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 _____ Александров Р.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.

Проблема усложняется тем, что последовательности могут различаться по длине, состоять из очень большого словарного запаса входных символов и могут потребовать от модели изучения долгосрочного контекста или зависимостей между символами во входной последовательности.

В данной лабораторной работе также будет использоваться датасет IMDb, однако обучение будет проводиться с помощью рекуррентной нейронной сети.

Постановка 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%

Требования.

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 (привести в отчете)

Выполнение работы.

В ходе работы были созданы две модели нейронной сети:

1. рекуррентная сеть;
2. рекуррентная сеть с добавлением слоя свертки.

Исходный код представлен в приложении А.

Результаты тестирования первой сети представлены на рис. 1-2, второй сети – на рис. 3-4.

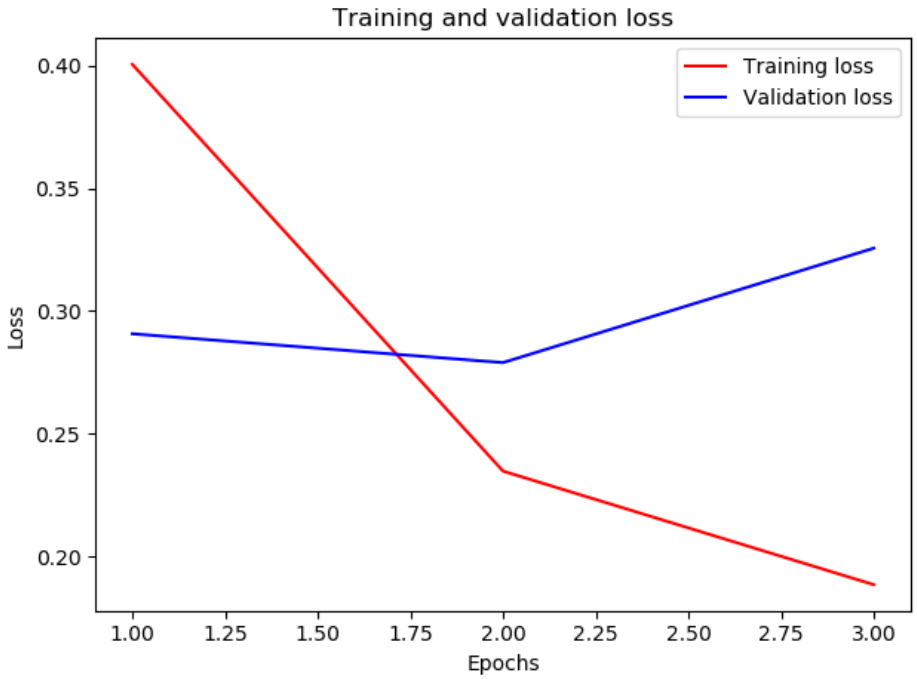

Рисунок 1 – Ошибки модели №1

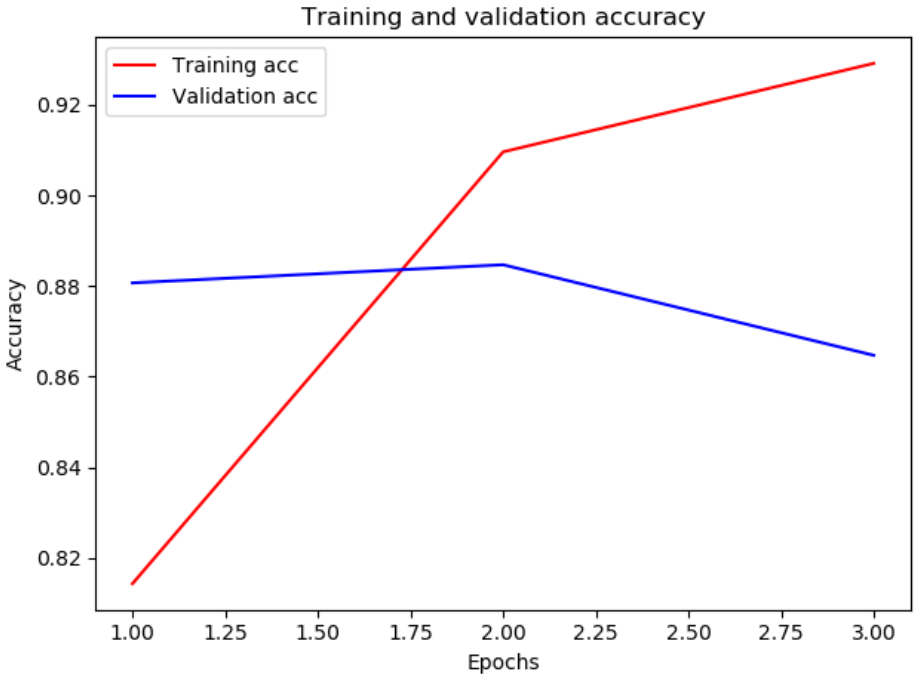

Рисунок 2 – Точность модели №1

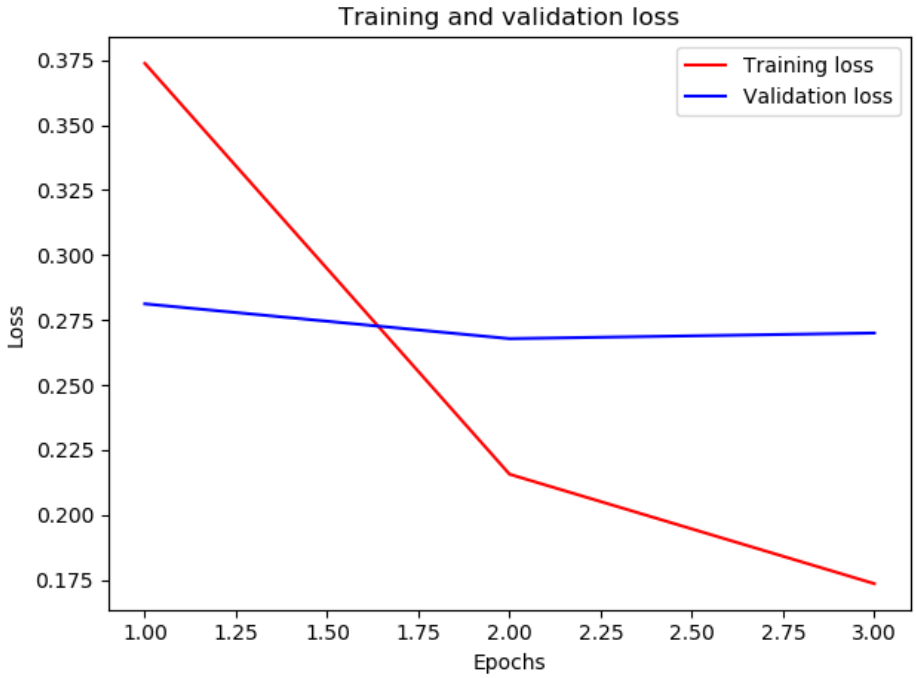

Рисунок 3 – Ошибки модели №2

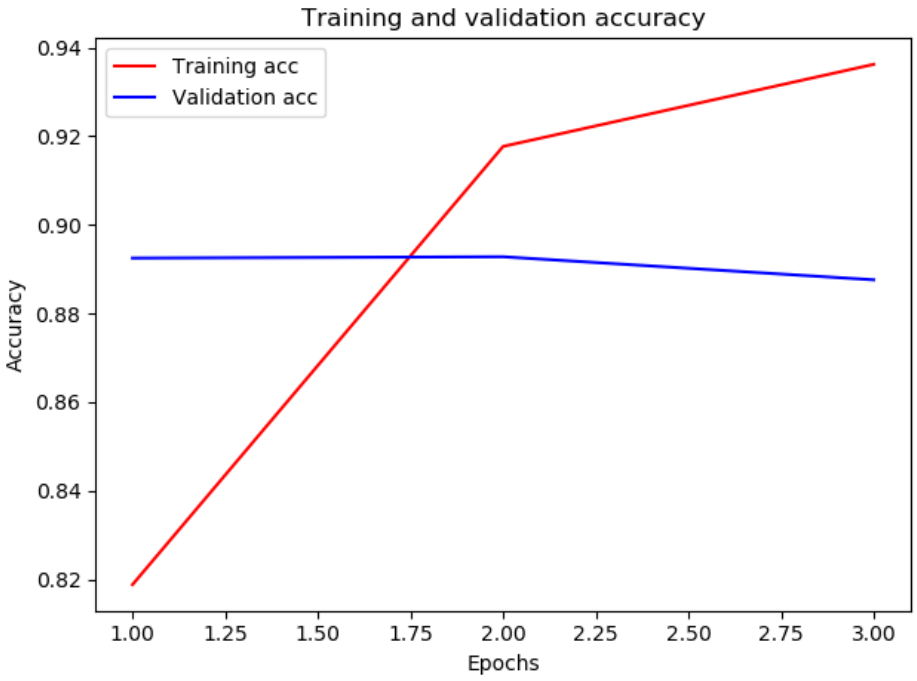

Рисунок 4 – Точность модели №2

Для ансамблирования моделей была написана функция `ensemble_models()`, результирующая точность ансамбля из двух моделей составляет 88.24%.

Для загрузки собственного текста была написана функция `get_user_text()`. Был протестирован отзыв "This is the worst film i ever seen", результат ансамбля равен 0.58.

