**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: Прогноз успеха фильма по обзорам**

Студентка гр. 7382 _____ Петрова А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

**Цель работы.**

Классификация последовательностей - это проблема прогнозирующего моделирования, когда у вас есть некоторая последовательность входных данных в пространстве или времени, и задача состоит в том, чтобы предсказать категорию для последовательности.

**Задачи**

Ознакомиться с рекуррентными нейронными сетями

Изучить способы классификации текста

Ознакомиться с ансамблированием сетей

Построить ансамбль сетей, который позволит получать точность не менее 97%

**Требования к выполнению задания.**

1. Найти набор оптимальных ИНС для классификации текста
2. Провести ансамблирование моделей
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
4. Провести тестирование сетей на своих текстах

**Ход работы.**

Были созданы две модели нейронной сети — рекуррентная и рекуррентная сверточная сеть.

Данные потерь и точности представлены на рис. 1-4.

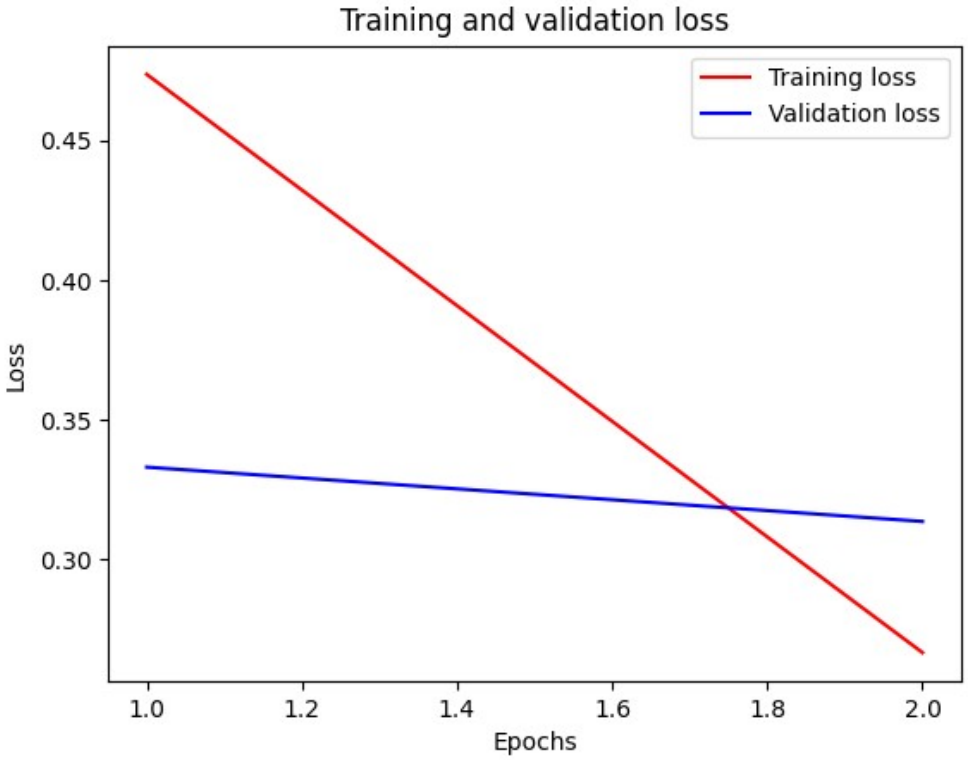


Рисунок 1 — График потерь для первой модели

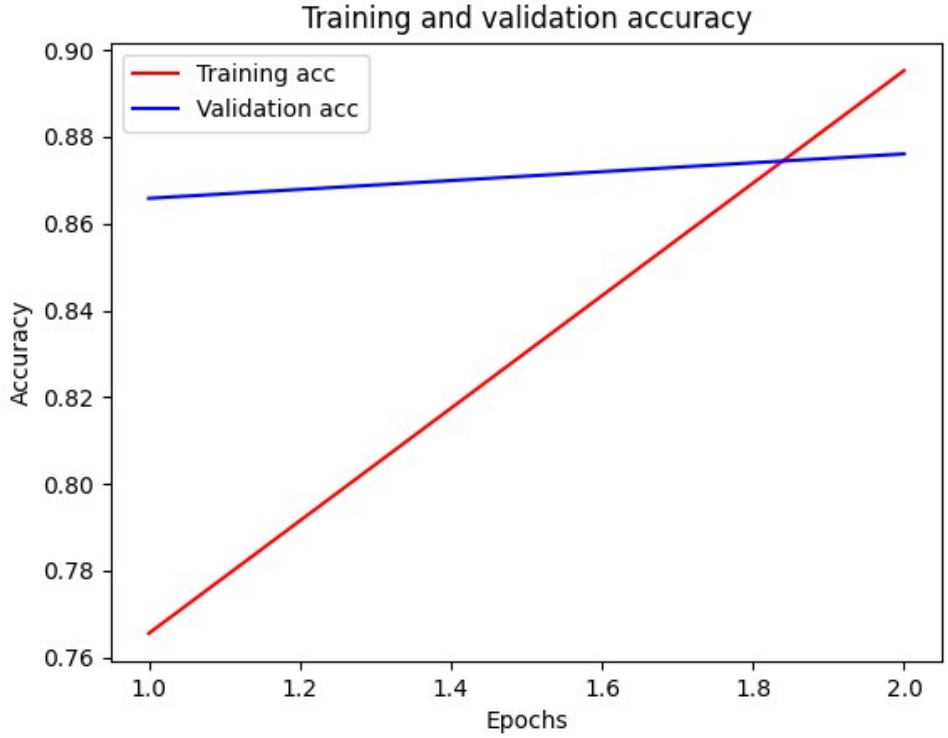


Рисунок 2 — График точности для первой модели

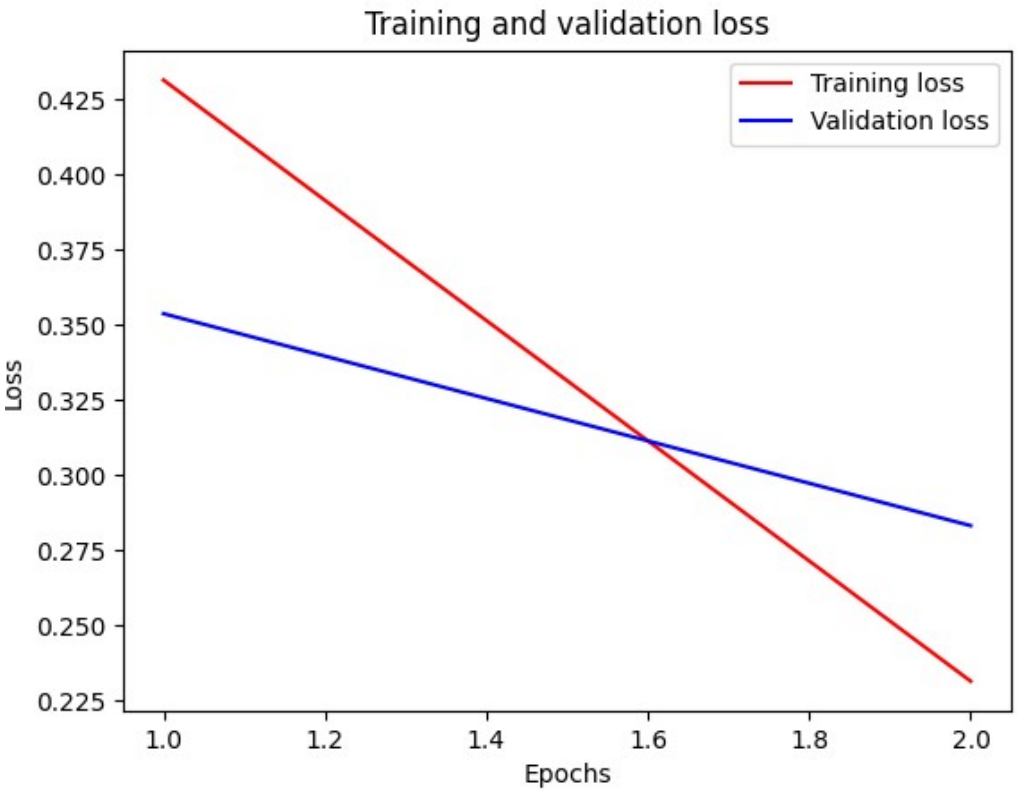


Рисунок 3 — График потерь для второй модели

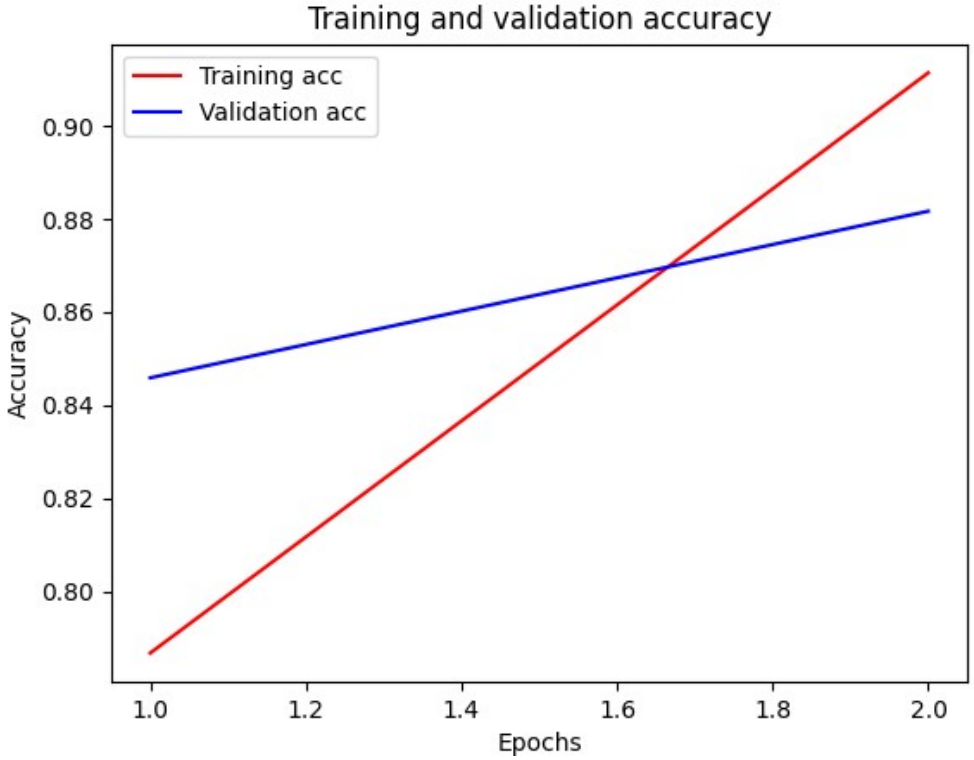


Рисунок 4 — График точности для второй модели

После этого было проведено ансамблирование моделей, как среднее арифметическое предсказаний. Точность — 95,3%. Полученный ансамбль был проверен на 2-х обзорах.

1)Very good film! Like it. Really fantastic!

2)Bad bad bad! Poor scenario, mean acting!

Первый обзор был оценен 0.709356, второй 0.364009, что соответствует действительности. Таким образом ансамбль работает корректно.

**Выводы.**

В ходе выполнения лабораторной работы были созданы две модели нейронных сетей для предсказания успеха фильма по обзорам. Было проведено ансамблирование сетей для улучшения точности предсказания. Были написаны функции, с помощью которых было проведено тестирование модели на пользовательских данных.

## ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential, load_model

from tensorflow.keras.layers import Dense, LSTM, Embedding, Dropout, Conv1D, MaxPooling1D
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.datasets import imdb
import matplotlib.pyplot as plt

(X_train, Y_train), (X_test, Y_test) = imdb.load_data(num_words=10000)
data = np.concatenate((X_train, Y_test), axis=0)
targets = np.concatenate((Y_train, Y_test), axis=0)
max_review_length = 500
top_words = 10000
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)

embedding_vector_length = 32


def build_models():
    models = []
    model_1 = Sequential()
                    model_1.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
    model_1.add(LSTM(100))
    model_1.add(Dropout(0.3))
    model_1.add(Dense(64, activation='relu'))
    model_1.add(Dropout(0.4))
    model_1.add(Dense(1, activation='sigmoid'))
    models.append(model_1)

    model_2 = Sequential()
                    model_2.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
                model_2.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model_2.add(MaxPooling1D(pool_size=2))
    model_2.add(Dropout(0.3))
    model_2.add(LSTM(100))
```

```
    model_2.add(Dropout(0.3))
    model_2.add(Dense(1, activation='sigmoid'))
    models.append(model_2)
    return models


def fit_models(models):
    i = 1
    for model in models:
                        model.compile(loss='binary_crossentropy',    optimizer='adam',
metrics=['accuracy'])
             history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test),
epochs=2, batch_size=64)
        scores = model.evaluate(X_test, Y_test, verbose=0)
        model.save('model' + str(i) + '.h5')
        print("Accuracy: %.2f%%" % (scores[1] * 100))
        epochs = range(1, len(history.history['loss']) + 1)
