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

ОТЧЕТ

по лабораторной работе №7
по дисциплине «Искусственные нейронные сети»
Тема: «Классификация обзоров фильмов»

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург
2020

Цель работы.

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

Задачи.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей

Требования.

- Найти набор оптимальных ИНС для классификации текста
- Провести ансамблирование моделей
- Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
- Провести тестирование сетей на своих текстах (привести в отчете)

Ход работы.

1. Были созданы и обучены две модели искусственной нейронной сети, решающей задачу определения настроения обзора (код программы представлен в приложении А).

Архитектура первой сети:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(32, activation="relu"))
model.add(Dropout(0.4))
model.add(Dense(32, activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(1, activation="sigmoid"))
```

Это рекуррентная нейронная сеть с добавлением полносвязных слоев и

слоев разреживания.

Архитектура второй сети:

```
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(Dropout(0.3))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

Это рекуррентная нейронная сеть с добавлением слоя свертки.

При обучении моделей использовался оптимизатор Adam и функция потерь binary_crossentropy.

Результаты обучения первой сети приведены на рис. 1-2.

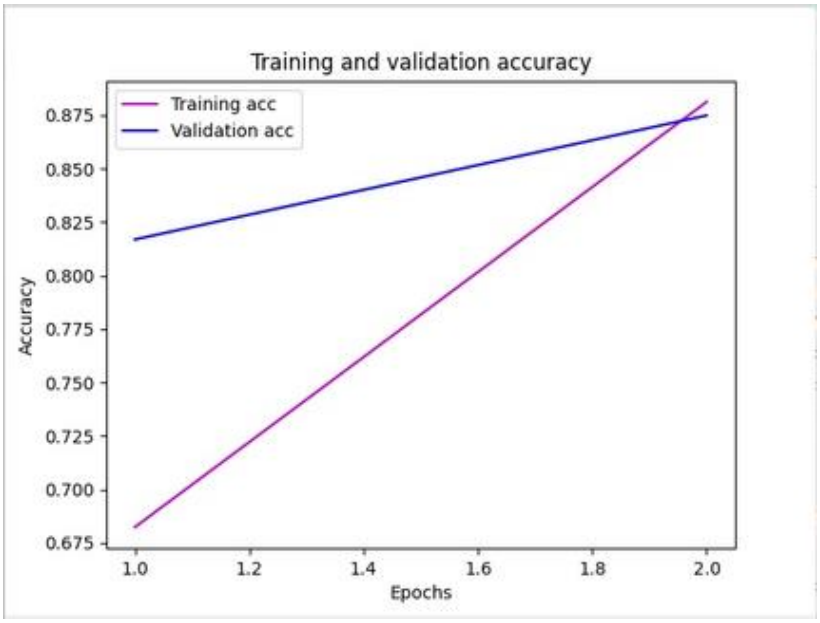

Рисунок 1 – График точности первой сети.

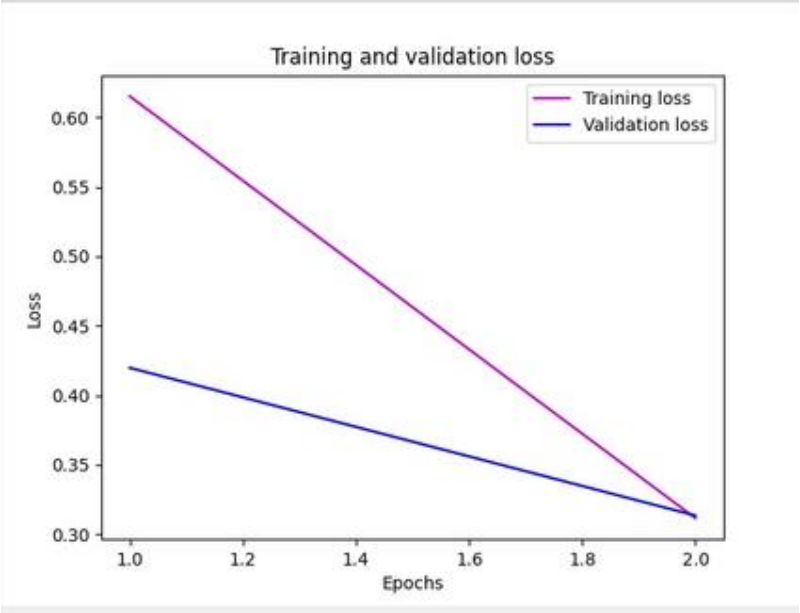

Рисунок 2 - График потерь первой сети.

Точность составила 87.48%.

Результаты обучения второй сети представлены на рис. 3-4.

