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

ОТЧЕТ

по лабораторной работе №6
"Прогноз успеха фильмов по обзорам"
по дисциплине «Искусственные нейронные сети»

Студент гр. 8382 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

Прогноз успеха фильмов по обзорам (Predict Sentiment From Movie Reviews).

Задание.

- Ознакомиться с задачей классификации
- Изучить способы представления текста для передачи в ИНС
- Достигнуть точность прогноза не менее 95%

Требования:

- Построить и обучить нейронную сеть для обработки текста
- Исследовать результаты при различном размере вектора представления текста
- Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте)

Выполнение работы.

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

Была создана функция fit_model(), в которой загружаются данные из датасета IMDb, после чего происходит векторизация для получения тензоров входных параметров.

Модель сети состоит из следующих слоев:

- Слой Dense, 50 нейронов, функция активации – relu, input_shape соответствует размеру вектора представления текста
- Слой Dropout с параметром rate = 0.5
- Слой Dense, 50 нейронов, функция активации – relu
- Слой Dropout с параметром rate = 0.5
- Слой Dense, 50 нейронов, функция активации – relu

- Слой Dropout с параметром rate = 0.2
- Слой Dense, 1 нейрон, функция активации – sigmoid

Стоит отметить, что выбраны достаточно большие параметры rate для слоев Dropout между слоями с функцией активации relu, так как при более низком коэффициенте отсеивания быстро возникало переобучение сети.

Были выбраны следующие параметры для компиляции:

- Оптимизатор: adam
- Функция потерь: бинарная кросс-энтропия, так как данную задачу можно рассматривать как задачу бинарной классификации (0 – фильм не успешный, 1 – фильм успешный)
- Метрики: точность

Для обучения были выбраны следующие параметры:

- Размер батча: 4000. При небольших размерах батча модель переобучается уже после 1-й эпохи. Также больший размер позволил увеличить точность модели, что компенсирует более долгое обучение
- Число эпох: 7
- Размер обучающей выборки: 40000
- Размер валидационной выборки: 10000

Далее были проведены тесты модели с различным значением размера вектора представления текста.

1. 3000 исследуемых слов

Графики потерь и точности при обучении модели представлены на рис. 1, 2.

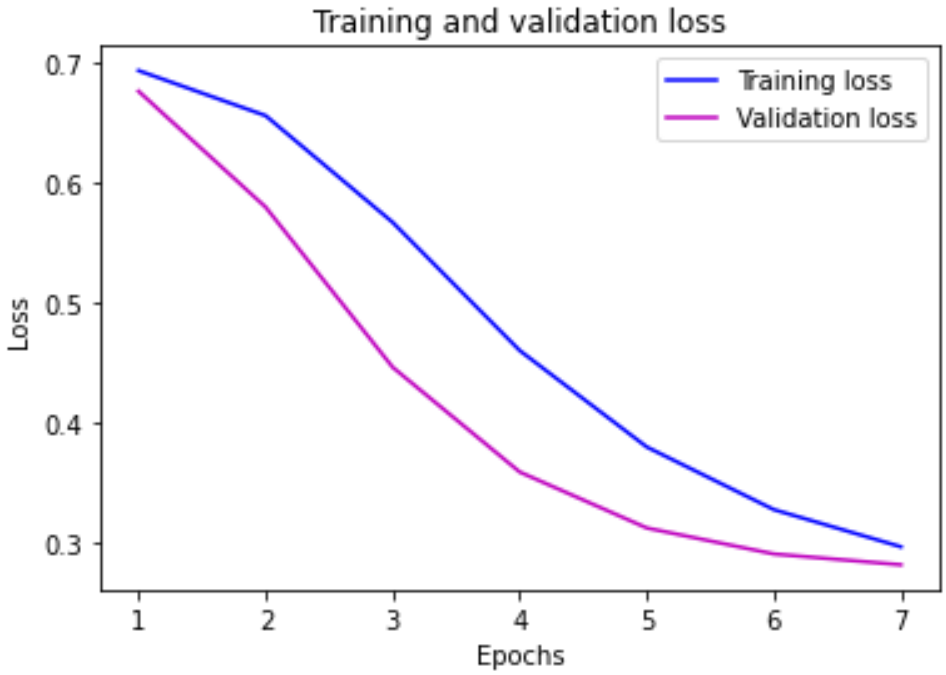

Рисунок 1 – Графики потерь (3000 слов)

