

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

ОТЧЕТ

По индивидуальному заданию
по дисциплине «Искусственные нейронные сети»
ТЕМА: Определение рейтинга на основе отзыва

Студент гр. 7382

Бахеров Д.В.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Задача заключается в классификации рейтинга по отзыву, который оставил пользователь.

Описание датасета.

Датасет представляет собой таблицу из 568454 строк и 10 столбцов, содержащую информацию об отзыве.



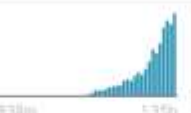
# HelpfulnessDen...	# Score	# Time	A Summary	A Text
Number of users who indicated whether they found the review helpful or not	Rating between 1 and 5	Timestamp for the review	Brief summary of the review	Text of the review
			295744 unique values	393579 unique values
1	5	1383862400	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good...
0	1	1345976800	Not as Advertised	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. No...
1	4	1219017600	"Delight" says it all	This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin wi...

Рис. 1

Для решения поставленной задачи не требуется все данные, которые представлены в датасете. Было решено взять 7 столбец с информацией об оценке и 10 столбец с соответствующим отзывом.

Ход работы.

1. Обработка данных

Набор данных загружается напрямую с помощи pandas. Затем разделяются столбцы на входные(reviews) и выходные(rates) данные. А так же было принято решение взять половину отзывов(был взят каждый второй отзыв в таблице), так как личный компьютер не выдерживал нагрузки.

```
dataframe = pandas.read_csv("reviews.csv", header=0)
dataset = dataframe.values
reviews = dataset[:,9]
rates = dataset[:,6]
```

Рис. 3

В ходе работы данные были разделены на обучающую и контрольную выборки.

```
train_reviews = dataset[:BORDER:NUM][:,9]
test_reviews = dataset[BORDER::NUM][:,9]

train_rates = dataset[:BORDER:NUM][:,6]
test_rates = dataset[BORDER::NUM][:,6]
```

Рис. 4

Далее преобразовали выходные атрибуты(столбцы) из векторов в матрицу при помощи функции to_categorical().

```
train_rates = to_categorical(train_rates)
train_rates = train_rates[:,1:]
test_rates = to_categorical(test_rates)
test_rates = test_rates[:,1:]
```

Рис. 5

А входные данные были представлены в виде чисел при помощи функции one_hot.

```
def encode_review(rev):
    rev = [one_hot(d, VOCAB_SIZE) for d in rev]
    padded_reviews = sequence.pad_sequences(rev, MAX_REVIEW_LENGTH)
    return padded_reviews

encoded_reviews = encode_review(reviews)
encoded_train_reviews = encode_review(train_reviews)
encoded_test_reviews = encode_review(test_reviews)
```

Рис. 6

Так же длина отзывов варьируется, среднее значение равно 436 слов.

```
Average Review length: 436.25485263539355
```

Рис. 7

Длина отзывов была ограничена 400 словами при помощи `pad_sequences`(см. рис. 6).

2. Создание модели

Сеть состоит из 8 слоёв:

- входного Embedded, который использует 32 вектора длины для представления каждого слова.
- одномерный слой CNN
- слой Dropout с коэффициентом исключения 0,3
- максимальный пул MaxPoling1D
- слой Dropout с коэффициентом исключения 0,5
- слой LSTM со 100 единицами памяти
- слой Dropout с коэффициентом исключения 0,3
- выходной слой Dense с 5 нейронами, так как всего пять классов и функцией активации `sigmoid`, так как данная задача – это задачи классификации. Функция потерь – `categorical_crossentropy`, так как несколько классов.

```
model = Sequential()
model.add(Embedding(VOCAB_SIZE, 32, input_length=MAX_REVIEW_LENGTH))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(Dropout(0.3))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.5))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(5, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рис .8

3. Обучение модели сети

- 1) Начальные тесты проводились на неразделенных данных(обучающая выборка включает в себя контрольную).

Данные включали в себя 5685 отзывов:

Тестирование:

Accuracy: 81,48%

Loss: 48,79%

