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. 7381 _____ Давкаева В.С.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цель

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%

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).

```
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, noise_shape=None, seed=None))
    model_1.add(Dense(50, activation='relu'))
    model_1.add(Dropout(0.2, noise_shape=None, seed=None))
    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(Conv1D(filters=64, kernel_size=3, padding='same', activation='relu'))
    model_2.add(MaxPooling1D(pool_size=2))
   model_2.add(Dropout(0.4))
    model_2.add(LSTM(100))
   model_2.add(Dropout(0.3))
    model_2.add(Dense(1, activation='sigmoid'))
   models.append(model_2)
    return models
```

Результаты обучения первой сети приведены на рис. 1. Точность составила 86,68%

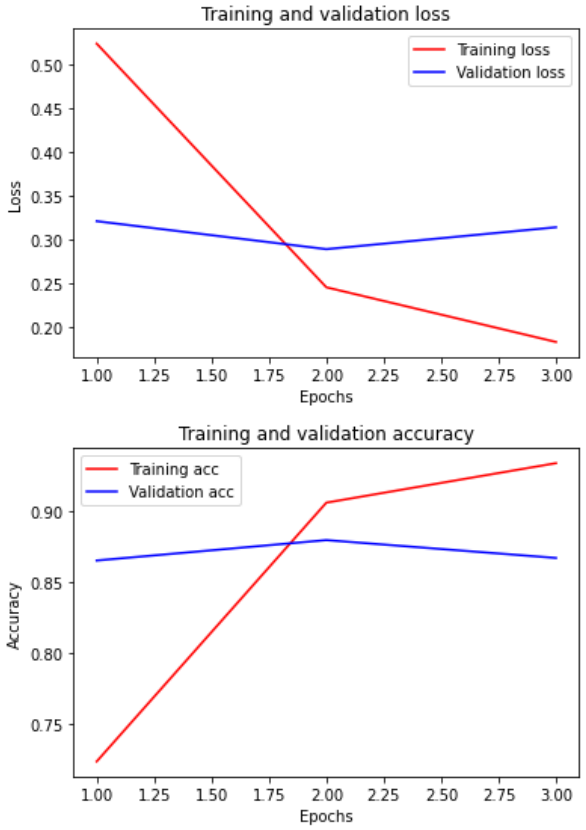

Рисунок 1 – График потерь и точности первой сети.

Результаты обучения второй сети представлены на рис. 2. Точность составила 88%

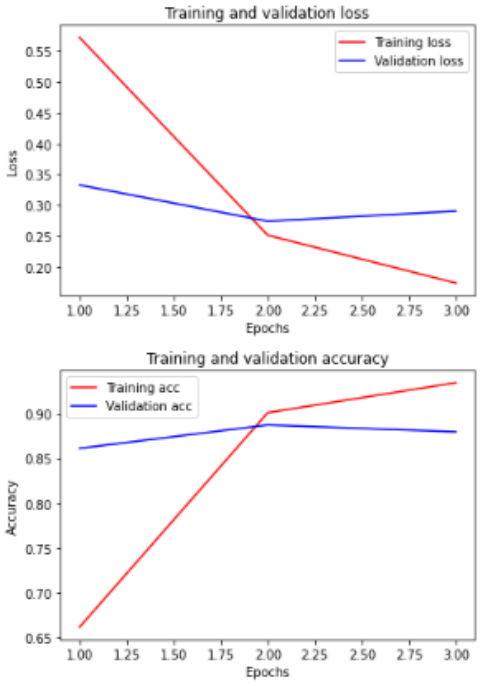

Рисунок 2 - График точности и потерь второй сети

2.Точность при ансамблировании составила 96.49
Полученый ансамбль был проверен на отзыве.
Отзыв: Fantastic! Perfect comedia! I like this film! Best movie i have ever seen! Recommend it for everyone
Обзор был оценен 0.7518927, что соответствует его положительности.
Ансамбль работает корректно.

Вывод.

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

```
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

top_words = 10000
embedding_vector_length = 32
max_review_length = 500

(training_data, training_targets), (testing_data, testing_targets) = imdb.load_
data(num_words=10000)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

training_data = sequence.pad_sequences(training_data, maxlen=max_review_length)
testing_data = sequence.pad_sequences(testing_data, maxlen=max_review_length)

def graphics(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 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, noise_shape=None, seed=None))
    model_1.add(Dense(50, activation='relu'))
    model_1.add(Dropout(0.2, noise_shape=None, seed=None))
    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='r
elu'))
    model_2.add(MaxPooling1D(pool_size=2))
    model_2.add(Dropout(0.3))
    model_2.add(Conv1D(filters=64, kernel_size=3, padding='same', activation='r
elu'))
    model_2.add(MaxPooling1D(pool_size=2))
    model_2.add(Dropout(0.4))
    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 train_models():
    models = build_models()
    i = 1
    for model in models:
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['a
ccuracy'])
        H=model.fit(training_data, training_targets, validation_data=(testing_d
ata, testing_targets), epochs=3,batch_size=256)
        scores = model.evaluate(testing_data, testing_targets, verbose=0)
        model.save('model' + str(i) + '.h5')
        print("Accuracy: %.2f%%" % (scores[1] * 100))
        graphics(H)
        i += 1

def ensembling():
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    predictions_1 = model1.predict(training_data)
    predictions_2 = model2.predict(training_data)
    predictions = np.divide(np.add(predictions_1, predictions_2), 2)
    targets = np.reshape(training_targets, (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 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")
        results = []
        results.append(model1.predict(test_x))
        results.append(model2.predict(test_x))
        result = np.array(results).mean(axis=0)
        print(result)


train_models()
ensembling()
test_text('text.txt')
```