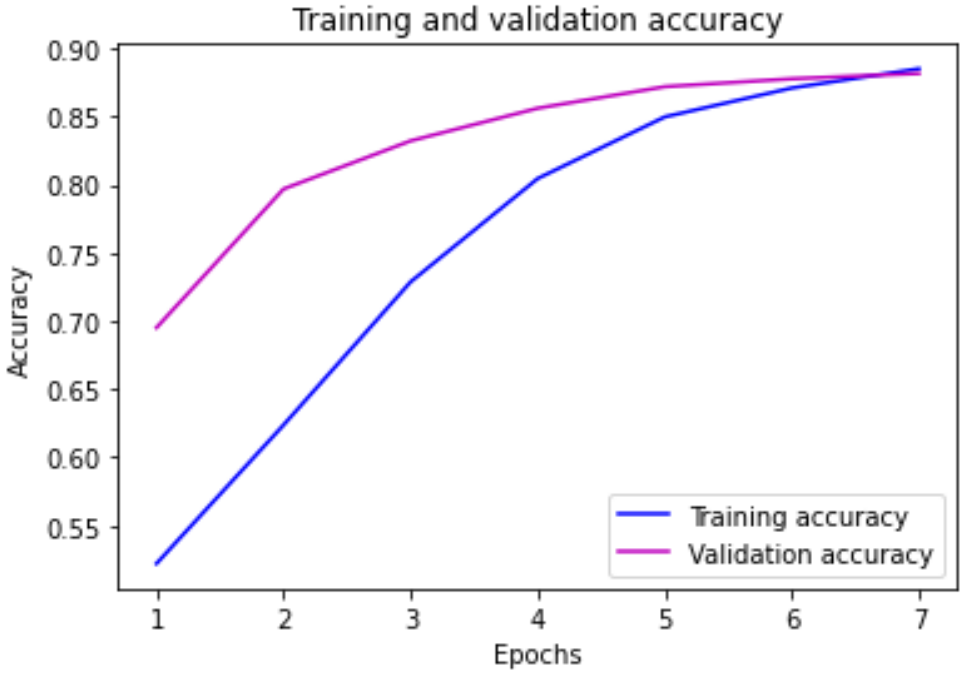

Рисунок 2 – Графики точности (3000 слов)

Итоги:

- Переобучение модели не было замечено
- Точность на тестовой выборке составила 88.15%
- Потери на тестовой выборке составили 0.2815

2. 5000 исследуемых слов

Графики потерь и точности при обучении модели представлены на рис. 3, 4.

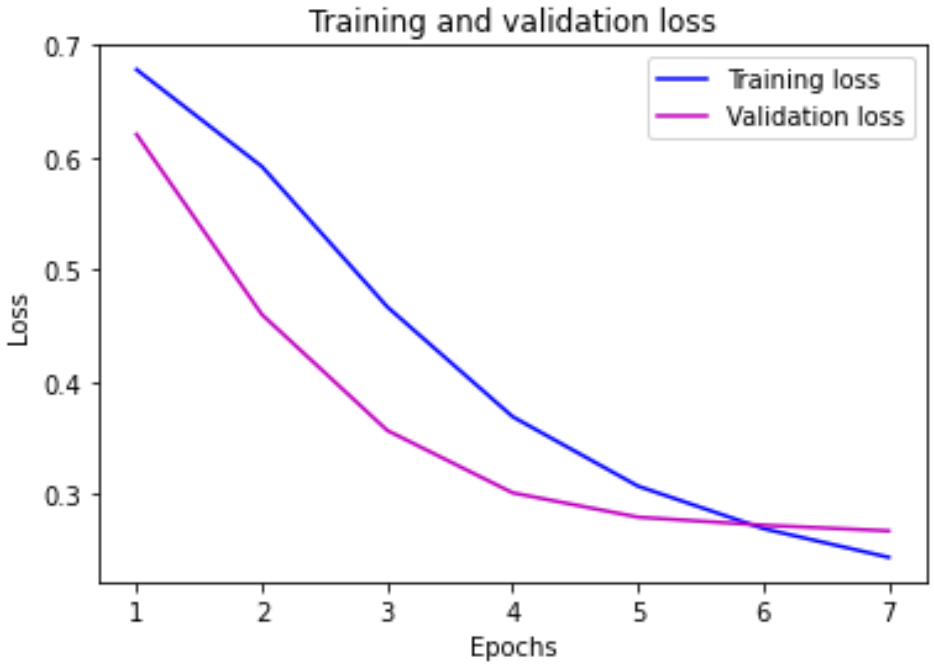

Рисунок 3 – Графики потерь (5000 слов)

