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

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Искусственные нейронные сети»

Тема: Классификация обзоров фильмов

Студент гр. 7383 _____ Зуев Д. В.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы:

Реализовать прогнозирование успеха фильмов по обзорам (Predict Sentiment From Movie Reviews)

Задачи.

1. Ознакомиться с рекуррентными нейронными сетями
2. Изучить способы классификации текста
3. Ознакомиться с ансамблированием сетей
4. Построить ансамбль сетей, который позволит получать точность не менее 97%

Ход работы.

1. Были созданы и обучены две модели искусственной нейронной сети, решающей задачу определения настроения обзора. Первая нейронная сеть рекуррентная с добавлением полносвязных слоев и слоев разреживания. Её архитектура представлена на рис. 1.

Модель представлена на рис. 1.

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model.add(LSTM(100))

model.add(Dropout(0.3, noise_shape=None, seed=None))
model.add(Dense(50, activation="relu"))
model.add(Dropout(0.2, noise_shape=None, seed=None))

model.add(Dense(1, activation="sigmoid"))
```

Рисунок 1 – Модель первой сети

Вторая нейронная сеть рекуррентная с добавлением слоя свертки. Её архитектура представлена на рис. 2.

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))

model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

model.add(Conv1D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))

model.add(LSTM(100))
model.add(Dropout(0.3))

model.add(Dense(1, activation='sigmoid'))
```

Рисунок 2 - Модель второй сети

При обучении моделей использовался оптимизатор Adam и функция потерь бинарная кросс энтропия. Результаты обучения первой сети представлены на рис. 3-4. Результаты обучения второй сети представлены на рис. 5-6.

