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

ОТЧЕТ

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

Студент гр. 8382 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

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.

Задание.

- 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)

Выполнение работы.

Работа выполнялась на базе операционной системы Windows 10 в среде разработки PyCharm и в онлайн сервисе Google Colab.

Модели.

В работе было использовано три модели:

1. Модель LTSM + Dense (Embedding → LSTM (100) → Dropout (0.4) → Dense (50, relu) → Dropout (0.2) → Dense (1, sigmoid)
2. Модель со сверткой и LSTM
    a. Embedding
    b. Conv1D(32 filters, kernel size 3, padding same, relu)
    c. MaxPooling1D (2)
    d. Dropout (0.3)
    e. Conv1D(64 filters, kernel size 3, padding same, relu)
    f. MaxPooling1D (2)
    g. Dropout (0.3)
    h. LSTM (150)
    i. Dense (1, sigmoid)
3. Модель на слоях LSTM (Embedding → LSTM (100) → Dense (1, sigmoid))

Параметры обучения всех моделей:

- Оптимизатор – adam
- Функция потерь – binary_crossentropy
- Метрика – точность
- Число эпох – 2
- Размер батча – 256
- Данных для валидации (в %) – 10

Обучение моделей.

Графики точности и потерь модели №1 LSTM + Dense во время обучения представлены на рис. 1, 2.

