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: Прогноз успеха фильма по обзорам

Студентка гр. 7383 _____ Прокопенко Н.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы.

Реализовать прогноз успеха фильма по обзорам.

Выполнение работы.

В ходе выполнения лабораторной работы, были созданы две модели нейронной сети. Первая модель является рекуррентной, вторая – рекуррентной сверточной. Код программы представлен в приложении А. После их обучения были достигнуты точности 85.84 % и 88.54% соответственно. Графики точности и потерь представлены на рис. 1-2.

Рисунок 1 – График потерь точности для рекуррентной модели

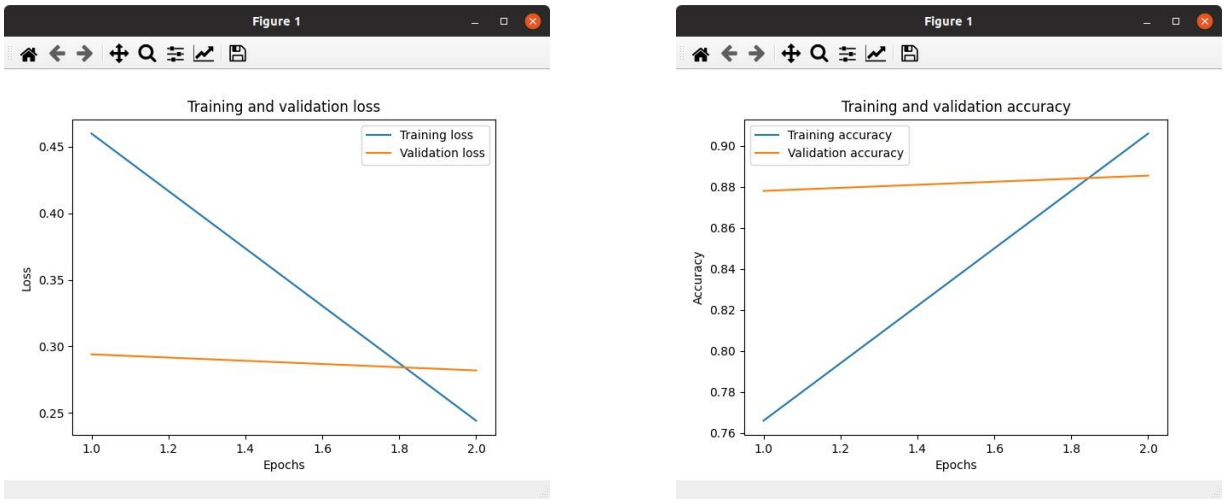

Рисунок 2 – График потерь и точности для рекуррентной сверточной модели

После было проведено ансамблирование двух сетей с помощью функции ensembling. Ансамблирование было выполнено как среднее арифметическое предсказаний тестовых данных. Точность предсказания ансамбля сетей 94.56%.

Была также написана функция считывания пользовательского текста из файла. Протестируем работу ансамбля на примере отзыва: «Very interesting movie! Great acting, exciting story.».

Результат ансамбля – 0.7151, классификация отзыва правильна.

Выводы.

В ходе выполнения данной работы были созданы две архитектуры сетей для предсказания успеха фильма по обзорам. Также было проведено ансамблирование сетей. Была создана функция для ввода пользовательского обзора и предсказания оценки фильма по нему.

ПРИЛОЖЕНИЕ А.
КОД ПРОГРАММЫ

```
import numpy as np

from tensorflow.keras.datasets import imdb

from tensorflow.keras.models import Sequential, load_model


from tensorflow.keras.layers import Dense, LSTM, Embedding, Dropout, Conv1D, MaxPooling1D

from tensorflow.keras.preprocessing import sequence

from tensorflow.keras.datasets import imdb

import matplotlib.pyplot as plt


(X_train, Y_train), (X_test, Y_test) = imdb.load_data(num_words=7500)

data = np.concatenate((X_train, Y_test), axis=0)

targets = np.concatenate((Y_train, Y_test), axis=0)

max_review_length = 500

top_words = 7500

X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)

X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)


embedding_vector_length = 32



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))
```

```
    model_1.add(Dense(64, activation='relu'))
    model_1.add(Dropout(0.4))
    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(LSTM(100))
    model_2.add(Dropout(0.3))
    model_2.add(Dense(1, activation='sigmoid'))
    models.append(model_2)
    return models



def fit_models(models):
    i = 1
    for model in models:
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=2, batch_size=64)
        scores = model.evaluate(X_test, Y_test, verbose=0)
        model.save('model' + str(i) + '.h5')
        print("Accuracy: %.2f%%" % (scores[1] * 100))
        epochs = range(1, len(history.history['loss']) + 1)
        plt.plot(epochs, history.history['loss'], label='Training loss')
```

```
        plt.plot(epochs,  history.history['val_loss'],  label='Validation
loss')
        plt.title('Training and validation loss')
        plt.xlabel('Epochs')
        plt.ylabel('Loss')
        plt.legend()
        plt.show()
        plt.clf()


        plt.plot(epochs,   history.history['accuracy'],   label='Training
accuracy')
        plt.plot(epochs,                  history.history['val_accuracy'],
label='Validation accuracy')
        plt.title('Training and validation accuracy')
        plt.xlabel('Epochs')
        plt.ylabel('Accuracy')
        plt.legend()
        plt.show()
        i += 1



def ensembling():
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    predictions1 = model1.predict(X_train)
    predictions2 = model2.predict(X_train)
    predictions = np.divide(np.add(predictions1, predictions2), 2)
    targets = np.reshape(Y_train, (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 load_text(filename):
    file = open(filename, 'rt')
    text = file.read()
    file.close()
    words = text.split()
    import string
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in words]
    stripped_low = []
    for w in stripped:
        stripped_low.append(w.lower())
    print(stripped_low)
    indexes = imdb.get_word_index()
    encoded = []
    for w in stripped_low:
        if w in indexes and indexes[w] < 7500:
            encoded.append(indexes[w])
    data = np.array(encoded)
    test = sequence.pad_sequences([data], maxlen=max_review_length)
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    results = []
    results.append(model1.predict(test))
    results.append(model2.predict(test))
    print(results)
    result = np.array(results).mean(axis=0)
    print(result)
```

```
models = build_models()
fit_models(models)
ensembling()
load_text("open.txt")
```