```
Train on 5685 samples, validate on 5685 samples
Epoch 1/5
5685/5685 [=====] - 472s 83ms/step - loss: 1.8085 - accuracy: 0.6383 - val_loss: 0.9606 - val_accuracy: 0.6848
Epoch 2/5
5685/5685 [=====] - 463s 81ms/step - loss: 0.8909 - accuracy: 0.6714 - val_loss: 0.7307 - val_accuracy: 0.7105
Epoch 3/5
5685/5685 [=====] - 468s 81ms/step - loss: 0.7744 - accuracy: 0.7069 - val_loss: 0.6323 - val_accuracy: 0.7497
Epoch 4/5
5685/5685 [=====] - 470s 83ms/step - loss: 0.6967 - accuracy: 0.7261 - val_loss: 0.5517 - val_accuracy: 0.7761
Epoch 5/5
5685/5685 [=====] - 464s 82ms/step - loss: 0.6257 - accuracy: 0.7478 - val_loss: 0.4880 - val_accuracy: 0.8141
5685/5685 [=====] - 3s 576us/step
Test [0.4879851113942598, 0.8148721321105957]
```

Рис .9

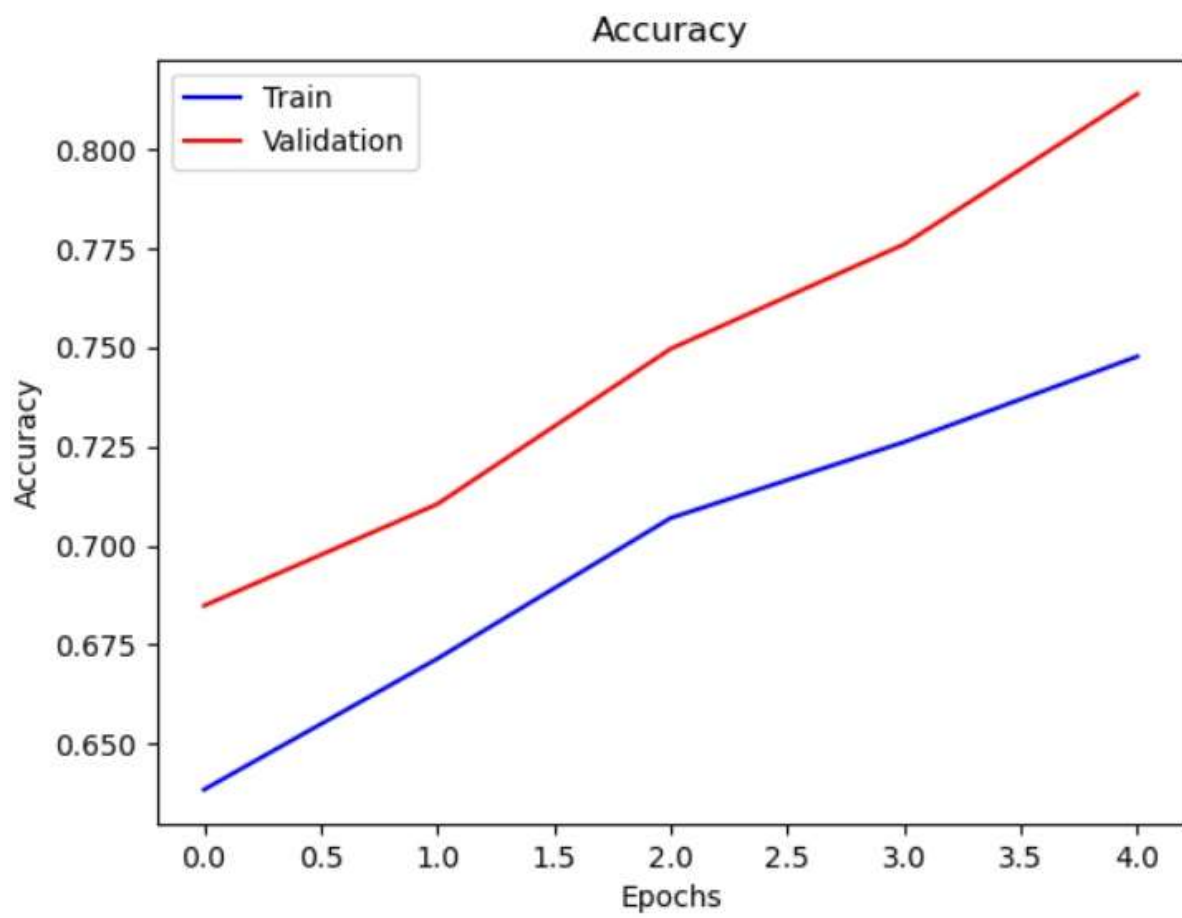


Рис. 10 – График точности

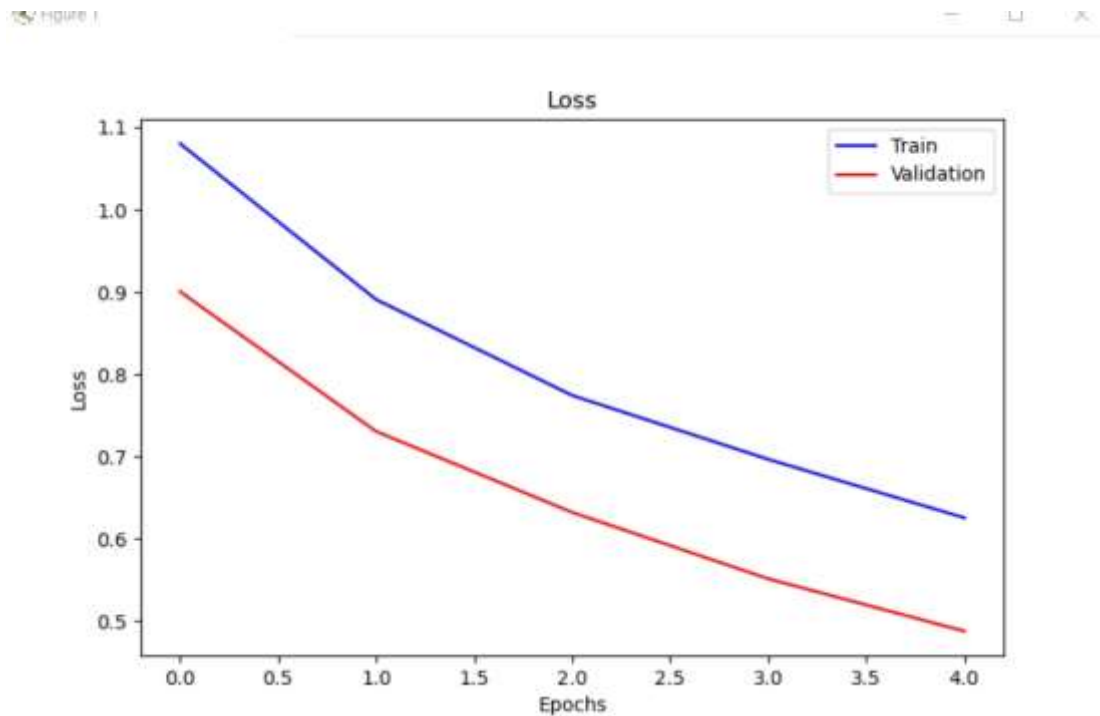


Рис. 11 – График потерь

- 2) Далее обучение уже проходило на всех разделенных данных при epochs=5 и bath_size=250, а так же был написан личный отзыв(с рейтингом 1):

