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. 7381 _____ Трушников А.П.

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.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Ход работы.

1. Был найден набор оптимальных ИНС для классификации текста.

   Первая ИНС дает точность 90.75%. Архитектура сети представлена на рис.1. Графики точности и потерь представлены на рис.2-3.

```
optimizer = optimizers.RMSprop(lr=0.001, rho=0.9)
model = Sequential()
model.add(Embedding(max_features, embedding_vecor_length, input_length=max_review_length))
model.add(Dropout(0.3))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer, loss='binary_crossentropy', metrics=['accuracy'])
h = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=3, batch_size=64)
```

Рисунок 1 – Архитектура первой сети

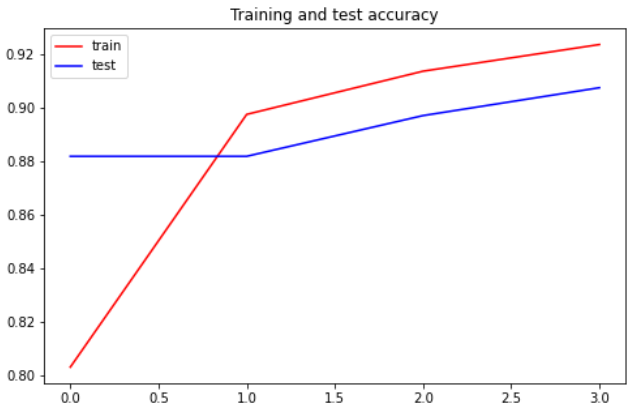

Рисунок 2 – График точности

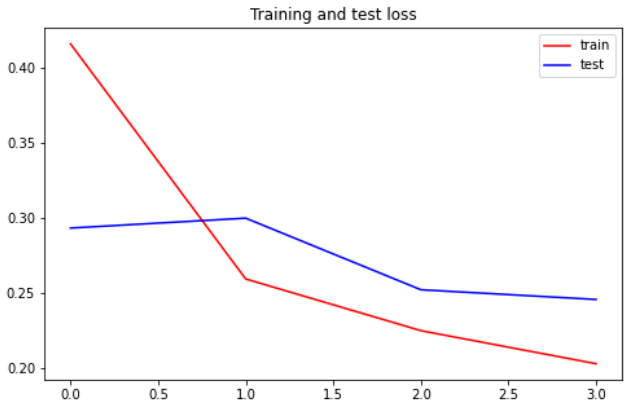

Рисунок 3 – График потерь

Вторая ИНС дает точность 91.03%. Архитектура сети представлена на рис.4. Графики точности и потерь представлены на рис.5-6.

```
optimizer = optimizers.RMSprop(lr=0.001, rho=0.9)
model = Sequential()
model.add(Embedding(max_features, embedding_vecor_length, input_length=max_review_length))
model.add(Dropout(0.3))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))
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(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(Dense(1, activation='sigmoid'))
model.compile(optimizer, loss='binary_crossentropy', metrics=['accuracy'])
h = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=3, batch_size=64)
```

Рисунок 4 – Архитектура второй сети

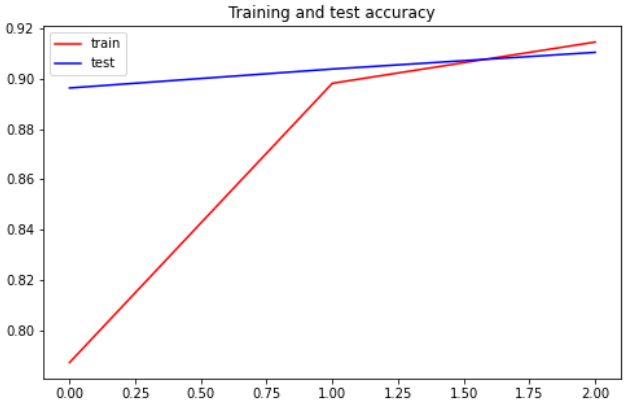

Рисунок 5 – График точности

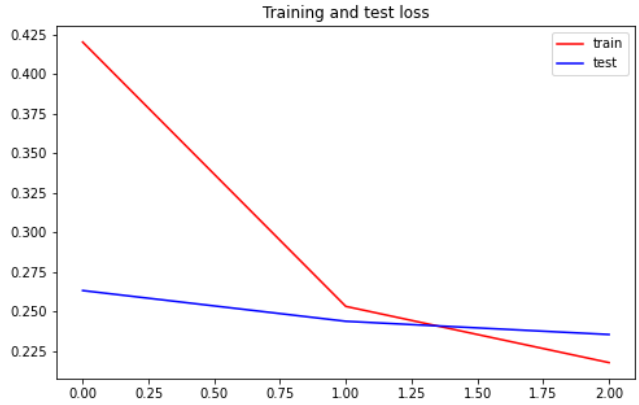

Рисунок 6 – График потерь

Третья ИНС дает точность 90.94%. Архитектура сети представлена на рис.7. Графики точности и потерь представлены на рис.8-9.

```
optimizer = optimizers.RMSprop(lr=0.001, rho=0.9)
model = Sequential()
model.add(Embedding(max_features, embedding_vecor_length, input_length=max_review_length))
model.add(Dropout(0.3))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))
model.add(GRU(64,return_sequences=True))
model.add(SimpleRNN(64))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer, loss='binary_crossentropy', metrics=['accuracy'])
h = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=4, batch_size=64)
```

