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

ОТЧЕТ

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

Студентка гр. 7382 _____ Петрова А.

Преподаватель _____ Жукова Е.Е.

Санкт-Петербург
2020

Цель

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

Задачи

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

Выполнение работы

Датасет IMDb состоит из 50 000 обзоров фильмов от пользователей, помеченных как положительные (1) и отрицательные (0).

Рецензии предварительно обрабатываются, и каждая из них кодируется последовательностью индексов слов в виде целых чисел.

Слова в обзорах индексируются по их общей частоте появления в датасете. Например, целое число «2» кодирует второе наиболее частое используемое слово.

50 000 обзоров разделены на два набора: 25 000 для обучения и 25 000 для тестирования.

Для обработки текста была построена и обучена нейронная сеть, код которой представлен в приложении А.

Параметры :

- Размер словаря = 10000 слов
- Размер обзора = 500 слов
- Количество эпох = 2
- Функция потерь = binary_crossentropy

Результат работы сети представлен на рис. 1-2.

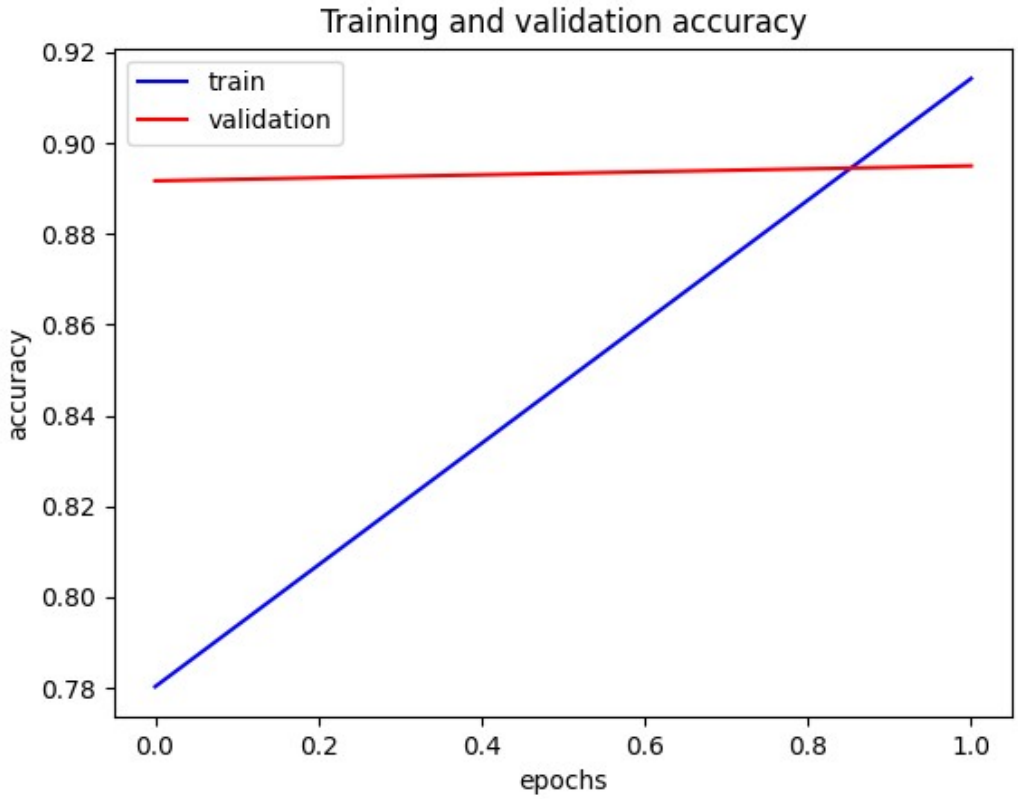

Рисунок 1 — График потерь

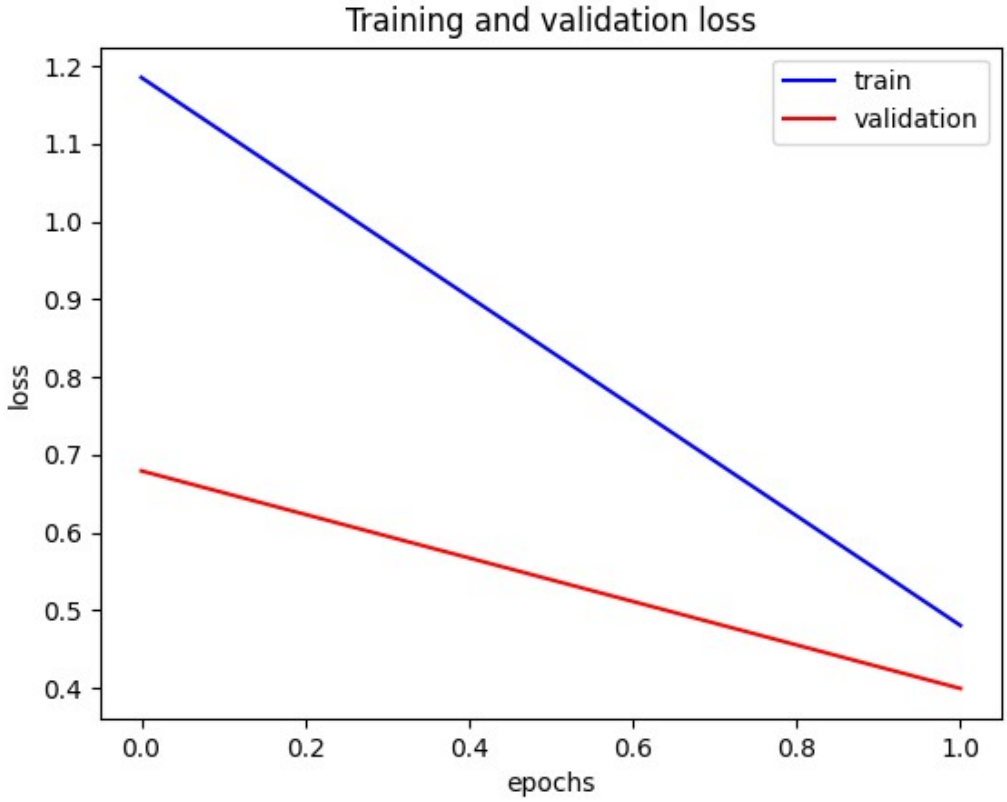

Рисунок 2 — График точности

Из графика на рис. 1 видно, что на тренировочных данных точность модели 91,9%, на тестовых — 89,5%.

Чтобы исследовать поведение сети при различных размерах вектора текста, был изменен размер словаря.