```
my_review_1 = ['Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. '
               'Not sure if this was an error or if the vendor intended to represent the product as ""Jumbo"" .']
```

Рис. 12

```
encoded_my_review_1 = encode_review(my_review_1)
pred_1 = model.predict(encoded_my_review_1)
pred1 = np.array(pred_1)
print('Prediction of my review:', pred1)
```

Рис. 13 - Работа с личным отзывом

Тестирование:

Accuracy: 74,68%

Loss: 69,59%

На рис. 14 можно увидеть, что вероятность того, что отзыв имеет рейтинг 1 самая большая(91,27%), то есть сеть верно дала оценку.


```

Train on 288000 samples, validate on 4227 samples
Epoch 1/5
288000/288000 [=====] - 379s 1ms/step - loss: 0.9865 - accuracy: 0.6792 - val_loss: 0.8447 - val_accuracy: 0.6891
Epoch 2/5
288000/288000 [=====] - 384s 1ms/step - loss: 0.7276 - accuracy: 0.7258 - val_loss: 0.7732 - val_accuracy: 0.7128
Epoch 3/5
288000/288000 [=====] - 381s 1ms/step - loss: 0.6636 - accuracy: 0.7473 - val_loss: 0.7867 - val_accuracy: 0.7388
Epoch 4/5
288000/288000 [=====] - 372s 1ms/step - loss: 0.6268 - accuracy: 0.7632 - val_loss: 0.6999 - val_accuracy: 0.7462
Epoch 5/5
288000/288000 [=====] - 378s 1ms/step - loss: 0.5952 - accuracy: 0.7758 - val_loss: 0.6959 - val_accuracy: 0.7469
4227/4227 [=====] - 2s 588us/step
Test [0.6959468696115709, 0.7468653917312622]
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
0.7388671
0.7267565688024598
Prediction of my review: [[3.4267405e-06 9.1274065e-04 5.1923478e-03 7.1026824e-02 6.4388256e-01
1.4589547e-01]]

```

Рис. 14

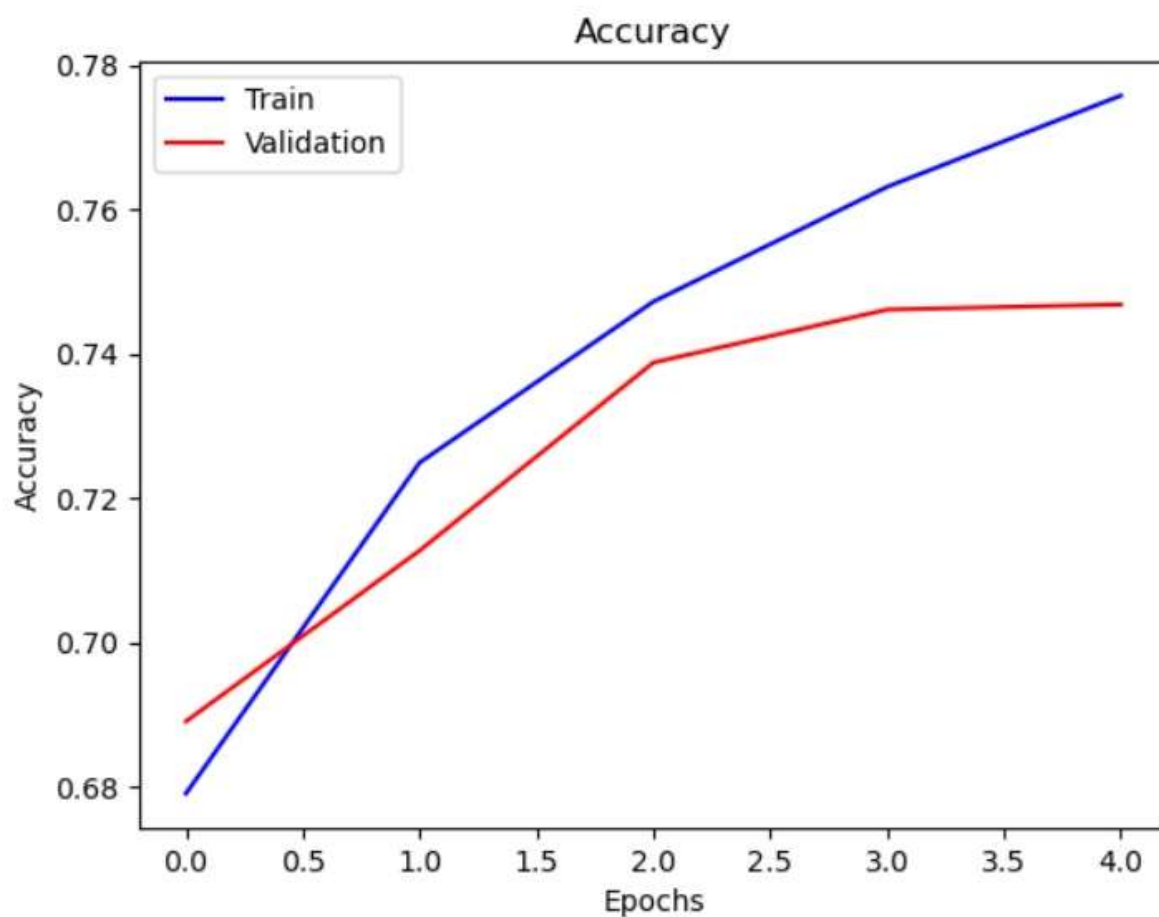


Рис. 15 – График точности

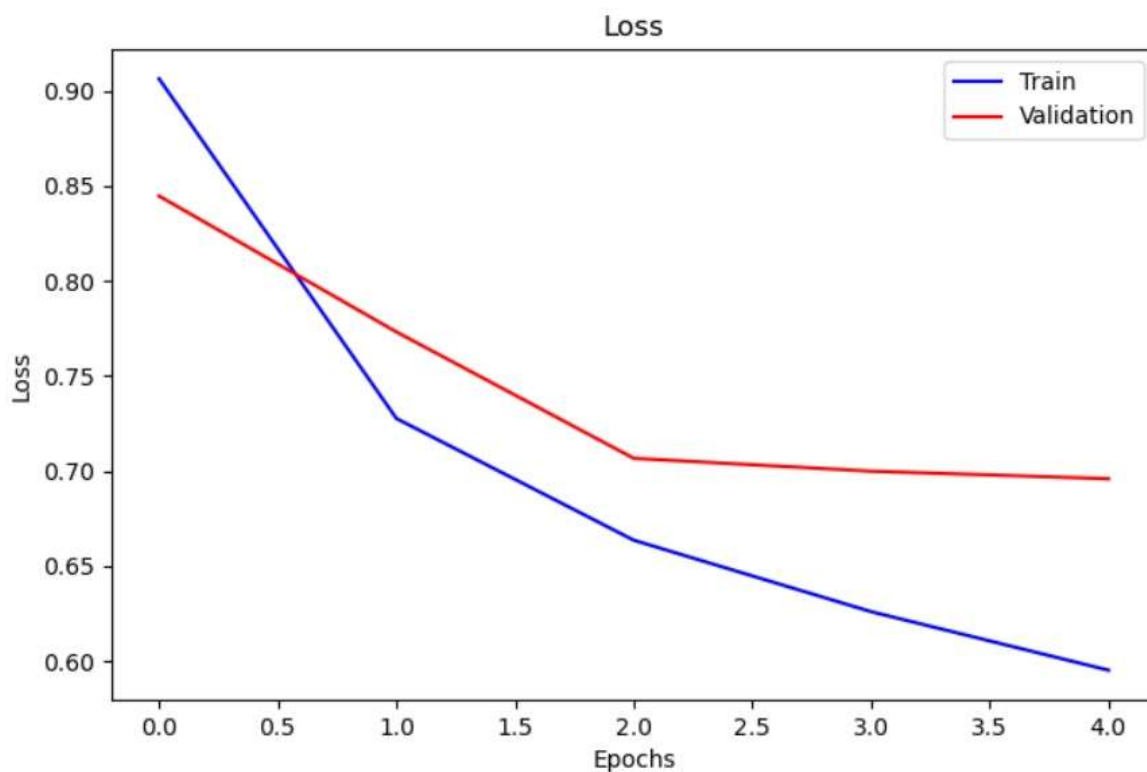


Рис. 16 – График потерь

3) Далее обучение проходило при epochs=10 и bath_size=100:

Проведем тест ИНС:

Accuracy: 76,48%

Loss: 71,89%