Выводы.

В ходе работы были изучены рекуррентные нейронные сети, изучены способы классификации текста, проведено ансамблирование моделей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

top_words = 10000
embedding_vector_length = 32
max_review_length = 500

def load_dataset():
    (X_train, y_train), (X_test, y_test) =
imdb.load_data(num_words=10000)
    data = np.concatenate((X_train, X_test), axis=0)
    targets = np.concatenate((y_train, y_test), axis=0)
    data = sequence.pad_sequences(data, maxlen=max_review_length)
    targets = np.array(targets).astype("float32")
    test_x = data[:10000]
    test_y = targets[:10000]
    train_x = data[10000:]
    train_y = targets[10000:]

    return (train_x, train_y, test_x, test_y)

def build_model():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(100))
    model.add(Dropout(0.2))
    model.add(Dense(50, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

    return model
```

```
def build_model2():
    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(Dropout(0.2))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.2))
    model.add(LSTM(100))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',     optimizer='adam',
metrics=['accuracy'])

    return model

def fit_model(model):
    X_train, y_train, X_test, y_test = load_dataset()

    return  model.fit(X_train,  y_train,  validation_data=(X_test,
y_test), epochs=3, batch_size=64)

def create_plot(history_dict):
    loss_values = history_dict['loss']
    val_loss_values = history_dict['val_loss']
    acc = history_dict['accuracy']
    val_acc = history_dict['val_accuracy']
    epochs = range(1, len(loss_values) + 1)

    plt.plot(epochs, loss_values, 'r', label='Training loss')
    plt.plot(epochs, val_loss_values, '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 build_models():
    model1 = build_model()
    model2 = build_model2()

    history1 = fit_model(model1)
    model1.save('model1.h5')
    create_plot(history1.history)

    history2 = fit_model(model2)
    model1.save('model2.h5')
    create_plot(history2.history)

def ensemble_models(models, X_test, y_test):
    predict = np.array([0])
    for i in range(0, len(models)):
        predict = np.add(predict,
np.array(models[i].predict(X_test)))
    print(predict / len(models))
    print(accuracy_score(y_test, (predict / len(models)).round(),
normalize=False) / 100)

X_train, y_train, X_test, y_test = load_dataset()
models = [load_model('model1.h5'), load_model('model2.h5')]
ensemble_models(models, X_test, y_test)

def get_user_text(review, models):
    text = review.split()
    dict = imdb.get_word_index()
    words_num = []
    for word in text:
        word = dict.get(word)
        if word is not None and word < 10000:
            words_num.append(word)
    text = sequence.pad_sequences([words_num],
maxlen=max_review_length)
    res = []
    for model in models:
        tmp = model.predict(text)
        res.append(tmp)
    print((res[0]+res[1])/len(models))

review = "This is the worst film i ever seen"
```