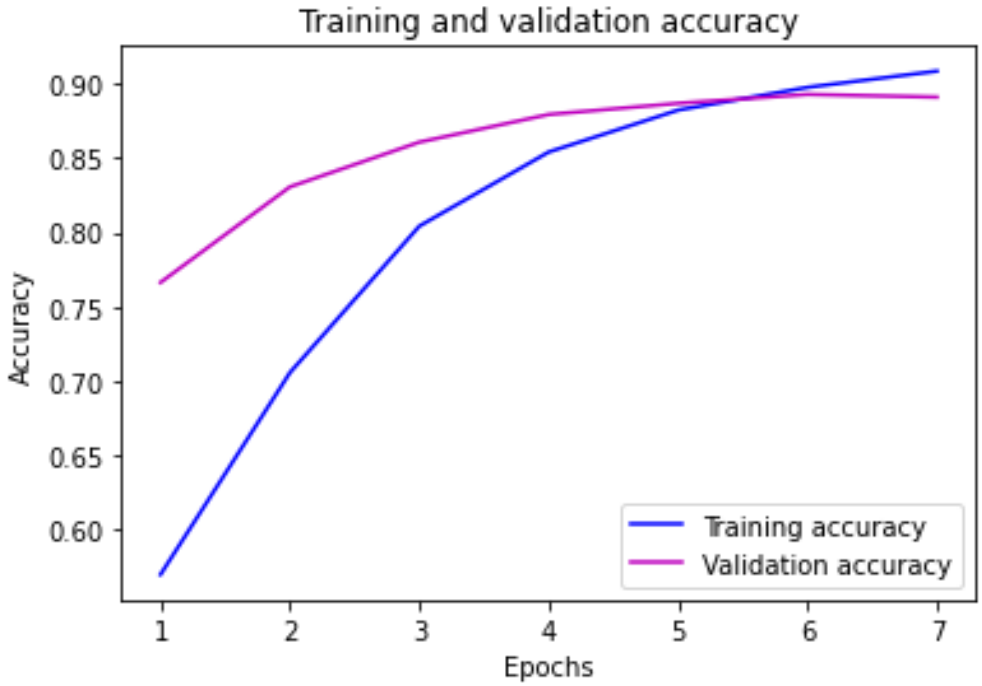

Рисунок 4 – Графики точности (5000 слов)

Итоги:

- Переобучение не было замечено
- Точность на тестовой выборке составила 89.1%
- Потерь на тестовой выборке составили 0.2676
- Увеличение количества исследуемых слов в тексте с 3000 до 5000 положительно повлияло на точность модели, также уменьшились потерь

3. 10000 исследуемых слов

Графики потерь и точности при обучении модели представлены на рис. 5, 6.

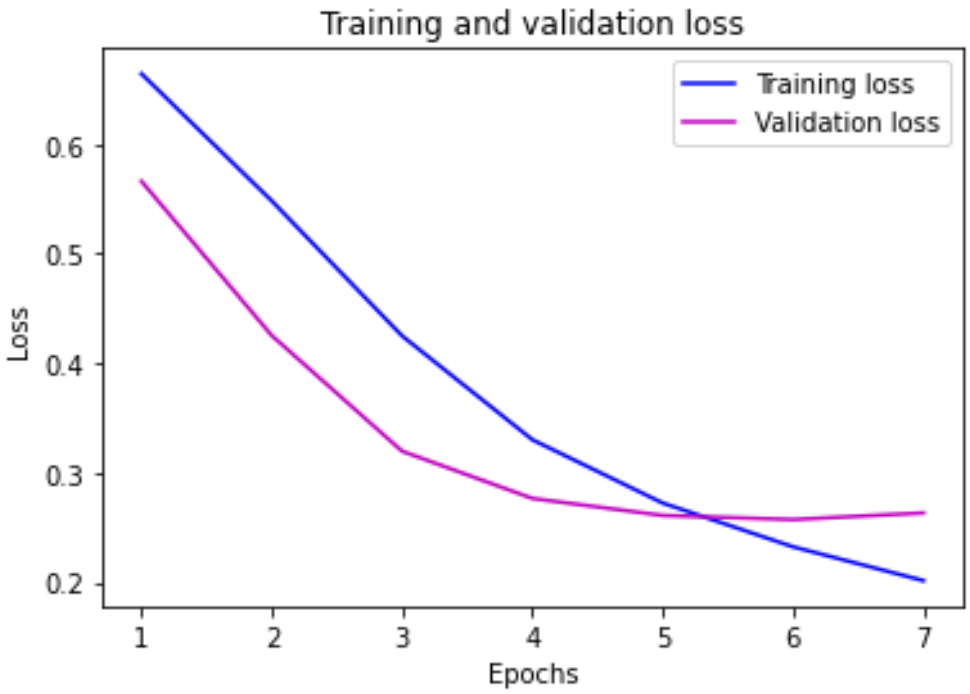

Рисунок 5 – Графики потерь (10000 слов)