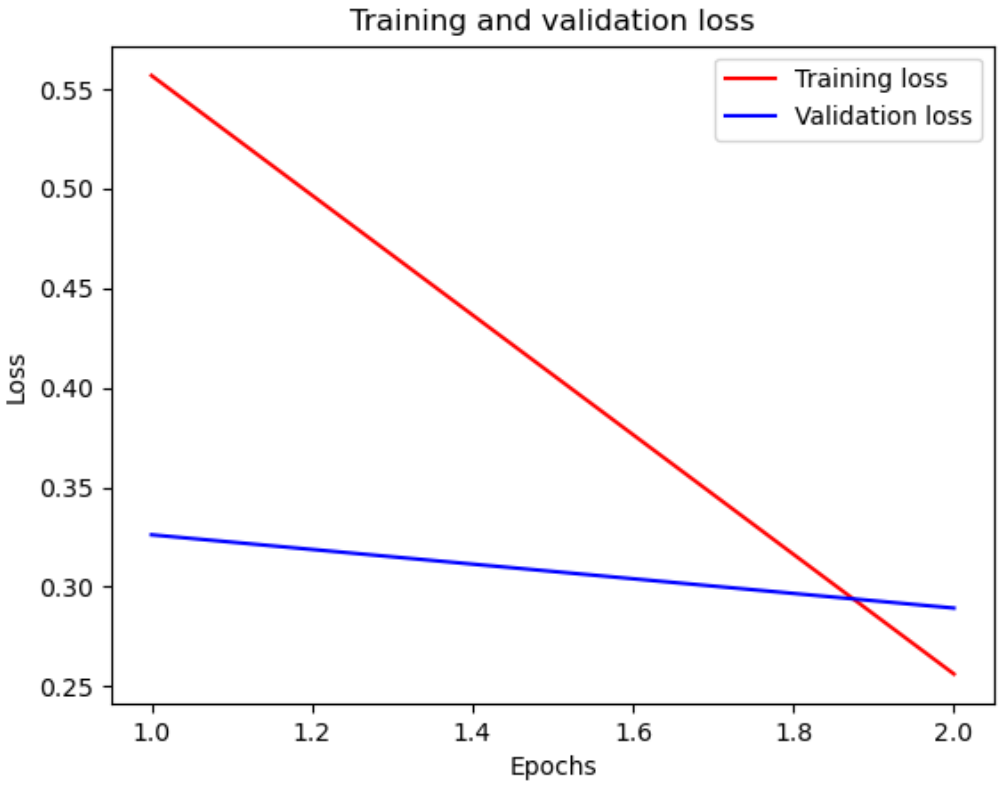

Рисунок 3 - Потери первой сети

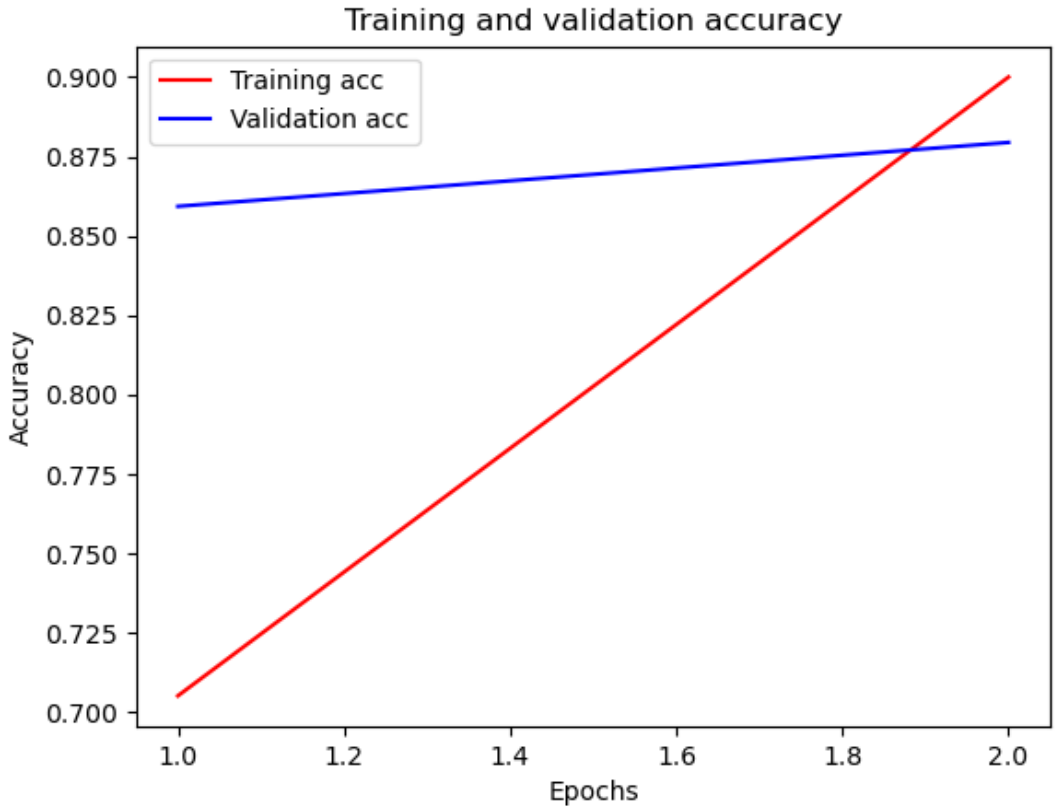

Рисунок 4 - Точность первой сети

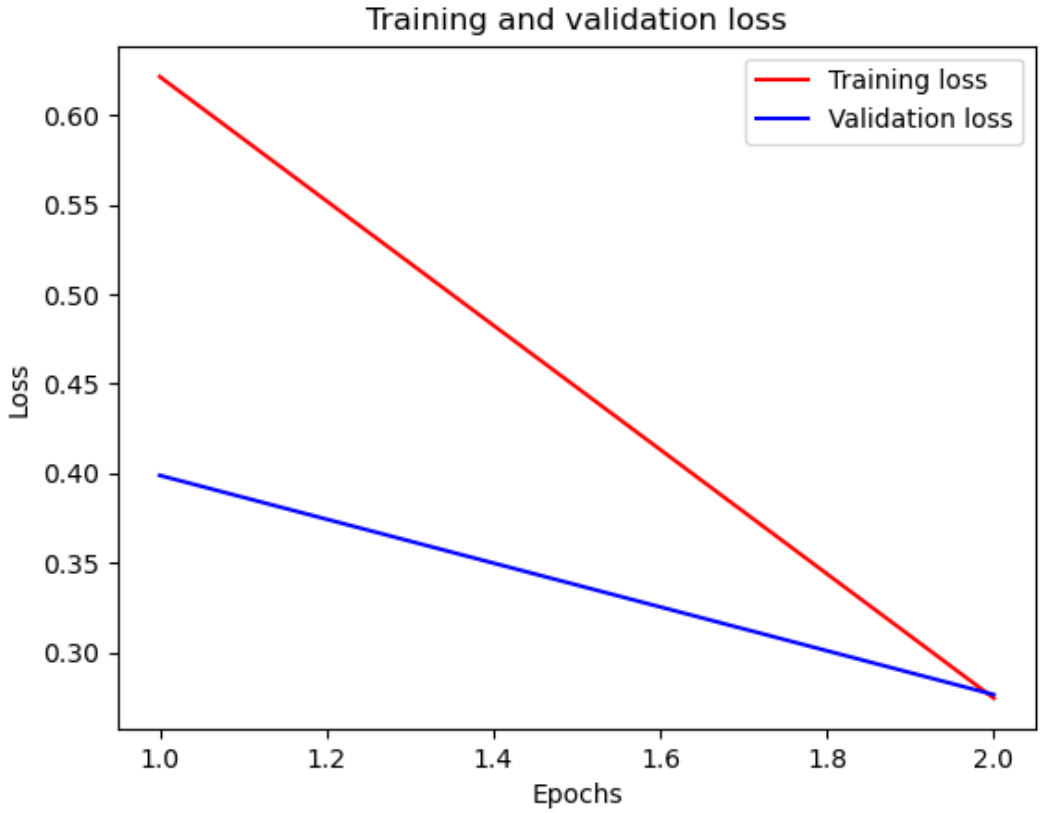

Рисунок 5 - Потери второй сети

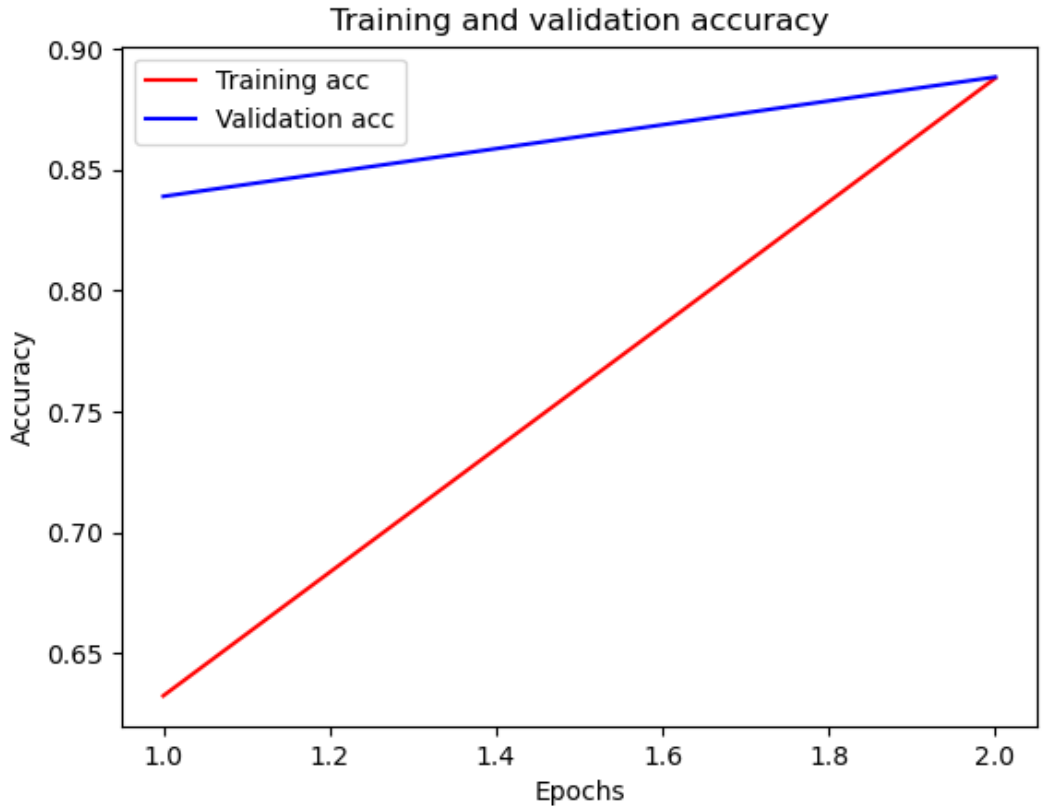

Рисунок 6 - Точность второй сети

2. Для ансамблирования моделей была написана функция ensembling_models, объединение результатов работы сетей происходило по принципу среднего арифметического результатов обработки каждого обзора. Результат работы ансамблирования сетей представлен на рис. 7

```
Accuracy of ensembling models is 0.88724
```

