpy as np                                          # Общие
математические и числовые операции

import matplotlib.pyplot as plt                             # Библиотека для
визуализации графиков

from keras.datasets import imdb                             # Internet Movie
Database

from keras.optimizers import Adam

from keras.layers import Dense, Dropout

from keras import Sequential, regularizers


# Загрузка набора данных IMDb, который содержит 50 000 отзывов к
кинолентам

(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)


data = np.concatenate((training_data, testing_data), axis=0)

targets = np.concatenate((training_targets, testing_targets), axis=0)


# Изучение датасета

print("--------------------------------")

print("Категории:", np.unique(targets))

print("Количество уникальных слов:", len(np.unique(np.hstack(data))))

length = [len(i) for i in data]

print("Средний размер обзора:", np.mean(length))

print("Стандартное отклонение:", round(np.std(length)))

print("--------------------------------")

print("Настроение обзора:", targets[8])

print(data[8])

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[8]])
print(decoded)
print("---------------------------------")

# Подготовка данных
def vectorize(sequences, dimension=10000):
    # Каждый элемент входных данных нашей нейронной сети
    # должен иметь одинаковый размер
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results

reviews = [
    "The best movie I have ever seen!!!",
    "I like this movie! Nice plot.",
    "Nice but very boring movie. I wouldn't like to watch it one more
time.",
    "It really is horribly inert, and every time Downey opens his mouth
to say something unintelligible, the film dies a bit more.",
    "I do not advise watching this film, it is disgusting."
]

review_mood = [1.0, 1.0, 0.0, 0.0, 0.0]

def text_load():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    for string in reviews:
        words = string.replace(',', ' ').replace('.', ' ').replace('?', '
```

```
').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = word.lower()
            word = dictionary.get(word)
            num_words.append(word)
        test_x.append(num_words)
    #print(test_x)
    return test_x

    index = imdb.get_word_index()
    reverse_index = dict([(value, key) for (key, value) in
index.items()])
    decoded = " ".join([reverse_index.get(i, "#") for i in test_x[1]])
    print(decoded)

reviews = text_load()
for j, i in enumerate(reviews):
    for index, value in enumerate(i):
        if value is None or value > 10000:
            reviews[j][index] = 0

# Векторизация обзоров
data = vectorize(data)
targets = np.array(targets).astype("float32")
review_mood = np.asarray(review_mood).astype("float32")

# Разделим датасет на обучающий и тестировочный наборы
test_x = data[:10000]                                  # \ тестировочный
набор
test_y = targets[:10000]                               # / состоит из 10
```

```
000

train_x = data[10000:]                                  # \ обучающий набор
train_y = targets[10000:]                               # / состоит из 40
000

# Создание модели
model = Sequential()
model.add(Dense(50, activation="relu", input_shape=(10000,)))

model.add(Dropout(0.2, noise_shape=None, seed=None))
model.add(Dense(50, activation="linear",
kernel_regularizer=regularizers.l2()))
model.add(Dropout(0.5, noise_shape=None, seed=None))
model.add(Dense(100, activation="relu",
kernel_regularizer=regularizers.l2()))
model.add(Dense(1, activation="sigmoid"))

# Инициализация параметров обучения
model.compile(Adam(), loss='binary_crossentropy', metrics=['accuracy'])

# Обучение сети
hist = model.fit(train_x, train_y, batch_size=500, epochs=2,
validation_data=(test_x, test_y))

# Построение графика ошибки
history_dict = hist.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
plt.figure(1, figsize=(8, 5))
plt.plot(loss_values, 'c', label='Train')
```

```
plt.plot(val_loss_values, 'g', label='Validation')
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend()
plt.grid()
plt.show()

# Построение графика точности
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(acc_values, 'c', label='Train')
plt.plot(val_acc_values, 'g', label='Validation')
plt.title('Model accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()

result = model.evaluate(test_x, test_y)
print("-------------------------------------------------------------")
print('Loss & Accuracy: ', result)

reviews = vectorize(reviews)

custom_loss, custom_acc = model.evaluate(reviews, review_mood)
print('Accuracy:', custom_acc)
prediction = model.predict(reviews)
plt.figure(3, figsize=(8, 5))
```

```
plt.title("Dataset predications")

plt.scatter([1, 2, 3, 4, 5], review_mood, marker='*', c='c', s=100,
label='Truth')

plt.scatter([1, 2, 3, 4, 5], prediction, c='g', s=50, label='Prediction')

plt.xlabel('Revocation number')

plt.ylabel('Mood')

plt.legend()

plt.grid()

plt.show()

print("--------------------------------")

print(prediction)
```