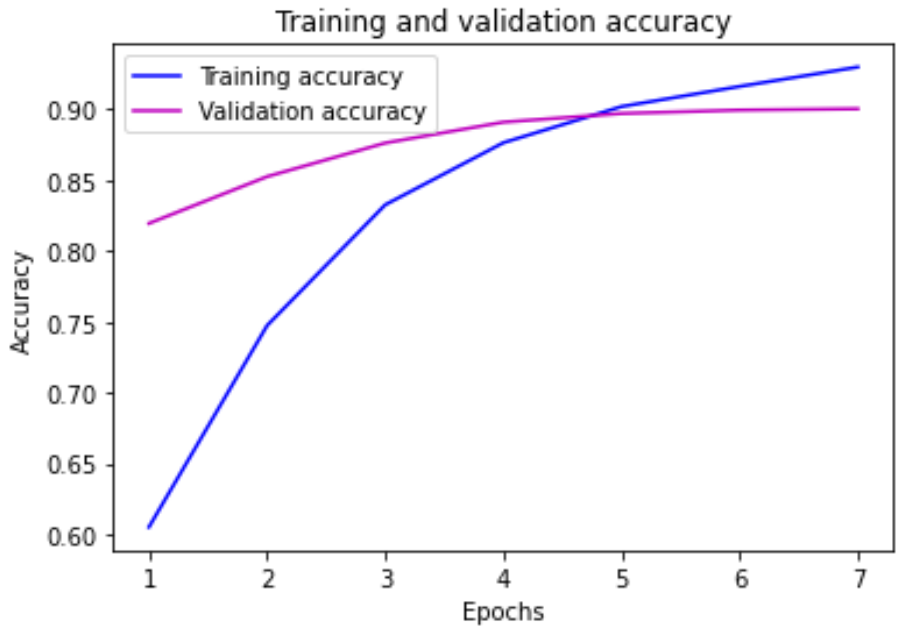

Рисунок 6 – Графики точности (10000 слов)

Итоги:

- После 6 эпохи возникает незначительное увеличение ошибки на валидационных данных, но точность все же возрастает. Следовательно, переобучение не ярко выражено
- Точность на тестовой выборке составила 90.1%
- Потери на тестовой выборке составили 0.2632
- Увеличение количества исследуемых слов в тексте с 5000 до 10000 положительно повлияло на точность модели – увеличение составило 1%. Потери уменьшились незначительно.

В результате можно сказать, что увеличение количества исследуемых слов увеличивает точность модели. Можно предположить, что большинство слов в текстах отзывов действительно определяют принадлежность его к отрицательному или положительному.

Кроме того, при увеличении количества исследуемых слов модель быстрее приходит к точке переобучения (при 8 эпохах оно было бы уже более ярко выраженным). Скорее всего это связано с тем, что при большом количестве параметров модель улавливает признаки отзывов, присущие только обучающей выборке.

Также была создана функция для загрузки пользовательского текста отзыва. Функция разбивает текст на слова, а далее индексирует их с помощью функции, входящей в библиотеку датасета IMDb keras: get_word_index, и векторизирует для дальнейшего использования в качестве входных данных нейронной сети.

В файлы с названиями “good1.txt”, “good2.txt”, “bad1.txt”, “bad2.txt” были загружены переводы фрагментов отзывов с платформы КиноПоиск. Оценка отзыва (положительный или отрицательный) соответствует названиям файлов. Тексты файлов и предсказания модели приведены ниже:

good1.txt :

In general, I liked the film. There are, of course, a couple of comments, but you can't please everyone. I didn’t regret spending my money on going to the movies and not waiting for the movie to appear on the Internet.

Результат на выходе нейронной сети - 0.88382953

Вывод – отзыв положительный (корректно)

good2.txt :

The film is filled with many scenes with magnificent landscapes of Crimea, undoubtedly arouses patriotism and touches to the quick. Definitely, this is the film that you need to watch and draw your own conclusions, and not rely on reviews before watching ..