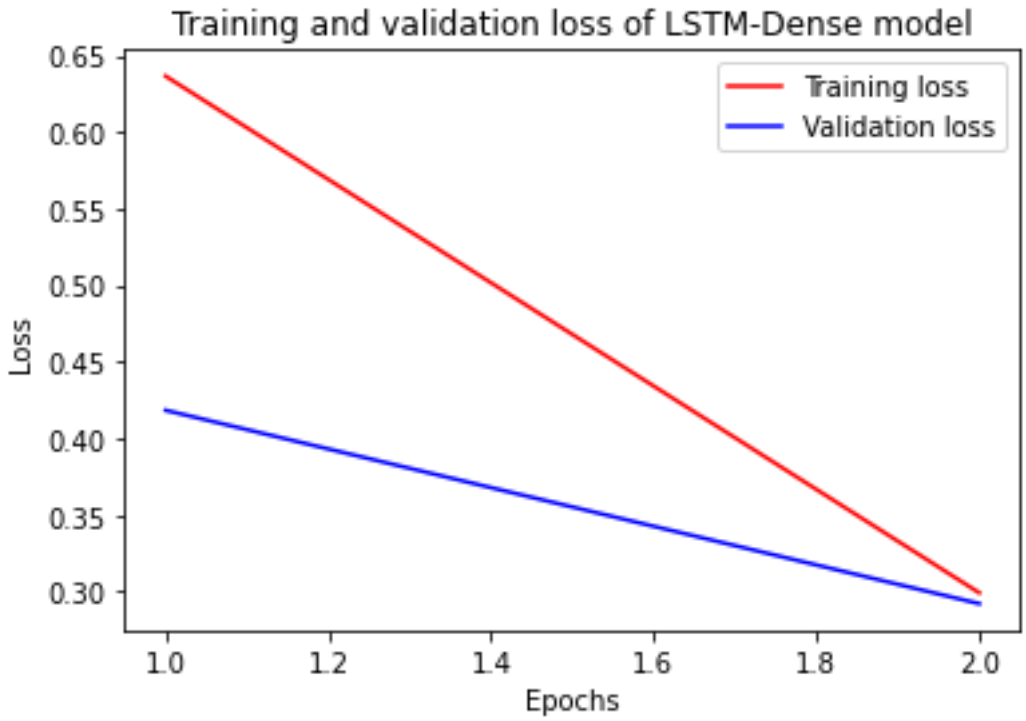

Рисунок 1 – Потери при обучении LSTM-Dense модели

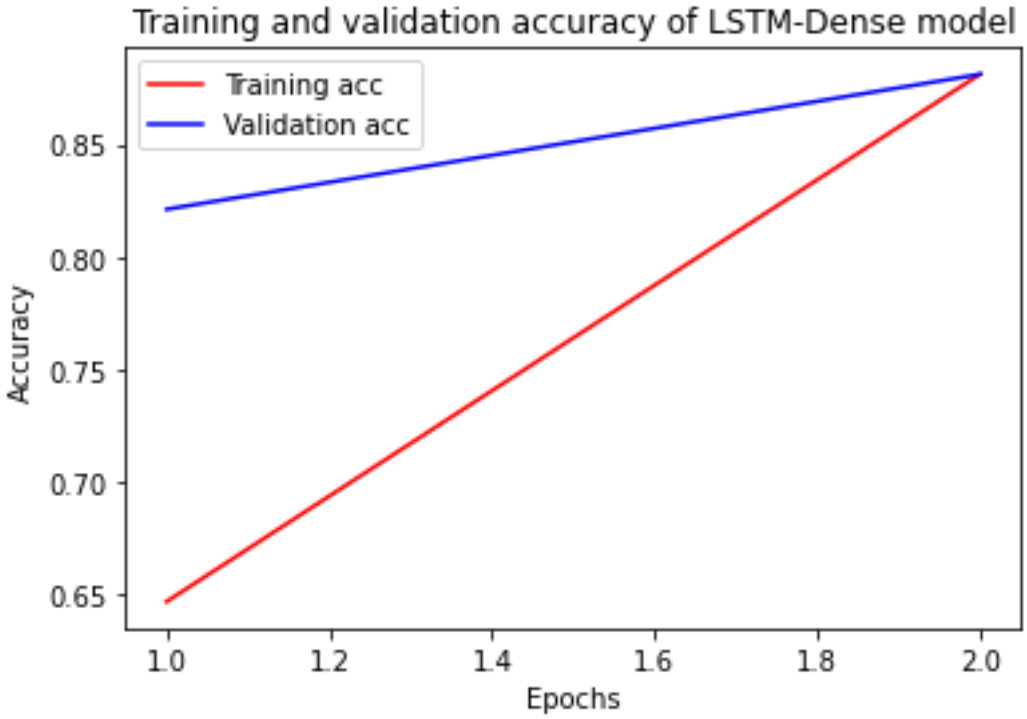

Рисунок 2 – Точность при обучении LSTM-Dense модели

Точность на валидационных данных составила 0.8822 на последней эпохе.

Графики точности и потерь модели №2 Conv + LSTM во время обучения представлены на рис. 3, 4.