        plt.plot(epochs, history.history['loss'], 'r', label='Training loss')
        plt.plot(epochs, history.history['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, history.history['accuracy'], 'r', label='Training acc')
        plt.plot(epochs, history.history['val_accuracy'], 'b', label='Validation acc')
        plt.title('Training and validation accuracy')
        plt.xlabel('Epochs')
        plt.ylabel('Accuracy')
        plt.legend()
        plt.show()
        i += 1


def ensembling():
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    predictions1 = model1.predict(X_train)
    predictions2 = model2.predict(X_train)
    predictions = np.divide(np.add(predictions1, predictions2), 2)
    targets = np.reshape(Y_train, (25000, 1))
    predictions = np.greater_equal(predictions, np.array([0.5]))
```

```
    predictions = np.logical_not(np.logical_xor(predictions, targets))
    acc = predictions.mean()
    print("Accuracy of ensembling  %s" % acc)


def load_text(filename):
    file = open(filename, 'rt')
    text = file.read()
    file.close()
    words = text.split()
    import string
    table = str.maketrans(", ", string.punctuation)
    stripped = [w.translate(table) for w in words]
    stripped_low = []
    for w in stripped:
        stripped_low.append(w.lower())
    print(stripped_low)
    indexes = imdb.get_word_index()
    encoded = []
    for w in stripped_low:
        if w in indexes and indexes[w] < 7500:
            encoded.append(indexes[w])
    data = np.array(encoded)
    test = sequence.pad_sequences([data], maxlen=max_review_length)
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    results = []
    results.append(model1.predict(test))
    results.append(model2.predict(test))
    print(results)
    result = np.array(results).mean(axis=0)
    print(result)


models = build_models()
fit_models(models)
ensembling()
load_text("text1.txt")
load_text("text2.txt")
```