1) Размер словаря = 5000, рис 2-3

2) Размер словаря = 1000, рис 5-6

3) Размер словаря = 500, рис 7-8

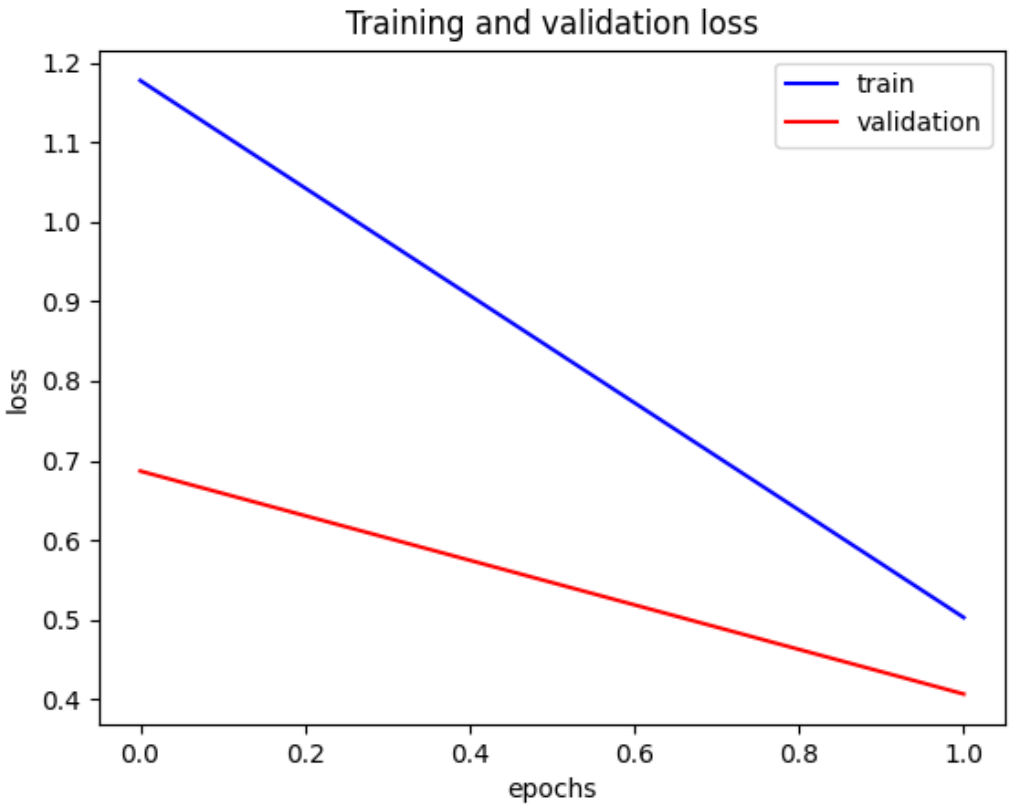

Рисунок 3 — График точности при 5000 слов

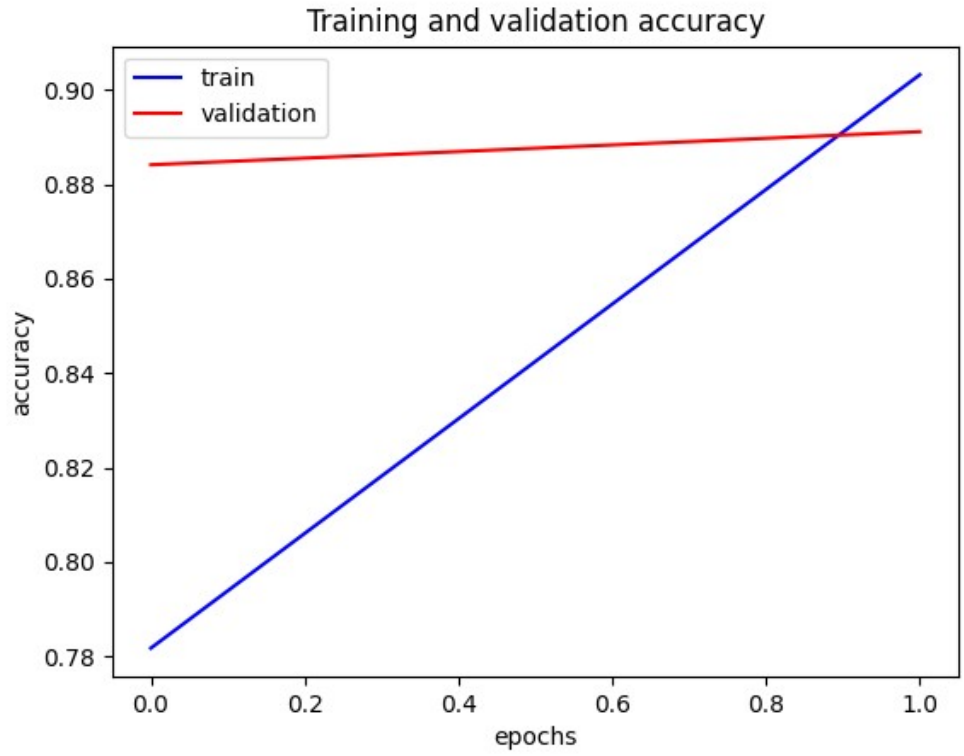

Рисунок 4 — График точности при 5000 слов