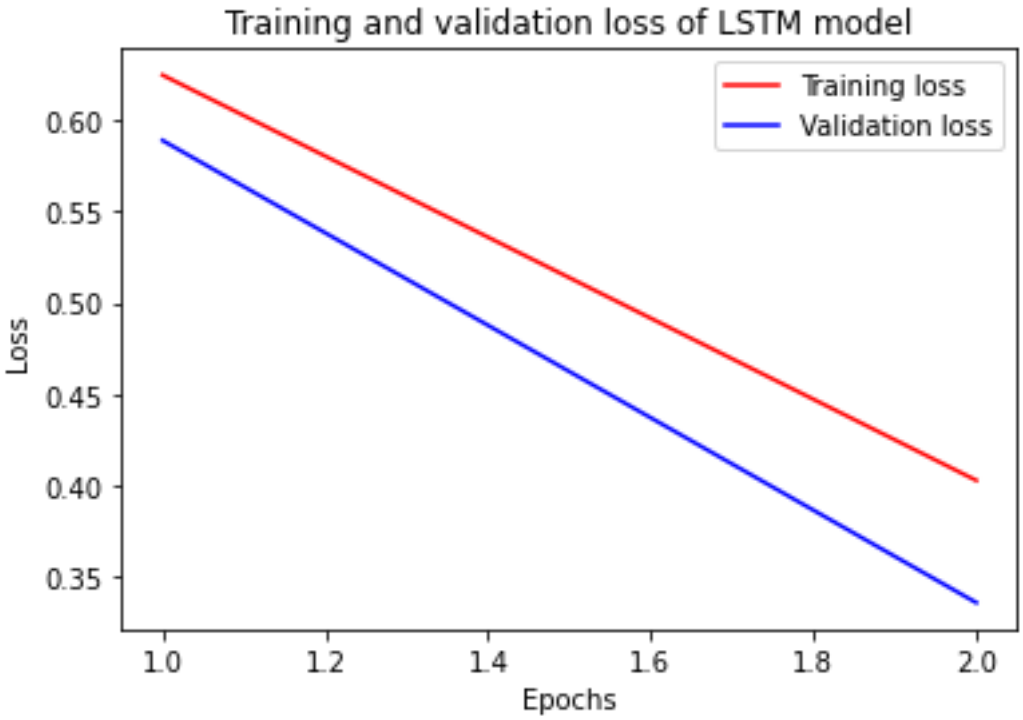

Рисунок 3 – Потери при обучении Conv-LSTM модели

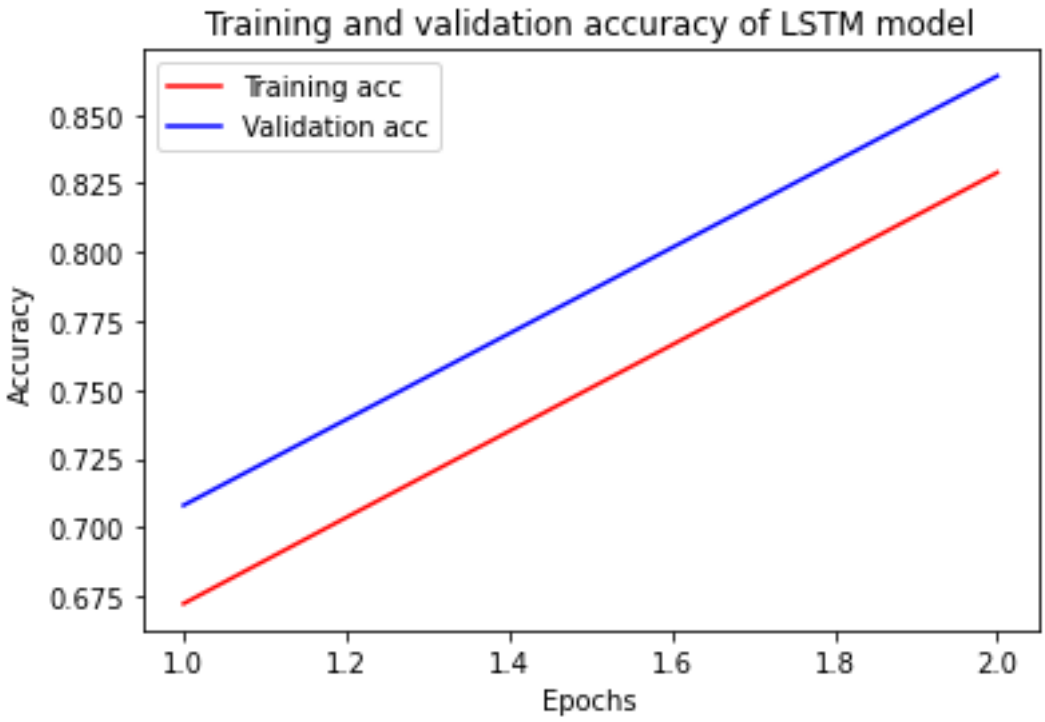

Рисунок 4 – Точность при обучении Conv-LSTM модели

Точность на валидационных данных составила 0.8814 на последней эпохе.

Графики точности и потерь модели №3 LSTM во время обучения представлены на рис. 5, 6.