Рисунок 7 – Архитектура третьей сети

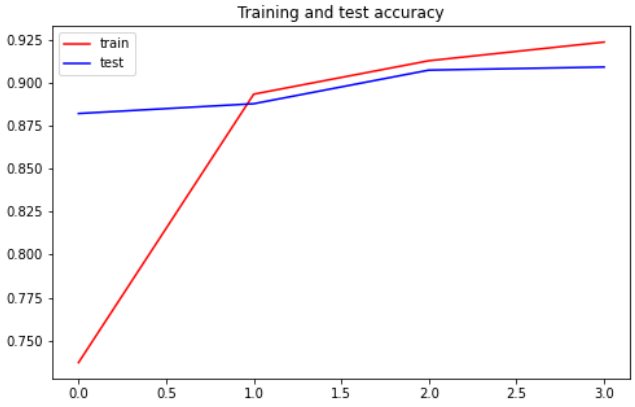

Рисунок 8 – График точности

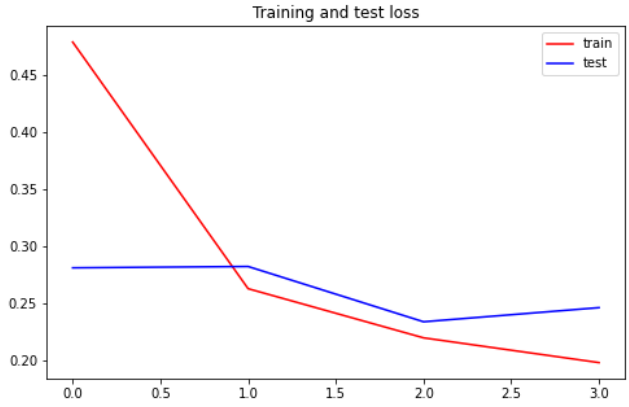

Рисунок 9 – График потерь

Четвертая ИНС дает точность 90.55%. Архитектура сети представлена на рис.10. Графики точности и потерь представлены на рис11-12.

```
optimizer = optimizers.RMSprop(lr=0.001, rho=0.9)
model = Sequential()
model.add(Embedding(max_features, embedding_vecor_length, input_length=max_review_length))
model.add(Dropout(0.3))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))
model.add(GRU(64, return_sequences=True))
model.add(Dropout(0.3))
model.add(GRU(32))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer, loss='binary_crossentropy', metrics=['accuracy'])
h = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=3, batch_size=64)
```

Рисунок 10 – Архитектура четвертой сети

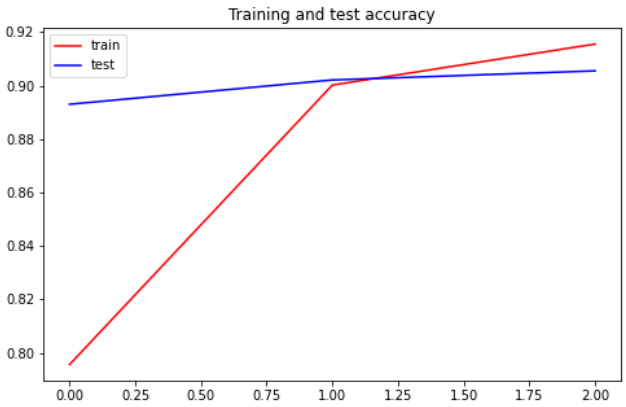

Рисунок 8 – График точности

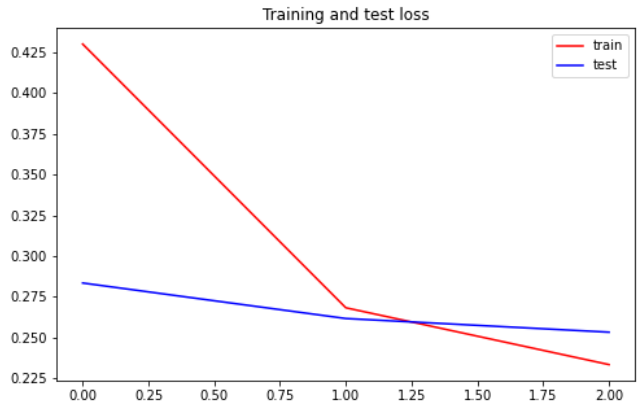

Рисунок 9 – График потерь

Ансамбль из этих четырех сетей дает точность 91.19% .