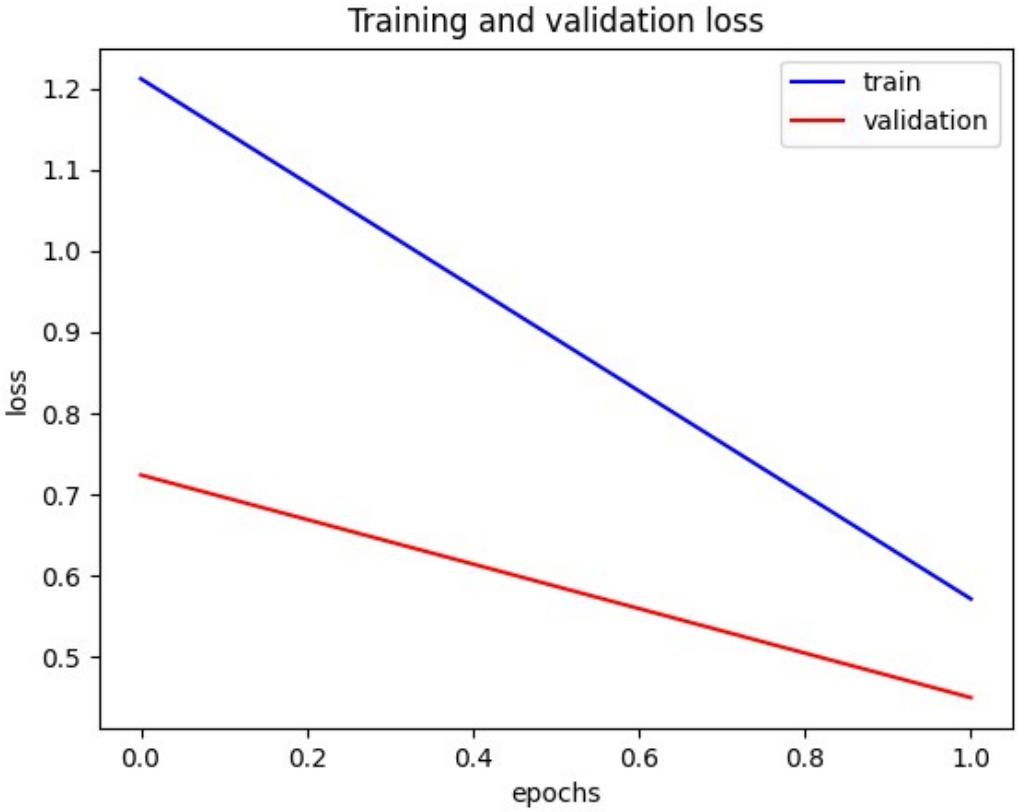

Рисунок 5 — График потерь при 1000 слов

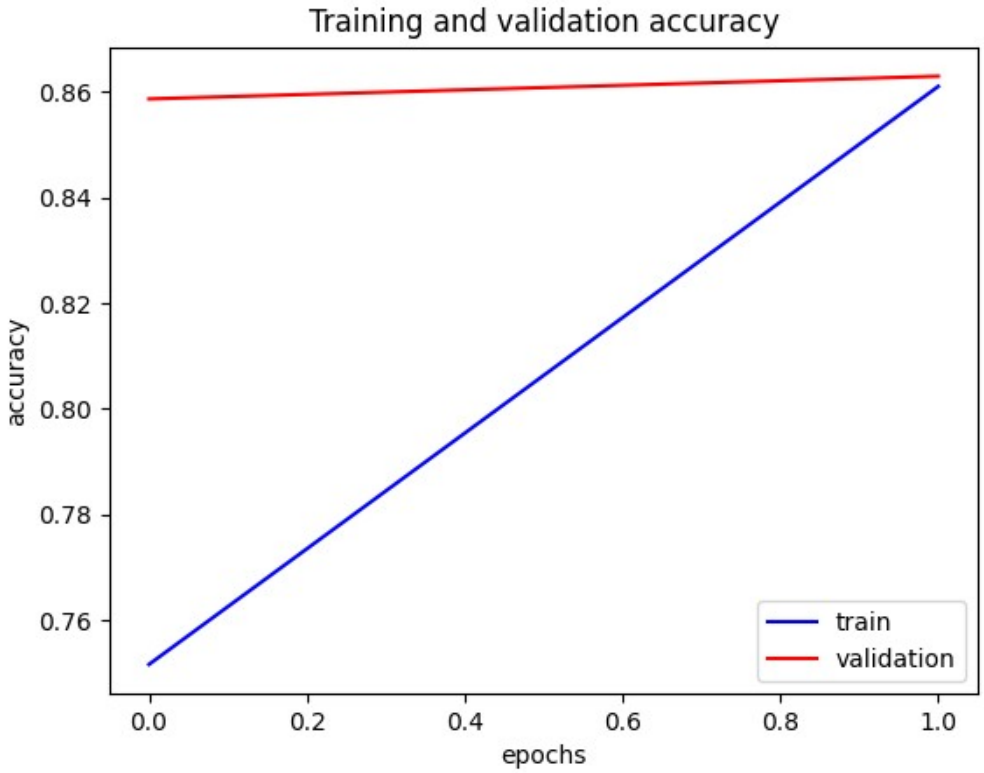

Рисунок 6 — График точности при 1000 слов

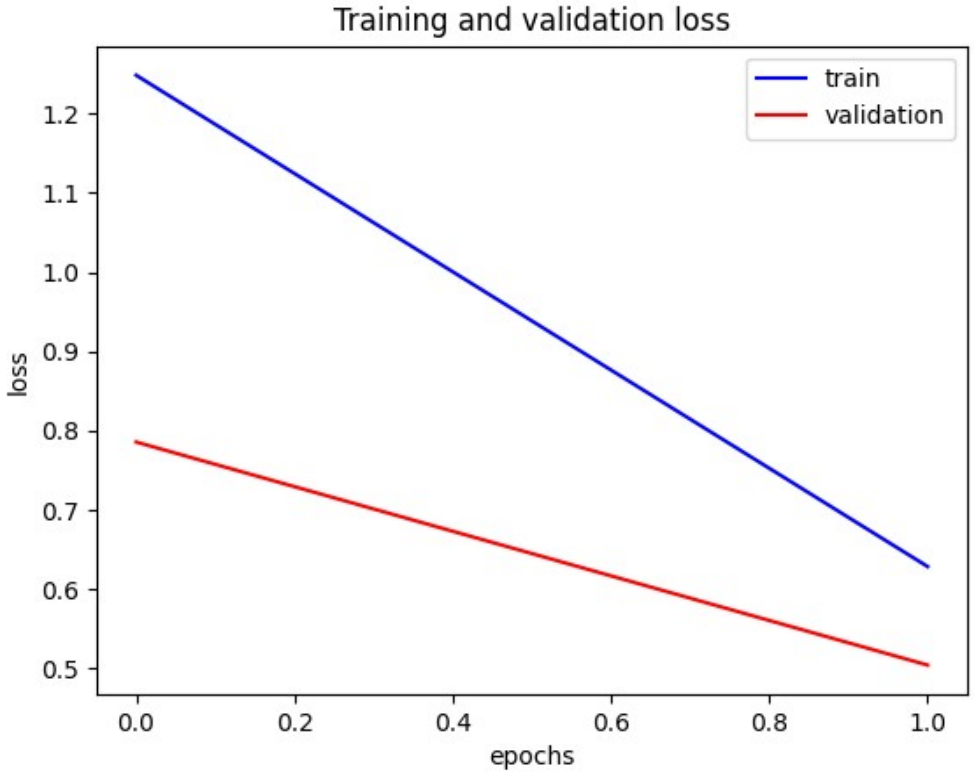

Рисунок 7 — График потерь при 500 словах