Рисунок 7 - Ансаблирование сетей

3. Была написана функция `test_my_text` для загрузки пользовательского текста и прогнозирования успеха фильма по этому тексту. Точность прогнозирования обзоров моделями обоих сетей по отдельности и ансамблированием этих моделей представлена на рис. 8.

```
Validation accuracy of 1st model is 1.0
Validation accuracy of 2nd model is 0.8333333134651184
Validation accuracy of ensembling models is 1.0
```

Рисунок 8 - Точность на пользовательском тексте

Как видно по точности, первая сеть показывает лучшие результаты прогнозирования, в отличие от точности моделей на тестовых образах, где все наоборот. Так же видно, что точность ансамблирования модели получилась достаточно высокой.

Результаты прогнозирования ансамблированием моделей представлены на рис. 9.

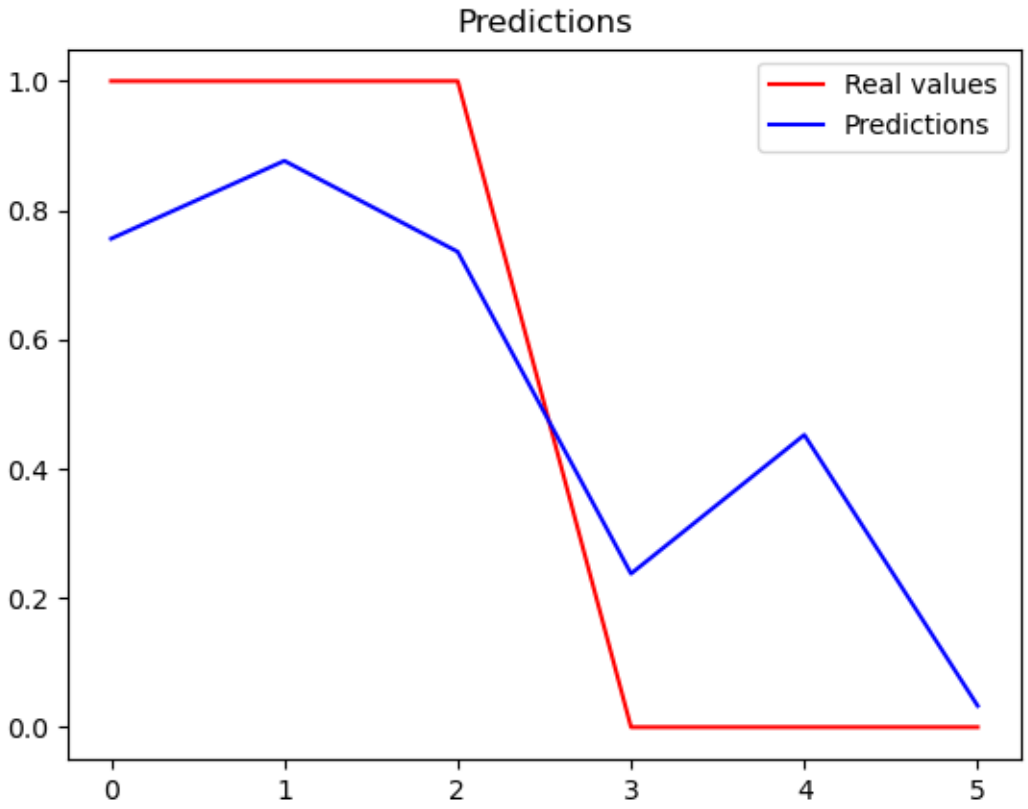

Рисунок 9 - Соответствие реальной оценки и предсказанной сетью

Выводы:

В ходе выполнения данной лабораторной работы были построены модели сетей, прогнозирующих оценку фильма по обзорам, и проведено ансамблирование этих моделей. Также была написана функция прогнозирования оценки по пользовательскому тексту с помощью ансамблированных моделей.