Результат на выходе нейронной сети - 0.915031

Вывод – отзыв положительный (корректно)

bad1.txt

Direction: absent. The person who filmed this has no idea either of the author's handwriting, or of the director's style, or of the vision ... or of the banal visual effects. No real atmosphere, no rhythm, no ... nothing! Looking at such a wretched line with the main characters, one wonders why the director even got into this, because he with such obviousness does not succeed in real human feelings.

Результат на выходе нейронной сети - 0.01972552

Вывод – отзыв отрицательный (корректно)

bad2.txt

Like a movie, 'Crimea' is a lazy, clumsy hack. From the point of view of shooting, everything is bad: the scale of events in Ukraine is not felt because of about 10 locations for the entire film, the camera angles are as banal and inventive as possible, as if a first-year journalist was filming. The editor in his skills did not go far from the operator: the transition between scenes is carried out only by darkening, color correction in different scenes, sometimes even at the same location, is not combined with each other, and the frame is flat separately, as if the frame was not painted at all. The director's disregard is clearly seen in the ridiculous scene with the attack of the revolutionaries on the Berkut and in the choreography of the battle in general. The writers lazily prescribed the heroes of the film, sometimes without giving motivation for the actions shown in the film, a huge number of plot holes and illogicalities, and the dialogues were written by the forty-year-old loader Vasily, there are so many cliches that can be found in the federal media. As a result, the actors do not know who to play and do their job poorly. A waste of money for ordinary citizens who want to see a movie.

Результат на выходе нейронной сети - 0.01289974

Вывод – отзыв отрицательный (корректно)

Модель верно предсказала настроения отзывов, однако стоит отметить, что отзывы подбирались так, чтобы в них было описано больше впечатлений и эмоций, нежели описания конкретных моментов фильма.

Выводы.

В ходе выполнения лабораторной работы было изучено решение задачи классификации отзывов на фильмы для определения успешности фильма. Было изучено, как передавать нейронной сети текстовые данные, а также, как влияет количество слов-индикаторов (входных параметров) на обучение сети. В результате была обучена нейронная сеть, показываются на тестовой выборке точность 90.1%.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл lr6.py

```
import matplotlib.pyplot as plt
import numpy as np
from keras import Sequential
from keras import layers
from keras.utils import to_categorical
from keras import models
from keras.datasets import imdb

files = ['good1.txt', 'good2.txt', 'bad1.txt', 'bad2.txt']

def vectorize(sequences, dimension):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results

def load_text(filename, idim):
    data = []
    print("\n", filename, ":")
    with open(filename, 'r') as file:
        for line in file.readlines():
            print(line)
            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] < idim:
            x_test.append(index[w])
    x_test = vectorize([np.array(x_test)], idim)
    return x_test

def predict_success(x_test, model):
    prediction = model.predict(x_test)
    print(prediction)
    if prediction > 0.5 :
      print("This film is going to be successful")
    else:
      print("This film is going to be unsuccessful")

def fit_model(idim):
    (training_data,   training_targets),   (testing_data,   testing_targets)   =
imdb.load_data(num_words=idim)
    data = np.concatenate((training_data, testing_data), axis=0)
```

```
    targets = np.concatenate((training_targets, testing_targets), axis=0)
    data = vectorize(data, idim)
    targets = np.array(targets).astype("float32")

    test_x = data[:10000]
    test_y = targets[:10000]
    train_x = data[10000:]
    train_y = targets[10000:]

    model = Sequential()
    model.add(layers.Dense(50, activation = "relu", input_shape=(idim, )))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(50, activation = "relu"))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(50, activation = "relu"))
    model.add(layers.Dropout(0.2))
    model.add(layers.Dense(1, activation = "sigmoid"))

    model.compile( optimizer = "adam", loss = "binary_crossentropy", metrics =
["accuracy"])
    history = model.fit(train_x, train_y, epochs = 7, batch_size = 4000,
validation_data = (test_x, test_y))

    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(loss) + 1)
    plt.plot(epochs, loss, 'b', label='Training loss')
    plt.plot(epochs, val_loss, 'm', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()

    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    plt.plot(epochs, acc, 'b', label='Training accuracy')
    plt.plot(epochs, val_acc, 'm', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

    results = model.evaluate(test_x, test_y)
    print(results)
```

```
    return model

sizes = [10000]
for size in sizes:
  print("Testing with size", size)
  model = fit_model(size)

print("Testing on custom texts")
good = load_text(files[0], 10000)
predict_success(good, model)

bad = load_text(files[1], 10000)
predict_success(bad, model)

good = load_text(files[2], 10000)
predict_success(good, model)

bad = load_text(files[3], 10000)
predict_success(bad, model)
```