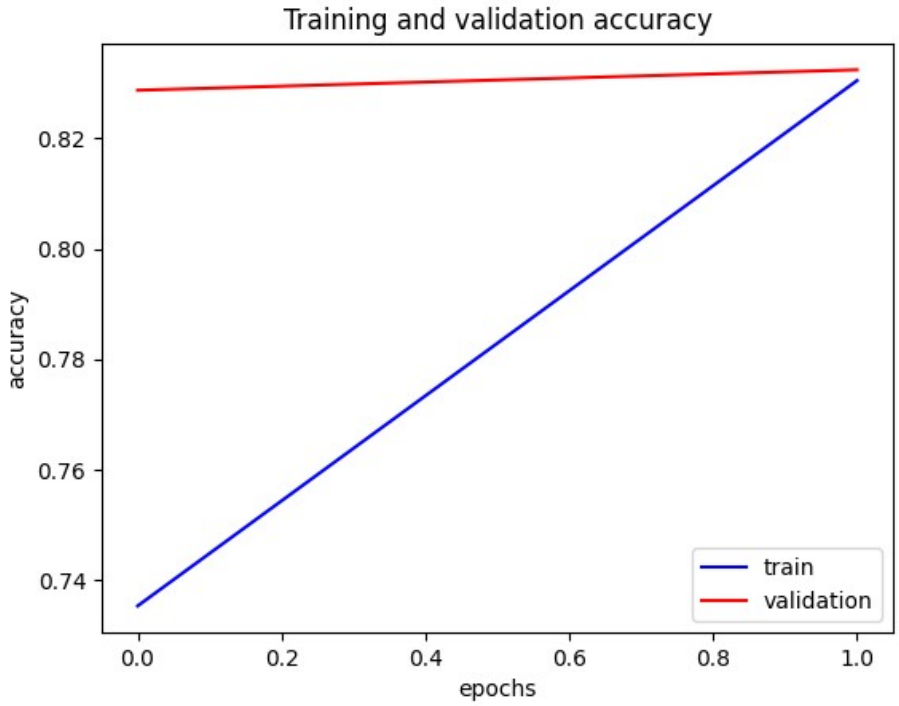

Рисунок 8 — График точности при 500 словах

При размере словаря в 5000 слов точность тренировочных данных упала до 90%, точность тестовых данных — 88,7%. При размере словаря в 1000 слов точность снизилась до 86%, а при размере словаря в 500 слов — до 83%. Это объясняется тем, что при уменьшении словаря меняется эмоциональная окраска образов, из-за чего сеть не может точно их классифицировать.

Чтобы ввести пользовательский текст, была написана специальная функция. Результат представлен на рис. 9.

