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

Студент гр.7382 _____ Токарев А.П.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Порядок выполнения работы.

1. Ознакомиться с задачей регрессии.
2. Изучить способы представления текста для передачи в ИНС.
3. Достигнуть точность прогноза не менее95%.

Требования к выполнению задания.

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

Основные теоретические положения.

Датасет IMDb состоит из 50 000 обзоров фильмов от пользователей, помеченных как положительные (1) и отрицательные (0). Это пример бинарной или двуклассовой классификации, важный и широко применяющийся тип задач машинного обучения.

1. Рецензии предварительно обрабатываются, и каждая из них кодируется последовательностью индексов слов в виде целых чисел.
2. Слова в обзорах индексируются по их общей частоте появления в датасете. Например, целое число «2» кодирует второе наиболее частое используемое слово.
3. 50 000 обзоров разделены на два набора: 25 000 для обучения и 25 000 для тестирования.

Ход работы.

1. Была построена и обучена нейронная сеть для обработки текста. Код предоставлен в приложении А. Данная архитектура дает точность: на тренировочной выборке ~ 82,3%, на контрольной ~ 83%. Графики точности и ошибки предоставлены на рис. 1 и рис. 2 соответственно.

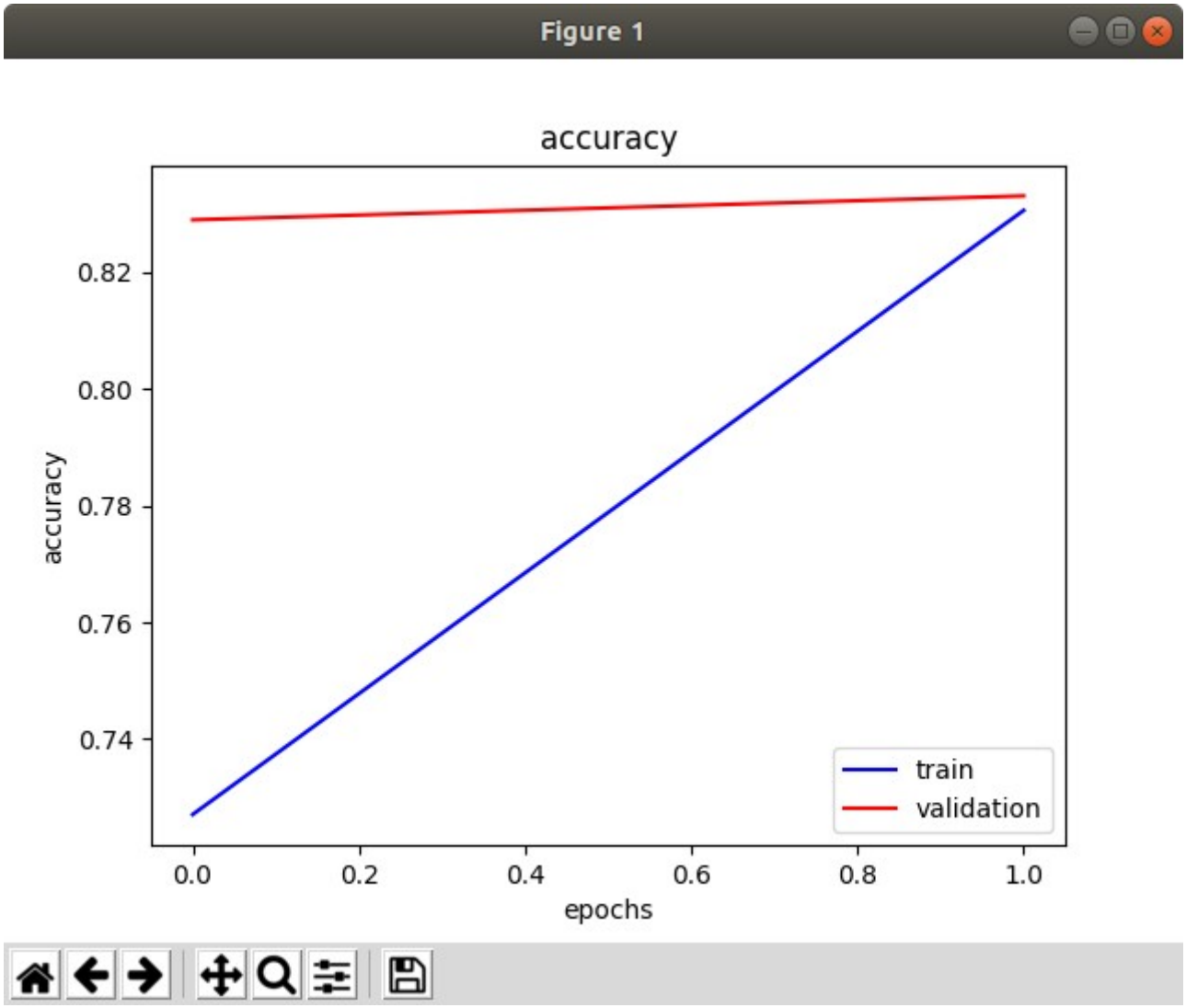

Рисунок 1 – График точности при размере словаря 10 тыс. обзоров

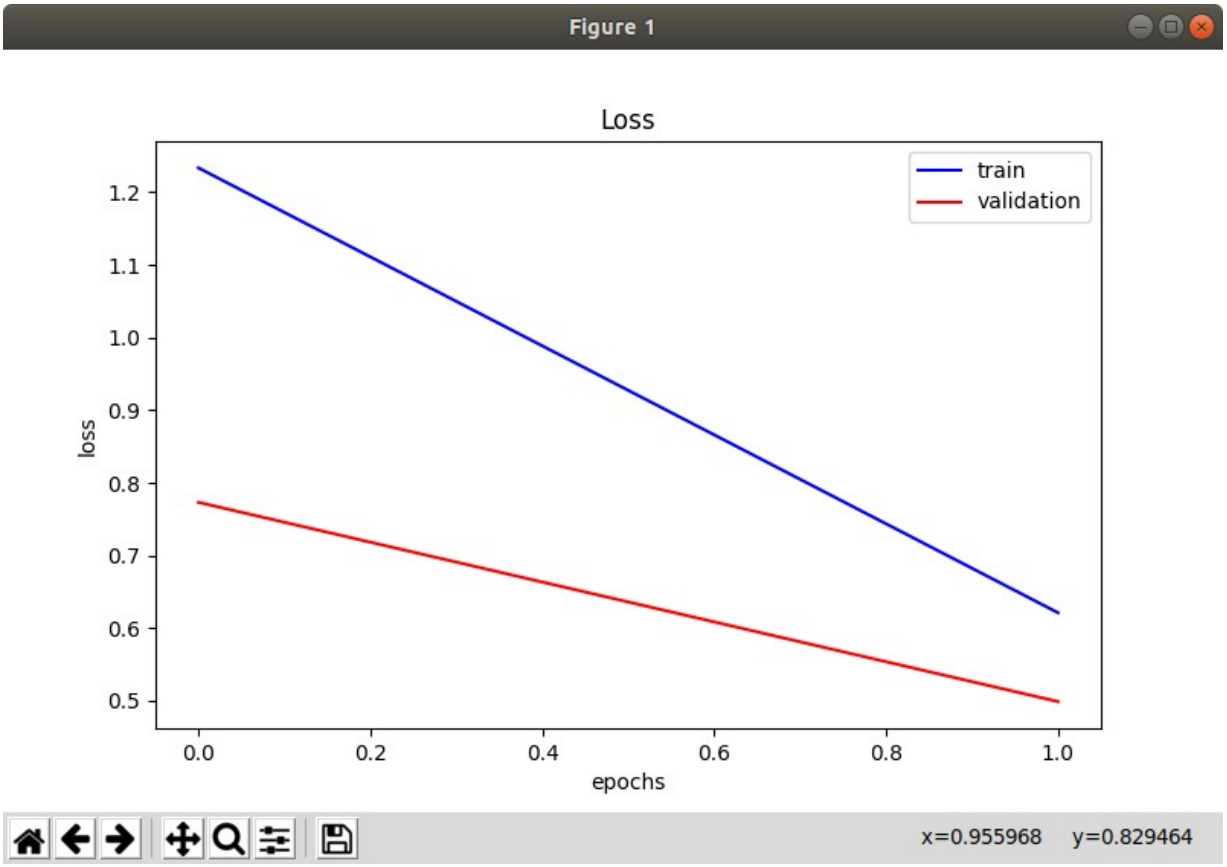

Рисунок 2 – График потерь при размере словаря 10 тыс. обзоров

2. Исследуем результаты при различном размере вектора представления текста.

Графики точности и ошибки предоставлены на рис. 3 и рис. 4 соответственно.