```

Train on 288800 samples, validate on 4227 samples.
Epoch 1/10
288800/288800 [=====] - 883s 3ms/step - loss: 0.8278 - accuracy: 0.6971 - val_loss: 0.7372 - val_accuracy: 0.7284
Epoch 2/10
288800/288800 [=====] - 875s 3ms/step - loss: 0.6843 - accuracy: 0.7412 - val_loss: 0.7128 - val_accuracy: 0.7372
Epoch 3/10
288800/288800 [=====] - 875s 3ms/step - loss: 0.6328 - accuracy: 0.7616 - val_loss: 0.6959 - val_accuracy: 0.7488
Epoch 4/10
288800/288800 [=====] - 898s 3ms/step - loss: 0.5955 - accuracy: 0.7764 - val_loss: 0.6846 - val_accuracy: 0.7488
Epoch 5/10
288800/288800 [=====] - 869s 3ms/step - loss: 0.5624 - accuracy: 0.7980 - val_loss: 0.6829 - val_accuracy: 0.7577
Epoch 6/10
288800/288800 [=====] - 888s 3ms/step - loss: 0.5355 - accuracy: 0.8080 - val_loss: 0.6787 - val_accuracy: 0.7568
Epoch 7/10
288800/288800 [=====] - 879s 3ms/step - loss: 0.5106 - accuracy: 0.8112 - val_loss: 0.6881 - val_accuracy: 0.7663
Epoch 8/10
288800/288800 [=====] - 876s 3ms/step - loss: 0.4899 - accuracy: 0.8197 - val_loss: 0.6997 - val_accuracy: 0.7691
Epoch 9/10
288800/288800 [=====] - 877s 3ms/step - loss: 0.4733 - accuracy: 0.8261 - val_loss: 0.7045 - val_accuracy: 0.7646
Epoch 10/10
288800/288800 [=====] - 879s 3ms/step - loss: 0.4564 - accuracy: 0.8334 - val_loss: 0.7189 - val_accuracy: 0.7648
4227/4227 [=====] - 5s 1ms/step
Test [0.718986050656799, 0.764845073223114]

```

Рис. 17

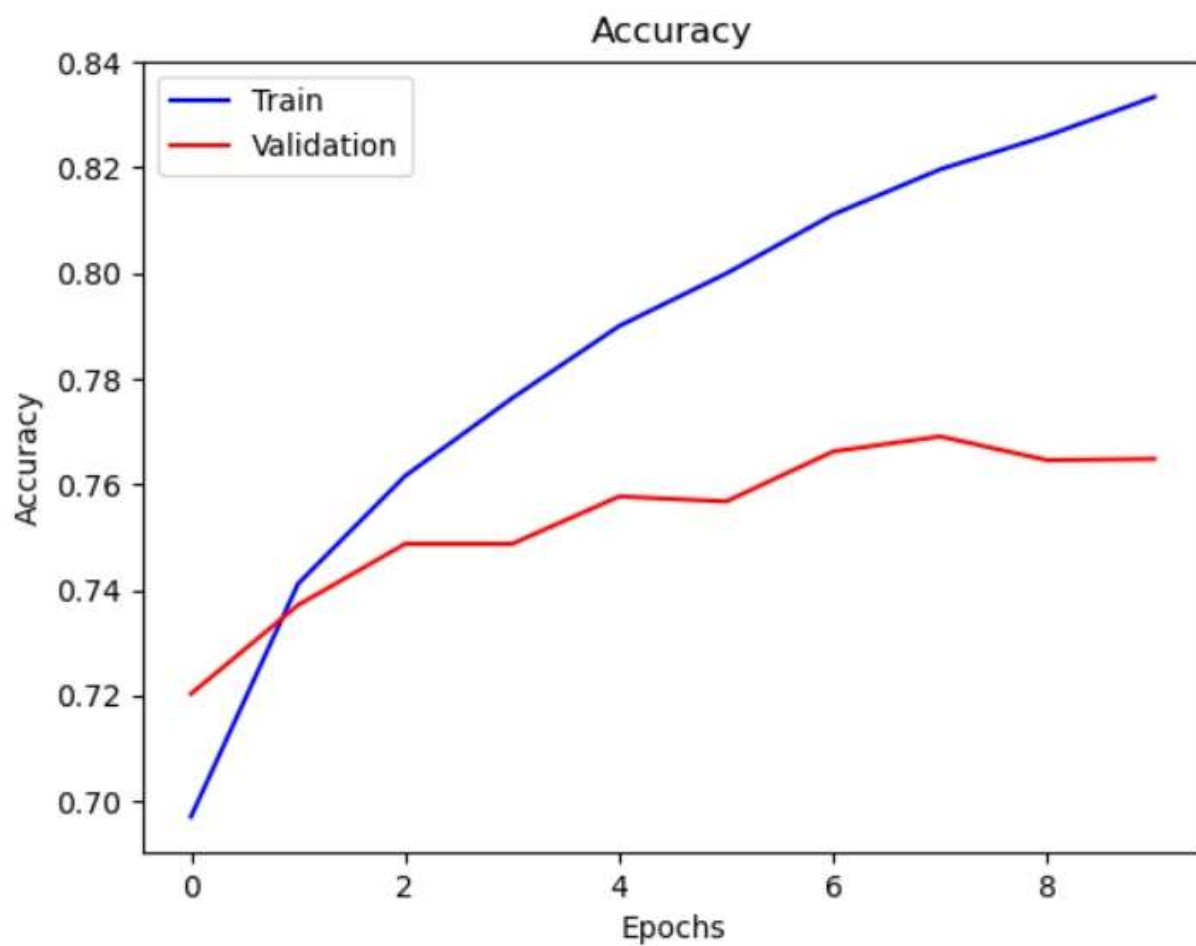


Рис. 18 – График точности

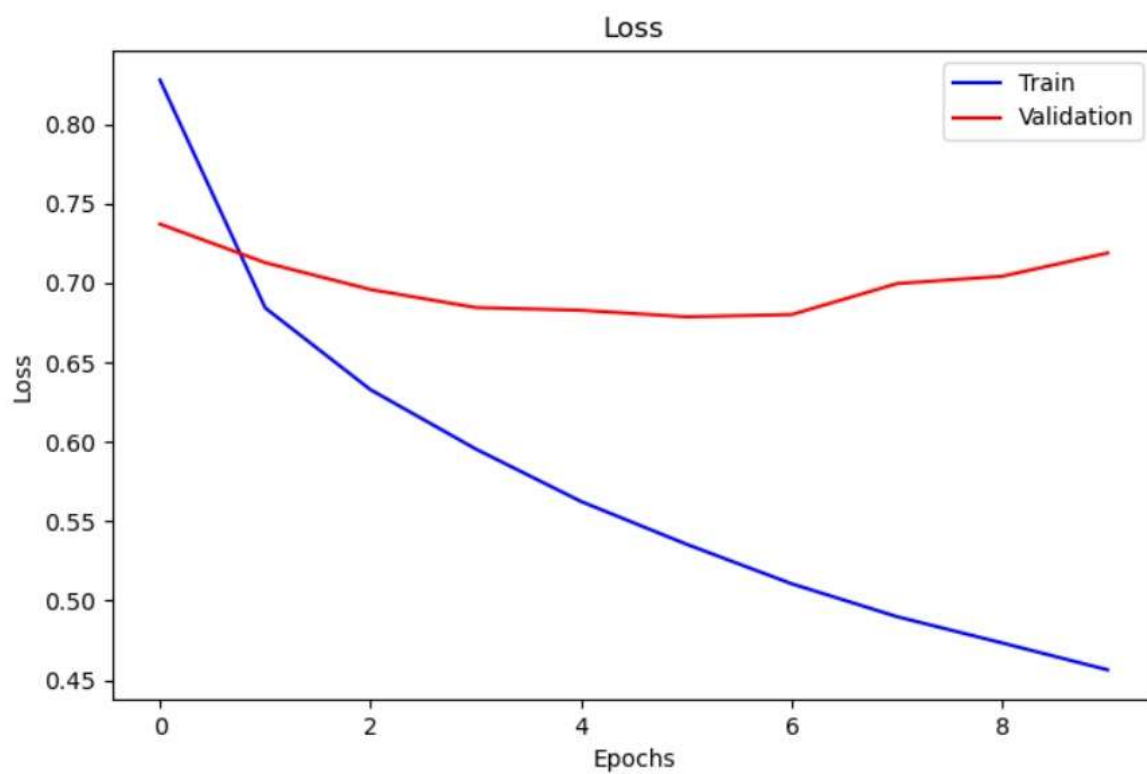


Рис. 19 – График потерь

Анализ результирующей модели.

Сделаем вывод по графикам, что увеличение эпох отрицательно сказывается на модели ИНС, т.к. падает точность и возрастают потери. Это связано с выбранной архитектурой сети, данная архитектура не очень подходит к данной задаче, что затрудняет получение высокой точности. Оптимальной будем считать модель, которая обучается за 7 эпохах.

Вывод.