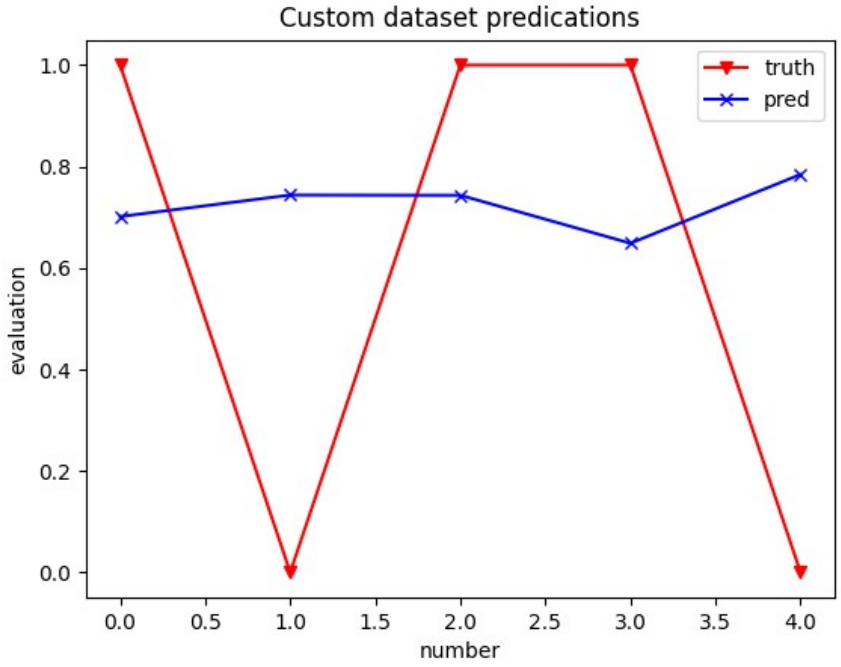

Рисунок 9 - График точности пользовательской оценки

Как видно из графика на рис. 9 — сеть угадала 3 из 5 оценок, т.е. точность оценки примерно 60%.

Вывод.

В ходе лабораторной работы была изучена задача классификации обзоров из IMDb. Была подобрана архитектура, которая дает точность 89,5%. Было выяснено, что при уменьшении максимального размера словаря снижается точность, т.к. по ограниченному словарному запасу нельзя точно угадать эмоциональную окраску слова. Функция для пользовательского текста дала точность результата в 60%.

ПРИЛОЖЕНИЕ А

Исходный код программы

```
import matplotlib.pyplot as plt
import numpy as np
from keras import Sequential, regularizers
from keras.datasets import imdb
from keras.layers import Dense, Dropout
from keras.optimizers import Adam


(X_train, y_train), (X_test, y_test) = imdb.load_data()


(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=500)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)
index = imdb.get_word_index()
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join([reverse_index.get(i - 3, "#") for i in data[0]])
print(decoded)


def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results



my_x = [
        "The film is excellent",
        "This picture is very bad",
        "It is a very good film",
        "Interesting film, amazing scenario",
        "Very boring film"
]
my_y = [1., 0., 1., 1., 0.]



def gen_my_x(my_x, word_index):
```

```
    def get_index(a, index):
        new_list = a.split()
        for i, v in enumerate(new_list):
            new_list[i] = index.get(v)
        return new_list
    for i in range(len(my_x)):
        my_x[i] = get_index(my_x[i], word_index)
    return my_x



my_x = gen_my_x(my_x, imdb.get_word_index())
for index_j, i in enumerate(my_x):
    for index, value in enumerate(i):
        if value is None:
            my_x[index_j][index] = 0


data = vectorize(data)
targets = np.array(targets).astype("float32")
my_y = np.asarray(my_y).astype("float32")


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



model = Sequential()
model.add(Dense(50, activation="relu", input_shape=(10000,)))
model.add(Dropout(0.2, noise_shape=None, seed=None))
model.add(Dense(50, activation="linear", kernel_regularizer=regularizers.l2()))
model.add(Dropout(0.5, noise_shape=None, seed=None))
model.add(Dense(100, activation="relu", kernel_regularizer=regularizers.l2()))
model.add(Dropout(0.5, noise_shape=None, seed=None))
model.add(Dense(50, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
model.compile(Adam(), loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(
          train_x,
          train_y,
```

```
            batch_size=500,
            epochs=2,
            verbose=1,
            validation_data=(test_x, test_y)
        )
H = history


loss = H.history['loss']
v_loss = H.history['val_loss']
plt.plot(loss, 'b', label='train')
plt.plot(v_loss, 'r', label='validation')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend()
plt.show()
plt.clf()


acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
plt.plot(acc, 'b', label='train')
plt.plot(val_acc, 'r', label='validation')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend()
plt.show()
plt.clf()


a, acc = model.evaluate(test_x, test_y)
print('Test', acc)


my_x = vectorize(my_x)


my_loss, my_acc = model.evaluate(my_x, my_y)
print('model_accuracy:', my_acc)
preds = model.predict(my_x)
plt.title("Custom dataset predications")
plt.plot(my_y, 'r', marker='v', label='truth')
```

```
plt.plot(preds, 'b', marker='x', label='pred')
plt.ylabel('evaluation')
plt.xlabel('number')
plt.legend()
plt.show()
plt.clf()
```