2. Была написана функция загрузки собственного текста prepInputText.

Проверим работу ансамбля на следующем тексте:

Cracking. Keeps attention from start to finish

Ансамбль в качестве ответа выдал 0.5028851.

Выводы.

В ходе выполнения работы познакомились с рекуррентными сетями. Были найдены оптимальные сети, из которых был построен ансамбль сетей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import numpy as np
import string
import matplotlib.pyplot as plt
from keras import optimizers
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import load_model
from keras.models import Sequential
from tensorflow.keras import regularizers
from keras.layers import Dropout, Dense, Embedding, GRU, Conv1D, MaxPooling1D,
SimpleRNN, Bidirectional
from keras.layers import Dense
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
from keras.datasets import imdb



def prepInputText(text, target):

    def genNum(data, dic):
        data =
data.translate(str.maketrans(dict.fromkeys(string.punctuation))).split()
        for i in range(len(data)):
            num = dic.get(data[i])
            if (num == None):
                data[i] = 0
            else:
                data[i] = num
        return data
    dic = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(target).astype("float32")
    for i in range(0, len(text)):
        test_x.append(genNum(text[i], dic))
    test_x = sequence.pad_sequences(test_x, maxlen=max_review_length)
    return test_x, test_y


def ensemble(models, test_x, test_y):
    preds = np.array(models[0].predict(test_x))
    models[0].evaluate(test_x, test_y, verbose=2)
    for i in range(1, len(models)):
        preds = preds + np.array(models[i].predict(test_x))
        models[i].evaluate(test_x, test_y, verbose=2)
    print(preds / len(models))
    print(accuracy_score(test_y, (preds / len(models)).round(),
normalize=False) / 100)


def getModel(max_review_length, embedding_vecor_length, max_features,
num_model):
```

```
    optimizer = optimizers.RMSprop(lr=0.001, rho=0.9)
    model = Sequential()
    model.add(Embedding(max_features, embedding_vecor_length,
input_length=max_review_length))
    model.add(Dropout(0.3))
    model.add(
        Conv1D(filters=32, kernel_size=3, padding='same', activation='relu',
kernel_regularizer=regularizers.l2(0.001)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.2))

    model.add(
        Conv1D(filters=32, kernel_size=3, padding='same', activation='relu',
kernel_regularizer=regularizers.l2(0.001)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.2))

    model.add(GRU(64, return_sequences=True))
    model.add(Dropout(0.3))
    model.add(GRU(32))

    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    h = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=3,
batch_size=64)
    scores = model.evaluate(test_x, test_y, verbose=0)
    accuracy = scores[1] * 100
    print("Accuracy: %.2f%%" % (accuracy))

    return model, accuracy, h


def best_bodel():
    num_model = 14
    accuracy = 0
    for i in range(10):
        print(i)
        model, acc, h = getModel(max_review_length, embedding_vecor_length,
max_features, num_model)
        if acc > accuracy:
            accuracy = acc
            model.save('model' + str(num_model) + '_' + str(round(accuracy, 2))
+ '%.h5')
            model.save_weights('weights' + str(num_model) + '.hdf5')
            plt.figure(1, figsize=(8, 5))
            plt.title("Training and test accuracy")
            plt.plot(h.history['acc'], 'r', label='train')
            plt.plot(h.history['val_acc'], 'b', label='test')
            plt.legend()
            plt.savefig('accuracy' + str(num_model) + '.png')
            # plt.show()
            plt.clf()

            plt.figure(1, figsize=(8, 5))
            plt.title("Training and test loss")
            plt.plot(h.history['loss'], 'r', label='train')
```

```
            plt.plot(h.history['val_loss'], 'b', label='test')
            plt.legend()
            plt.savefig('loss' + str(num_model) + '.png')
            # plt.show()
            plt.clf()


max_review_length = 500
embedding_vecor_length = 32
max_features = 10000

(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=max_features)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

data = sequence.pad_sequences(data, maxlen=max_review_length)
targets = np.array(targets).astype("float32")

train_x = data[max_features:]
train_y = targets[max_features:]

test_x = data[:max_features]
test_y = targets[:max_features]

# getModel(max_review_length, embedding_vecor_length, max_features)

models = []

models.append(load_model('model1\model1.h5'))
models.append(load_model('model2\model2.h5'))
models.append(load_model('model6\model6.h5'))
models.append(load_model('model7\model7.h5'))


data_x, data_y = prepInputText(["Cracking. Keeps attention from start to
finish"],[1])

ensemble(models, data_x, data_y  )

# best_bodel()
```