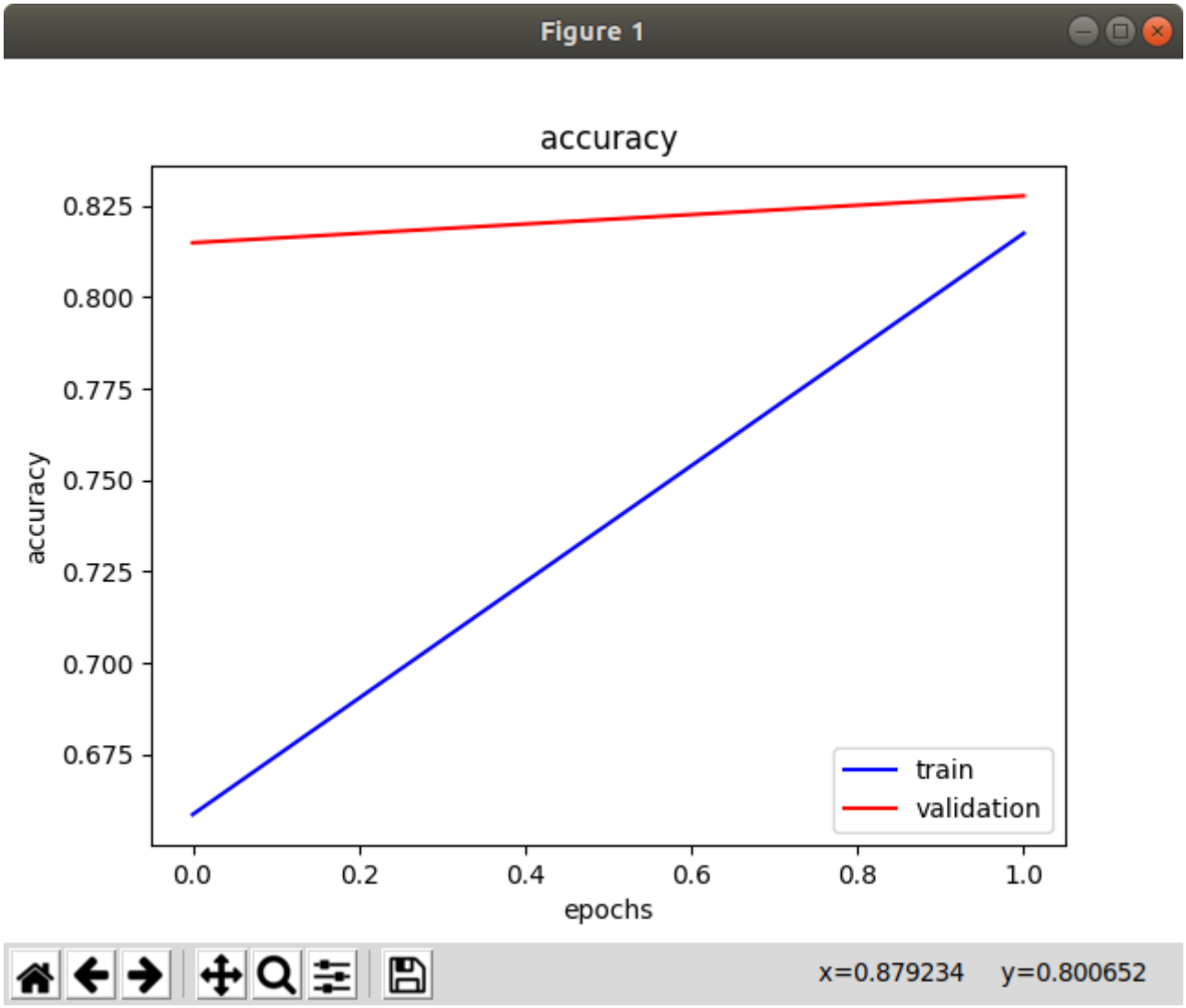

Рисунок 3 – График точности при размере словаря 1 тыс. обзоров

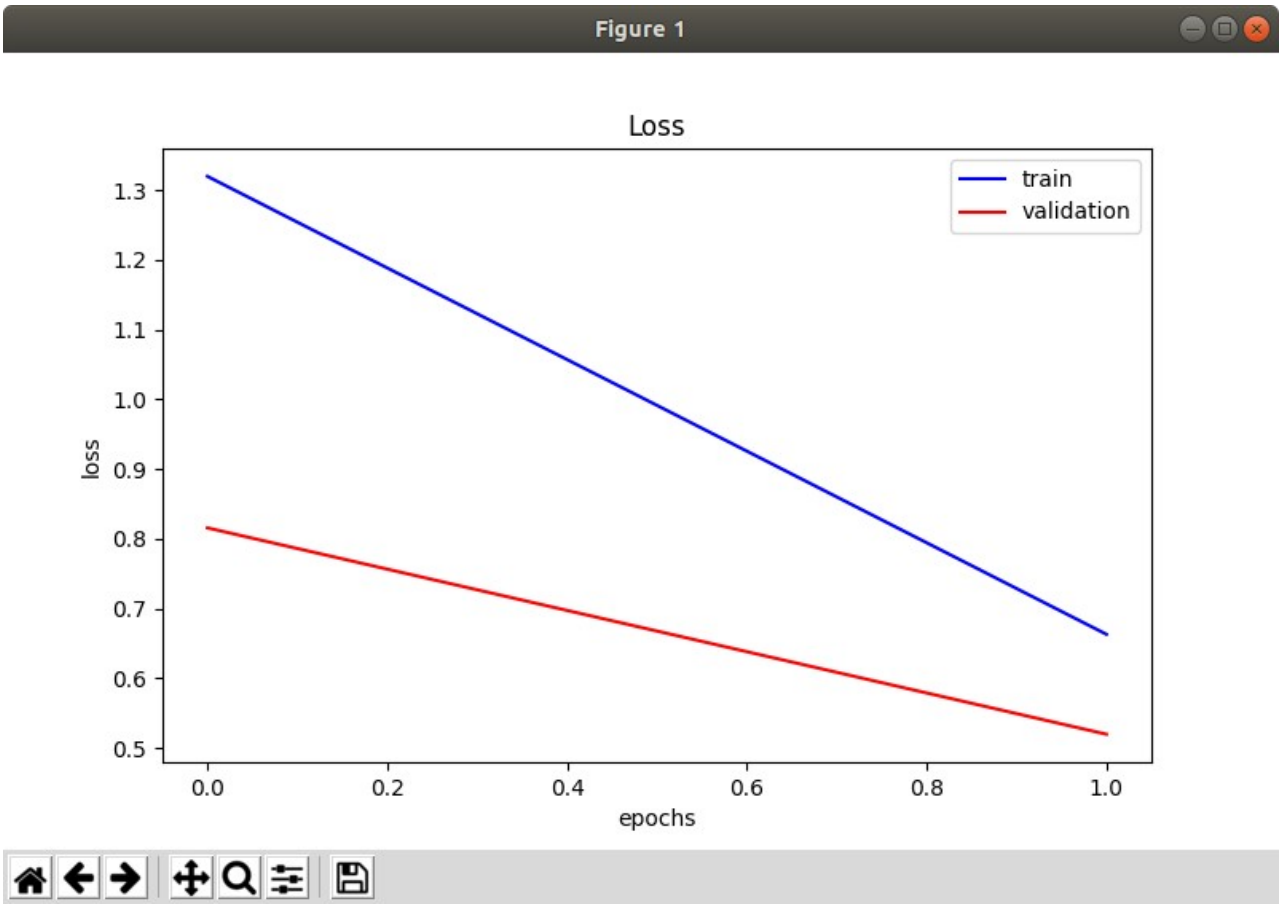

Рисунок 4 – График потерь при размере словаря 1 тыс.

Напишем функцию, которая позволяет ввести пользовательский текст.

```
def gen_custom_x(custom_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(custom_x)):
        custom_x[i] = get_index(custom_x[i], word_index)
    return custom_x
```

При помощи данной функции можно получить из массива строк (обзоров) массив представлений в виде индексов слов в imdb датасете и подготовленные для прогона через модель. График точности оценки фильма, при прогоне через написанный датасет из 5 обзоров (см. рис. 7), предоставлена на рис 8.

