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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Искусственные нейронные сети»

Тема: «Классификация обзоров фильмов»

Студент гр. 7383 _____ Рудоман В. 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.

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 сетей
3. Написать функцию, которая позволит загружать текст и получать результат ансамбля сетей.
4. Провести тестирование сетей на своих текстах

Ход работы.

Были созданы 2 модели нейронных сетей. Код программы приведен в приложении А. Модели были обучены на одинаковом наборе входных данных. Графики точности приведены ниже на рис. 1 и рис. 2 соответсвенно.

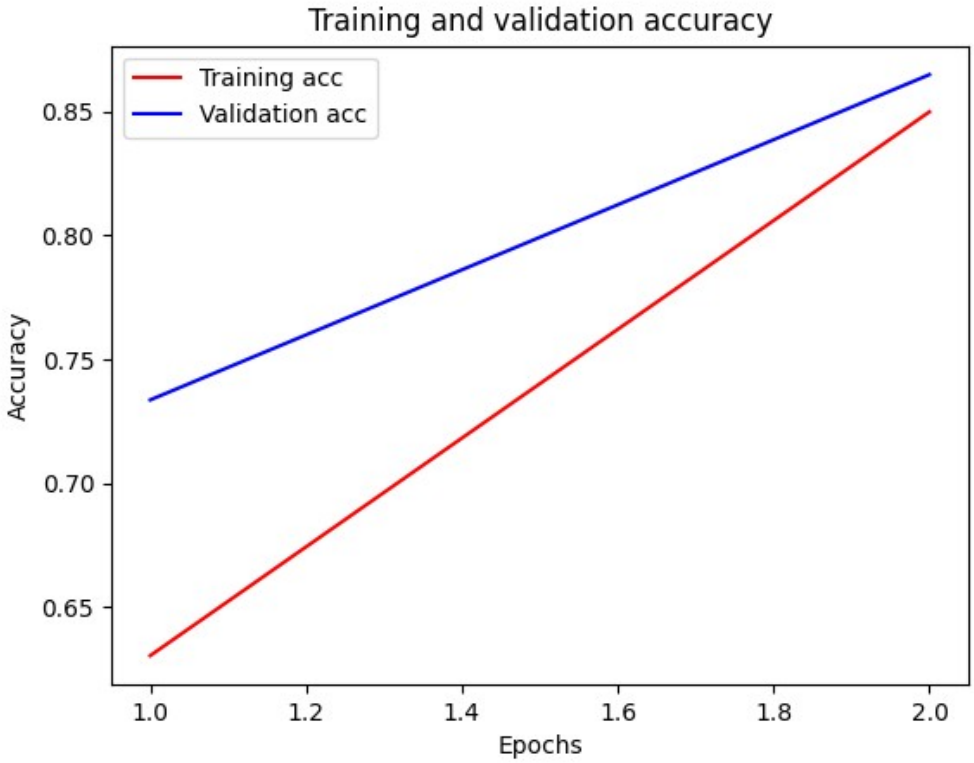

Рисунок 1 — График точности 1 модели

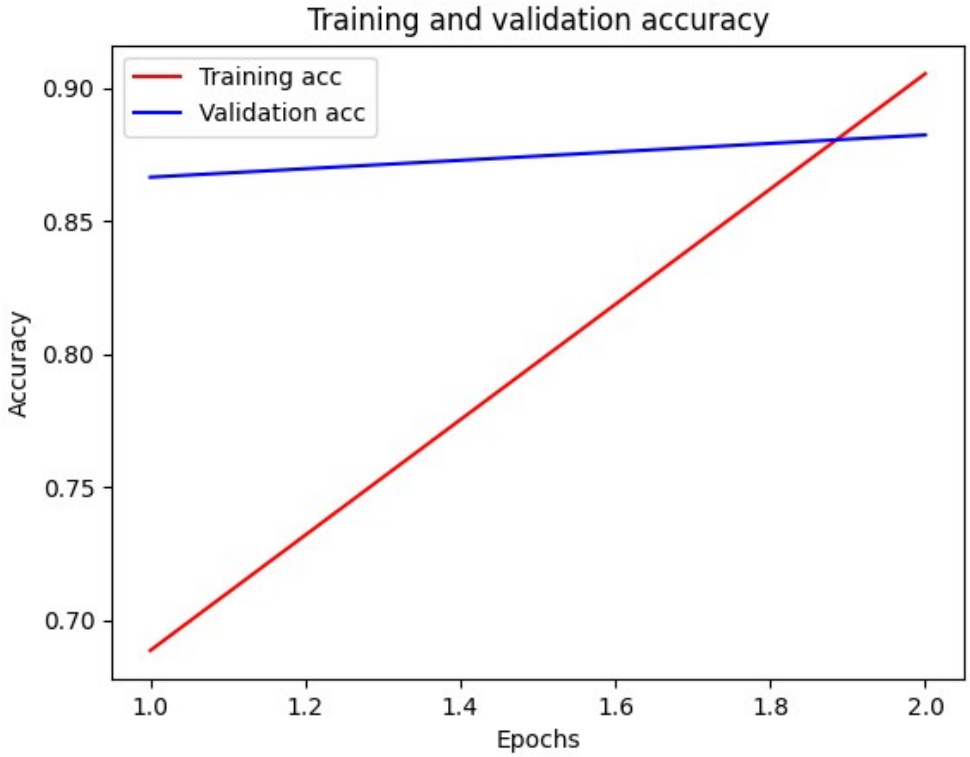

Рисунок 2 — График точности 2 модели

Ансамбль сетей показал точность около 0.88316.

Была реализована функция ввода пользовательского текста из файла. Она была протестирована на 2 отзывах.

One of the best films of this year. Great story, great actors.

Сеть оценила этот отзыв как положительный 85.21%

This is a terrible movie, I do not recommend watching it, disgusting plot.

Данный отзыв сеть оценила как негативный 66.61%

Вывод.

В результате выполнения лабораторной работы были созданы 2 модели нейронных сетей и проведено их ансамблирование. Была реализована функция оценки пользовательского отзыва на фильм.

ПРИЛОЖЕНИЕ А

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

from keras.datasets import imdb

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 encoding_text(file):
    separators = ".,:;!?()"
    txt = []
    with open(file, 'r') as text:
        f_text = text.read()
        f_text = f_text.replace('\n', ' ')
        txt = [word.strip(separators).lower() for word in
f_text.split(' ')]

    index = imdb.get_word_index()
    encoded = []
    for word in txt:
        if word in index and index[word] < top_words:
            encoded.append(index[word])
    encoded = sequence.pad_sequences([np.array(encoded)],
maxlen=max_review_length)
    return encoded
```

```
def prediction(encoded):
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")

    results = []
    results.append(model1.predict(encoded))
    results.append(model2.predict(encoded))
    result = np.array(results).mean(axis=0)
    result = np.reshape(result, result.shape[0])
    print("-"*50)
    print("It's a good movie at %.2f" %(result*100) + "%")
    print("-"*50)


#ensembling_models()
prediction(encoding_text("text.txt"))
```