В ходе выполнения ИДЗ была закреплена знания по обработке данных и работе с ними. Была проведена работа с датасетом, который содержит 568454 объектов данных, которые надо разделить на 5 классов. Была разработана модель на языке Python с использованием Keras API, которая выдает приемлемые результаты.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
# Подключение необходимых библиотек
import numpy as np
import pandas
from keras import Sequential
from keras.datasets import imdb
import matplotlib.pyplot as plt
from keras.layers import Embedding, Conv1D, Conv2D, Dropout, MaxPooling1D, LSTM, Dense, Flatten,
SpatialDropout1D
from keras_preprocessing import sequence
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.preprocessing.text import one_hot
from tensorflow.keras.utils import to_categorical

# Определение констант
MAX_REVIEW_LENGTH = 400
VOCAB_SIZE = 61000
NUM = 100
BORDER = 560000

# Личный отзыв с оценкой 1
my_review_1 = ['Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. '
               'Not sure if this was an error or if the vendor intended to represent the product as ""Jumbo"".']

# Получение данных
dataframe = pandas.read_csv("reviews.csv", header=0)
dataset = dataframe.values
reviews = dataset[:, :NUM][:, 9]
#rates = dataset[:, :NUM][:, 6]

# Разделение данных на выборки
train_reviews = dataset[:, BORDER:NUM][:, 9]
test_reviews = dataset[BORDER::NUM][:, 9]
train_rates = dataset[:, BORDER:NUM][:, 6]
test_rates = dataset[BORDER::NUM][:, 6]

# Преобразование выходных данных
train_rates = to_categorical(train_rates)
train_rates = train_rates[:, 1:]
test_rates = to_categorical(test_rates)
test_rates = test_rates[:, 1:]

#print(reviews)
#print(rates)

# Вывод средней длины отзывов
length = [len(i) for i in reviews]
print("Average Review length:", np.mean(length))

# Функция кодировки и органичения отзывов
def encode_review(rev):
    rev = [one_hot(d, VOCAB_SIZE) for d in rev]