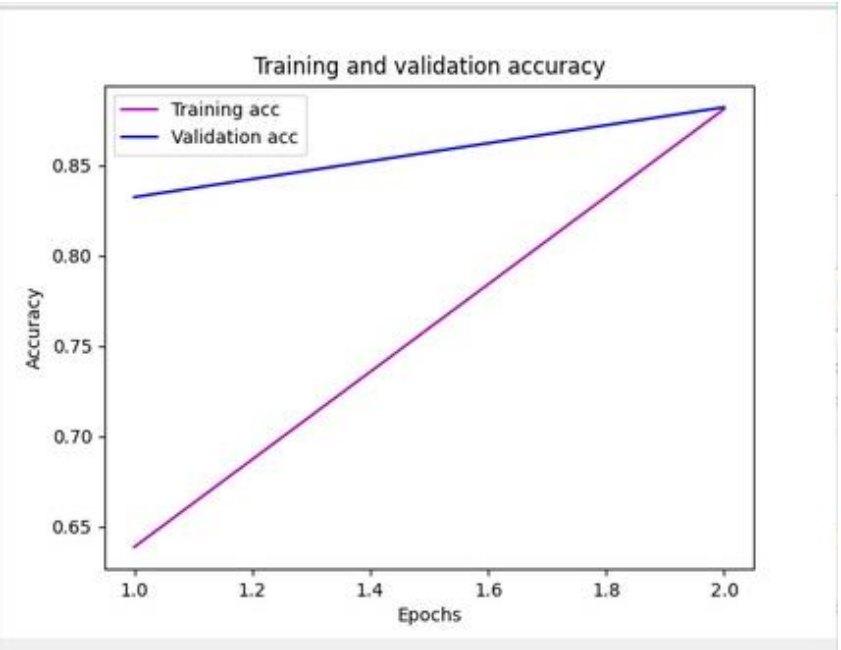

Рисунок 3 - График точности второй сети.

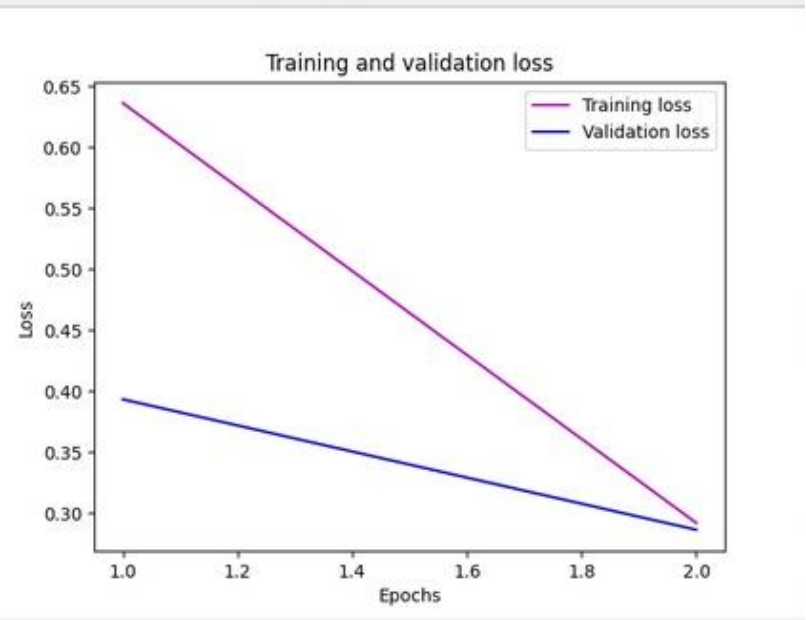

Рисунок 4 - График потерь второй сети.

Точность составила 88.21%.

2. Затем была написана функция для ансамблирования моделей ensembling_models. Объединение результатов работы сетей происходило по принципу среднего арифметического результатов обработки каждого обзора.

Точность ансамбля составила 88.34%.

3. Была также написана функция считывания пользовательского текста из файла. Протестируем работу ансамбля на примере отзыва: «Wonderful! This is the best movie that I've seen so far! I loved just everything! Amazing job!». Результат ансамбля – 0.8151, классификация отзыва правильна.

Вывод.

В ходе выполнения лабораторной работы были построены модели сетей, прогнозирующих оценку фильма по обзорам, и проведено ансамблирование этих моделей. Была также продемонстрирована работа ансамбля на пользовательском тексте.

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


filename = 'text.txt'
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(Dense(32, activation="relu"))
        model.add(Dropout(0.4))
        model.add(Dense(32, 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_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(Dropout(0.3))
        model.add(LSTM(100))
```

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


    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)
        loss_1, acc_1 = model1.evaluate(testing_data, testing_targets,
verbose=0)
        print("Accuracy: %.2f%%" % (acc_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)
        loss_2, acc_2 = model2.evaluate(testing_data, testing_targets,
verbose=0)
        print("Accuracy: %.2f%%" % (acc_2 * 100))
        model2.save('model2.h5')
        draw_plot(H)
```

```
    print("Accuracy of ensembling models is %s" % ((acc_1 + acc_2)/2))


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)
    plt.plot(epochs, acc, 'm', 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()
    plt.plot(epochs, loss, 'm', 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()


def test_text(filename):
```

```
        test_x = []
        with open(filename, 'r') as file:
            for line in file.readlines():
                line = line.lower()
                words = line.replace(',', ' ').replace('.', ' ').replace('?', ' ').replace('\n', ' ').split()
                num_words = []
                index = dict(imdb.get_word_index())
                for w in words:
                    w = index.get(w)
                    if w is not None and w < 10000:
                        num_words.append(w)
                test_x.append(num_words)
            print(test_x)
            test_x = sequence.pad_sequences(test_x, maxlen=max_review_length)
            model1 = load_model("model1.h5")
            model2 = load_model("model2.h5")
            prediction_1 = model1.predict(test_x)
            prediction_2 = model2.predict(test_x)

        print("Validation accuracy of 1st model is %s" % prediction_1)
        print("Validation accuracy of 2nd model is %s" % prediction_2)
        print('Validation accuracy of ensembling models is %.2f%%' % ((((prediction_1 + prediction_2)/2)*100)))
```

```
train_models()

test_text(filename)
```