ПРИЛОЖЕНИЕ А

```
import numpy as np
from keras.datasets import imdb
from keras.models import Sequential, load_model, Input
from keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
from keras.datasets import imdb
import matplotlib.pyplot as plt


strings = ["I've never seen anything like it. This film is really shocking and causes a storm of emotions!"
           "I definitely recommend watching this wonderful movie.",
           "Despite the simplicity of the script, the film was shot qualitatively, and the actors played at the "
           "highest level. I really liked this movie.",
           "The film was shot superbly. It's like being on the scene.",
           "A bad movie gives out a lot of things, from an absolutely meaningless script to unbearable graphics.",
           "This Director holds the bar for a terrible comedian. This Comedy does not cause any emotions other than "
           "disgust.",
           "The characters in this film have absolutely no motivation, their actions are devoid of any meaning. "
           "I hope the Director of this film will not make any more films."]


values = [1, 1, 1, 0, 0, 0]
max_review_length = 500
top_words = 10000
embedding_vector_length = 32



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

```
    training_data      =      sequence.pad_sequences(training_data,
maxlen=max_review_length)
    testing_data       =       sequence.pad_sequences(testing_data,
maxlen=max_review_length)
    return    (training_data,    training_targets),    (testing_data,
testing_targets)



def build_model_1():
    model = Sequential()
    model.add(Embedding(top_words,         embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(100))


    model.add(Dropout(0.3, noise_shape=None, seed=None))
    model.add(Dense(50, activation="relu"))
    model.add(Dropout(0.2, noise_shape=None, seed=None))


    model.add(Dense(1, activation="sigmoid"))
    model.compile(loss='binary_crossentropy',     optimizer='adam',
metrics=['accuracy'])
    return model



def build_model_2():
    model = Sequential()
    model.add(Embedding(top_words,         embedding_vector_length,
input_length=max_review_length))


    model.add(Conv1D(filters=32,   kernel_size=3,   padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))


    model.add(Conv1D(filters=64,   kernel_size=3,   padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.4))
```

```
    model.add(LSTM(100))
    model.add(Dropout(0.3))


    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',     optimizer='adam',
metrics=['accuracy'])
    return model



def draw_plot(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)
    print(len(loss))
    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()
    plt.plot(epochs, acc, 'r', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()



def train_models():
    (training_data,        training_targets),        (testing_data,
testing_targets) = load_data()
    model1 = build_model_1()
    model2 = build_model_2()
```

```
    H = model1.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=2,
                   batch_size=256)
    scores = model1.evaluate(testing_data, testing_targets,
verbose=0)
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    model1.save('model1.h5')
    draw_plot(H)
    H = model2.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=2,
                   batch_size=256)
    scores = model2.evaluate(testing_data, testing_targets,
verbose=0)
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    model2.save('model2.h5')
    draw_plot(H)


def ensembling_models():
    (_, _), (testing_data, testing_targets) = load_data()
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    predictions1 = model1.predict(testing_data)
    predictions2 = model2.predict(testing_data)
    predictions = np.divide(np.add(predictions1, predictions2), 2)
    testing_targets = np.reshape(testing_targets, (25000, 1))
    predictions = np.greater_equal(predictions, np.array([0.5]))
    predictions = np.logical_not(np.logical_xor(predictions,
testing_targets))
    acc = predictions.mean()
    print("Accuracy of ensembling models is %s" % acc)


def test_my_text():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(values).astype("float32")
    for string in strings:
        string = string.lower()
```

```
        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            =            sequence.pad_sequences(test_x,
maxlen=max_review_length)
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")

    predictions1 = model1.predict(test_x)
    predictions2 = model2.predict(test_x)
    predictions = np.divide(np.add(predictions1, predictions2), 2)

    plt.title("Predictions")
    plt.plot(test_y, 'r', label='Real values')
    plt.plot(predictions, 'b', label='Predictions')
    plt.legend()
    plt.show()
    plt.clf()

    predictions = np.greater_equal(predictions, np.array([0.5]))
    test_y = np.reshape(test_y, (6, 1))
    predictions    =    np.logical_not(np.logical_xor(predictions,
test_y))
    _, acc1 = model1.evaluate(test_x, test_y)
    _, acc2 = model2.evaluate(test_x, test_y)
    print("Validation accuracy of 1st model is %s" % acc1)
    print("Validation accuracy of 2nd model is %s" % acc2)
    acc = predictions.mean()
    print("Validation accuracy of ensembling models is %s" % acc)



train_models()
ensembling_models()
```

```
test_my_text()
```