    padded_reviews = sequence.pad_sequences(rev, MAX_REVIEW_LENGTH)
    return padded_reviews

# Преобразование выходных данных
encoded_reviews = encode_review(reviews)
encoded_train_reviews = encode_review(train_reviews)
encoded_test_reviews = encode_review(test_reviews)

# Вывод кол-ва уникальных слов
print("Number of unique words:", len(np.unique(np.hstack(encoded_reviews))))
```

```

# Архитектура модели
model = Sequential()
model.add(Embedding(VOCAB_SIZE, 32, input_length=MAX_REVIEW_LENGTH))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(Dropout(0.3))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.5))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(5, activation='sigmoid'))

# Компиляция модели
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Обучение модели
H = model.fit(encoded_train_reviews, train_rates, batch_size=16, epochs=7, verbose=1,
validation_data=(encoded_test_reviews, test_rates))

# Вывод ошибок и точности
acc = model.evaluate(encoded_test_reviews, test_rates)
print("Test", acc)

# Функция построения графика ошибок
def plot_loss(loss, v_loss):
    plt.figure(1, figsize=(8, 5))
    plt.plot(loss, 'b', label='Train')
    plt.plot(v_loss, 'r', label='Validation')
    plt.title('Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend()
    plt.show()
    plt.clf()

# Функция построения графика точности
def plot_acc(acc, val_acc):
    plt.plot(acc, 'b', label='Train')
    plt.plot(val_acc, 'r', label='Validation')
    plt.title('Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epochs')
    plt.legend()
    plt.show()
    plt.clf()

# Построение графиков
plot_loss(H.history['loss'], H.history['val_loss'])
plot_acc(H.history['accuracy'], H.history['val_accuracy'])

# Оценка моделью личного отзыва
encoded_my_review_1 = encode_review(my_review_1)
pred_1 = model.predict(encoded_my_review_1)
pred1 = np.array(pred_1)
print('Prediction of my review:', pred1)

```