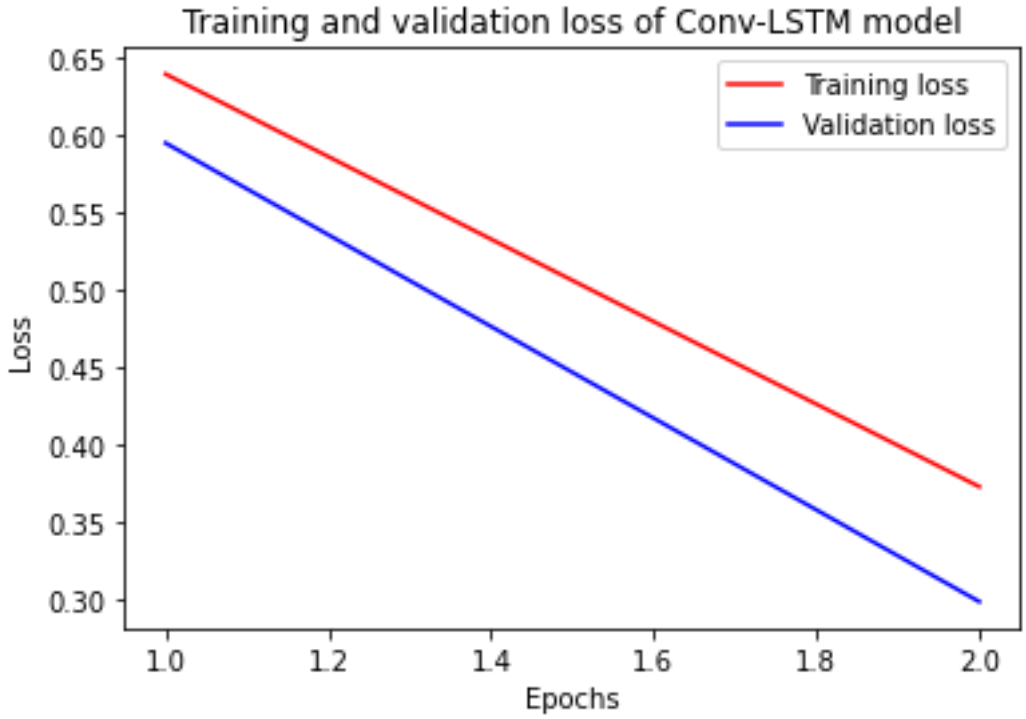

Рисунок 5 – Потери при обучении LSTM модели

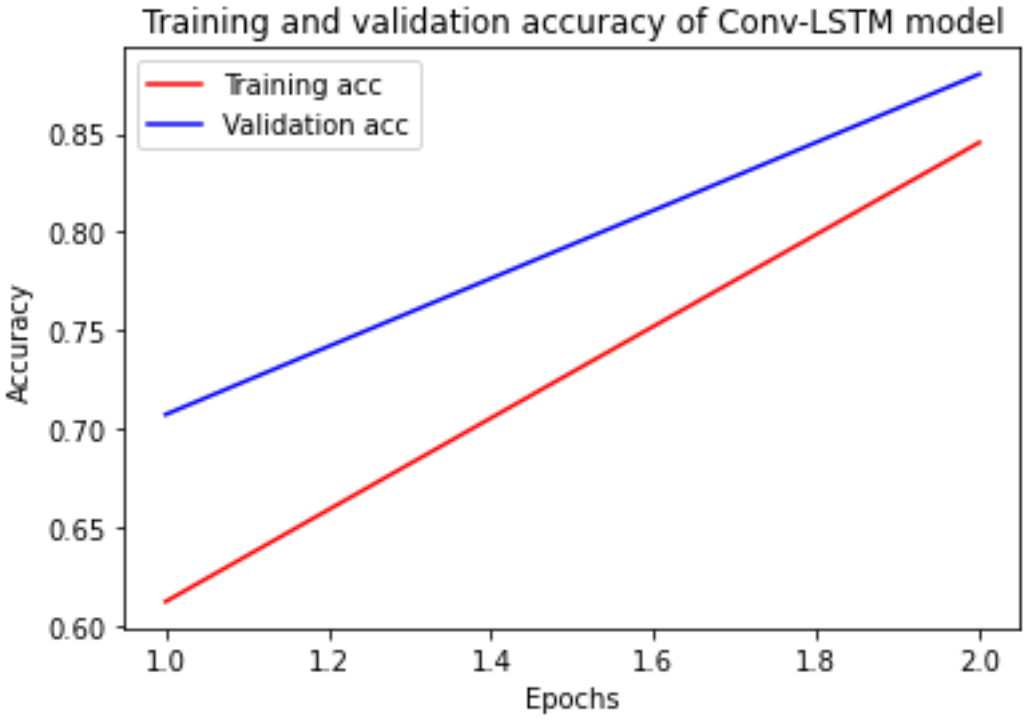

Рисунок 6 – Точность при обучении LSTM модели

Точность на валидационных данных составила 0.8740 на последней эпохе.

Ансамблирование моделей

По результатам обучения были выбраны веса для каждой из моделей:

- Модель №1 LSTM-Dense: 50%
- Модель №2 Conv-LSTM: 40%
- Модель №3 LSTM: 10%

Предсказание высчитывается по формуле:

$$p = 0.5 * p_1 + 0.4 * p_2 + 0.1 * p_3$$

Предсказание таже округляется для получения результата 0 или 1 (классификация).

Результат тестирования на тестовых данных показал, что точность ансамбля 0.88356, что выше, чем точность каждой из моделей в отдельности, хоть и незначительно.