Рисунок 7 – Пользовательский текст

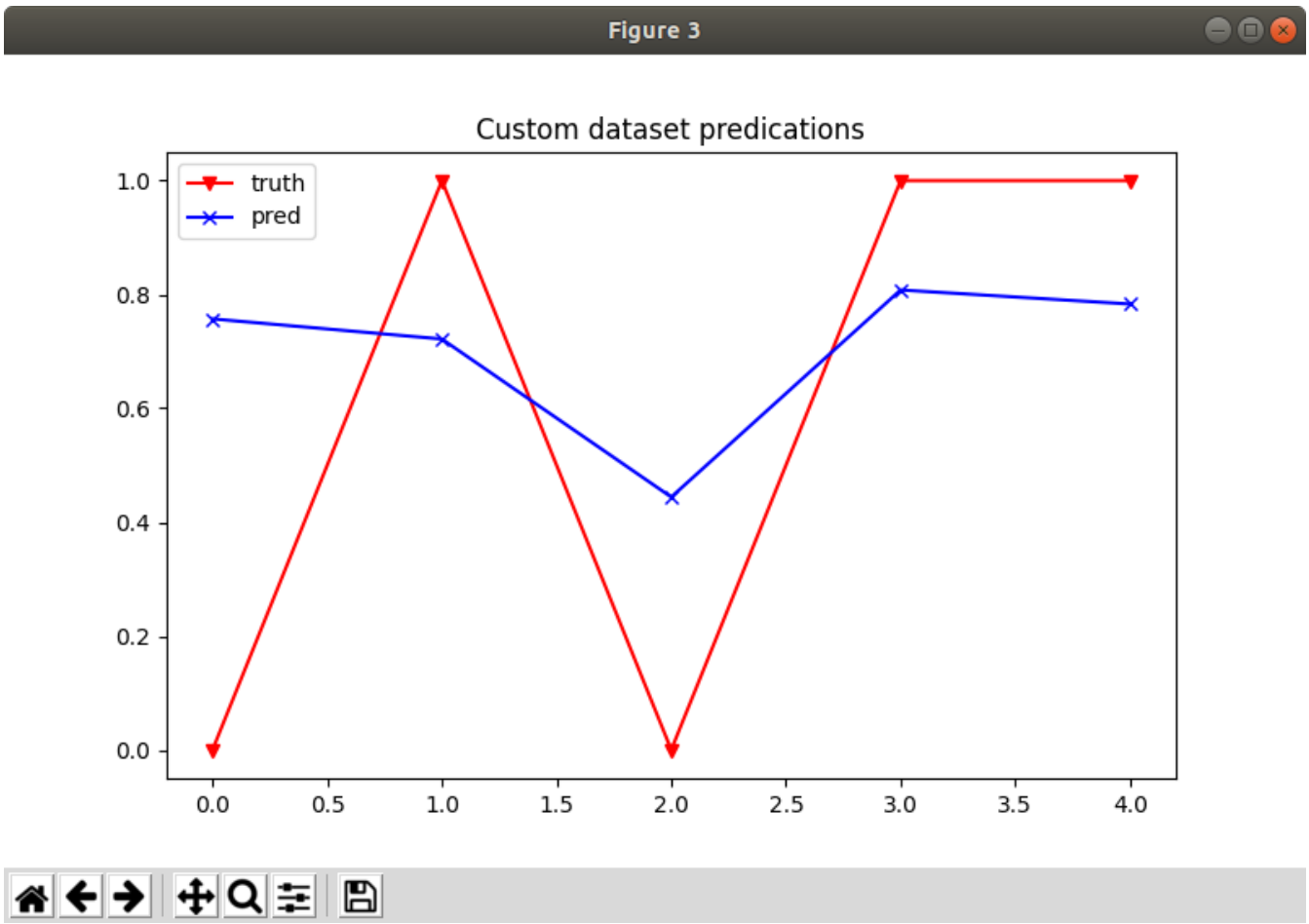

Рисунок 8 – График точности оценки фильма

Из графика на рис. 8 видно, что точность оценки фильма ~ 60%.

Выводы.

В ходе работы была изучена задача классификация обзоров из датасета IMDB. Подобрана архитектура, дающая точность 83%. Функция для подготовки вручную введенных обзоров, продемонстрировала точность 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 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()
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()
def vectorize(sequences, dimension=10000):
	results = np.zeros((len(sequences), dimension))
	for i, sequence in enumerate(sequences):
		results[i, sequence] = 1
	return results
custom_x = [
		"it is very boring, bad film",
			"fantastic, the best film ever",
			"i think this film is the worst film i've seen, nothing interesting, junk",
			"fantastic film, wonderful casting, good job, creators",
			"beautiful picture, good scenario, it's amazing"
]
custom_y = [0., 1., 0., 1., 1.]
def gen_custom_x(custom_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(custom_x)):
		custom_x[i] = get_index(custom_x[i], word_index)
	return custom_x
custom_x = gen_custom_x(custom_x, imdb.get_word_index())
for index_j, i in enumerate(custom_x):
	for index, value in enumerate(i):
		if value is None:
```

```
                    custom_x[index_j][index] = 0
data = vectorize(data)
targets = np.array(targets).astype("float32")
custom_y = np.asarray(custom_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
plot_loss(H.history['loss'], H.history['val_loss'])
plot_acc(H.history['accuracy'], H.history['val_accuracy'])
a, acc = model.evaluate(test_x, test_y)
print('Test', acc)
custom_x = vectorize(custom_x)
custom_loss, custom_acc = model.evaluate(custom_x, custom_y)
print('custom_acc:', custom_acc)
preds = model.predict(custom_x)
plt.figure(3, figsize=(8,5))
plt.title("Custom dataset predications")
plt.plot(custom_y, 'r', marker='v', label='truth')
plt.plot(preds, 'b', marker='x', label='pred')
plt.legend()
plt.show()
plt.clf()
```