Считывание пользовательского текста

Пользовательские обзоры хранятся в списке в программе и подаются на вход функции test_text, в которой производится индексация слов в отзыве, а далее предсказывается результат ансамблем моделей.

Текст №1 (правильный ответ – 1)

```
Wow, this film is really beautiful, I like it very much
```

Предсказание: 1

Текст №2 (правильный ответ – 0)

```
What a bad acting, feeling like I'm at a children's party. The premise of the film is empty and uninteresting
```

Предсказание: 0

Текст №3 (правильный ответ – 1)

```
I really love the actor who played this role, and in this film he did not disappoint, the picture is very memorable and surprising
```

Предсказание: 1

Стоит отметить, что несмотря на слово disappoint, в третьем тексте ансамбль верно предсказал настроение обзора. В этом и есть преимущество рекуррентных сетей, так как они позволяют учитывать контекст, а не только наличие слов.

Выводы.

В ходе выполнения лабораторной работы было изучено решение задачи классификации отзывов на фильмы для определения успешности фильма с помощью рекуррентных сетей. Был создан ансамбль из нескольких моделей, который корректно предсказать настроение пользовательских текстов.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл lr7.py

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

max_review_length = 500
top_words = 10000
embedding_vector_length = 32

reviews = ["Wow, this film is really beautiful, I like it very much",
           "What a bad acting, feeling like I'm at a children's party. The premise
of the film is empty and uninteresting",
           "I really love the actor who played this role, and in this film he did
not disappoint, the picture is very memorable and surprising"]

(training_data,    training_targets),    (testing_data,    testing_targets)    =
imdb.load_data(num_words=top_words)
training_data = sequence.pad_sequences(training_data, maxlen=max_review_length)
testing_data = sequence.pad_sequences(testing_data, maxlen=max_review_length)

def lstm_model():
    model = Sequential()
    model.add(Embedding(top_words,                      embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(100))
    model.add(Dense(1, activation="sigmoid"))
    model.compile(loss='binary_crossentropy',                optimizer='adam',
metrics=['accuracy'])
    return model

def lstm_dense_model():
    model = Sequential()
    model.add(Embedding(top_words,                      embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(100))
    model.add(Dropout(0.4))
    model.add(Dense(50, 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 conv_lstm_model():
    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(0.3))

    model.add(LSTM(150))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',                  optimizer='adam',
metrics=['accuracy'])
    return model

def train_model(model, name):
    H   =   model.fit(training_data,   training_targets,   validation_split=0.1,
epochs=2, batch_size=256)
    model.save(name + '.h5')
    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, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss of ' + name + ' model')
    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 of ' + name + ' model')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()
```

```
    return model

def ensemble(models):
    predict_a = models[0].predict(testing_data)
    predict_b = models[1].predict(testing_data)
    predict_c = models[2].predict(testing_data)
    prediction = numpy.array([1 if (0.3 * predict_a[k] + 0.6 * predict_b[k] + 0.1
* predict_c[k]) > 0.5 else 0 for k in range(len(predict_a))])
    acc = [1 if prediction[i] == testing_targets[i] else 0 for i in
range(len(prediction))]
    acc = acc.count(1) / len(acc)
    print(acc)

def test_text(line, models):
    data = [w.strip(''.join(['.', ',', ':', ';', '!', '?', '(', ')'])).lower()
for w in line.strip().split()]
    index = imdb.get_word_index()
    x_test = []
    for w in data:
        if w in index and index[w] < top_words:
            x_test.append(index[w] + 3)
    x_test = sequence.pad_sequences([x_test], maxlen=max_review_length)
    predict_a = models[0].predict(x_test)
    predict_b = models[1].predict(x_test)
    predict_c = models[2].predict(x_test)
    return 1 if (0.3 * predict_a + 0.6 * predict_b + 0.1 * predict_c > 0.5) else
0


lstm_dense_model = lstm_dense_model()
conv_lstm_model = conv_lstm_model()
lstm_model = lstm_model()
lstm_dense_model = train_model(lstm_dense_model, "LSTM-Dense")
conv_lstm_model = train_model(conv_lstm_model, "Conv-LSTM")
lstm_model = train_model(lstm_model, "LSTM")
ensemble([lstm_dense_model, conv_lstm_model, lstm_model])

for r in reviews:
  res = test_text(r, [lstm_dense_model, conv_lstm_model, lstm_model])
  print("Review:")
  print(r)
  print("Prediction:", res)
```