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

ОТЧЕТ

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

Студент гр. 7382 _____ Гиззатов А.С.

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.

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 отчете 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.

- 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

Ход работы.

Была написана и обучена нейронная сеть, исходный код предоставлен в приложении А.

Архитектура:

Модель:

```
model = Sequential()
model.add(layers.Dense(50, activation="relu", input_shape=(10000,)))
# Hidden - Layers
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(100, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
# Output- Layer
model.add(layers.Dense(1, activation="sigmoid"))
```

Параметры для компиляции:

```
model.compile(
    optimizer="adam",
    loss="binary_crossentropy",
    metrics=["accuracy"])
```

1. Данная архитектура дает точность примерно чуть более 90% на тренировочных данных и около 89% на тестовых данных.

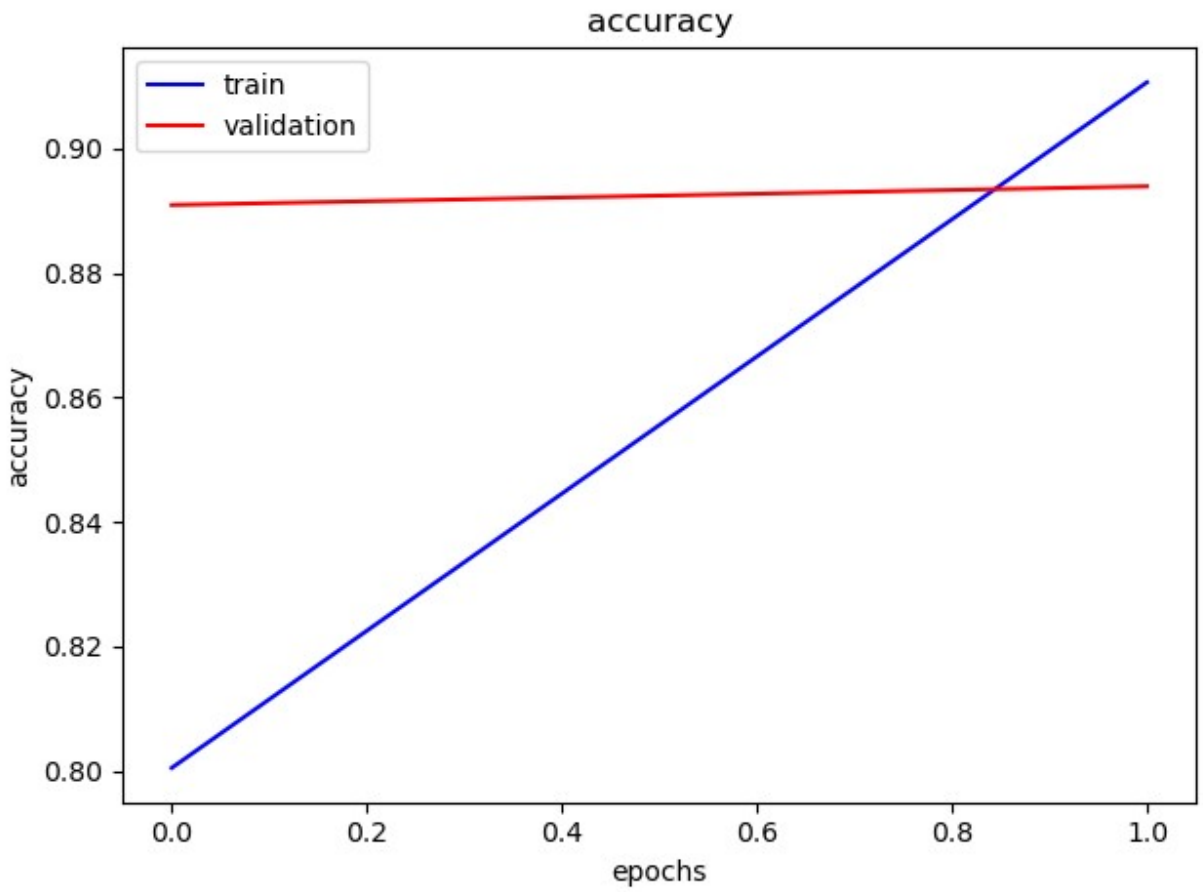

Рис.1 - Точность НС при векторе 10000

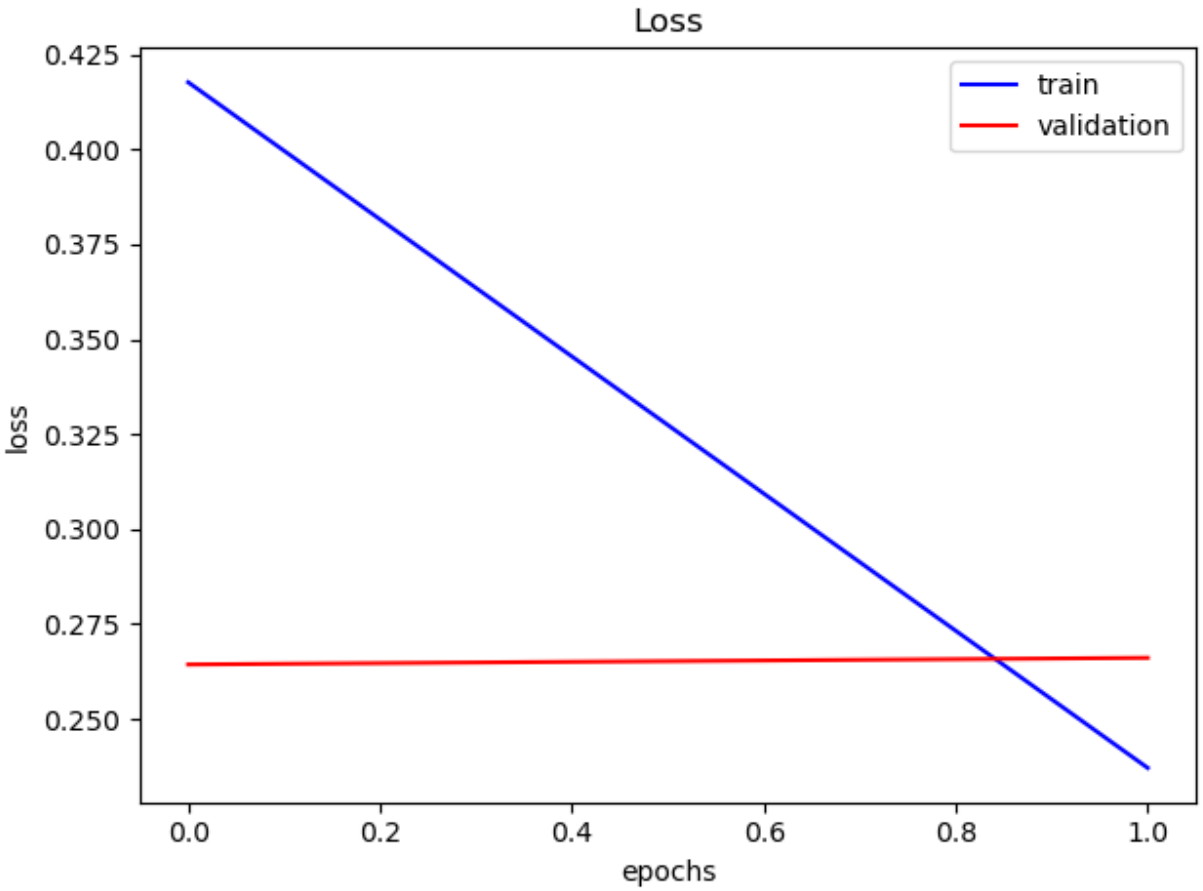

Рис.2 - Потери НС при векторе 1000

2. Рассмотрим результаты при различном размере вектора представлении текста.

При изменении максимального количества слов в словаре с 10 тыс. до 1 тыс. точность немного упала, а ошибка возросла изменения представлены на рис.3 и 4.

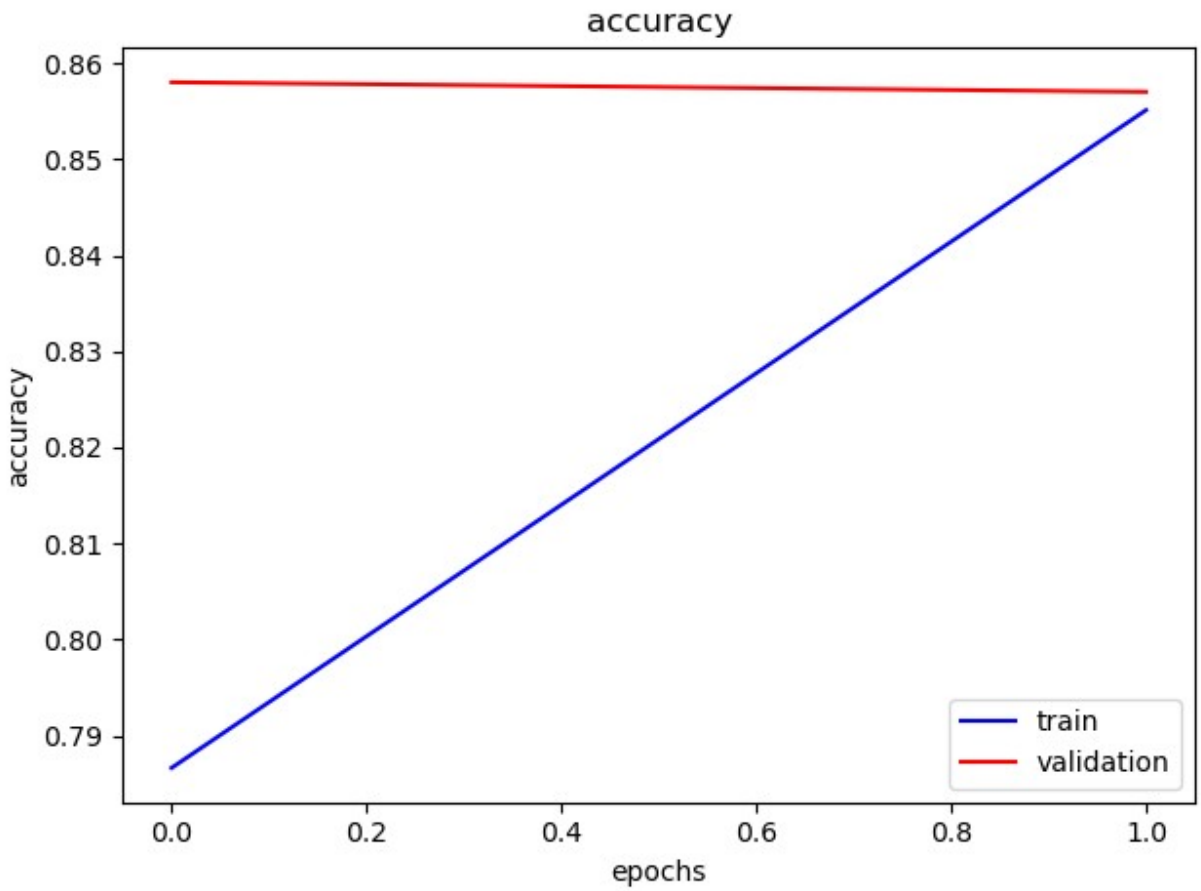

рис.3 – Точность НС при векторе 1000

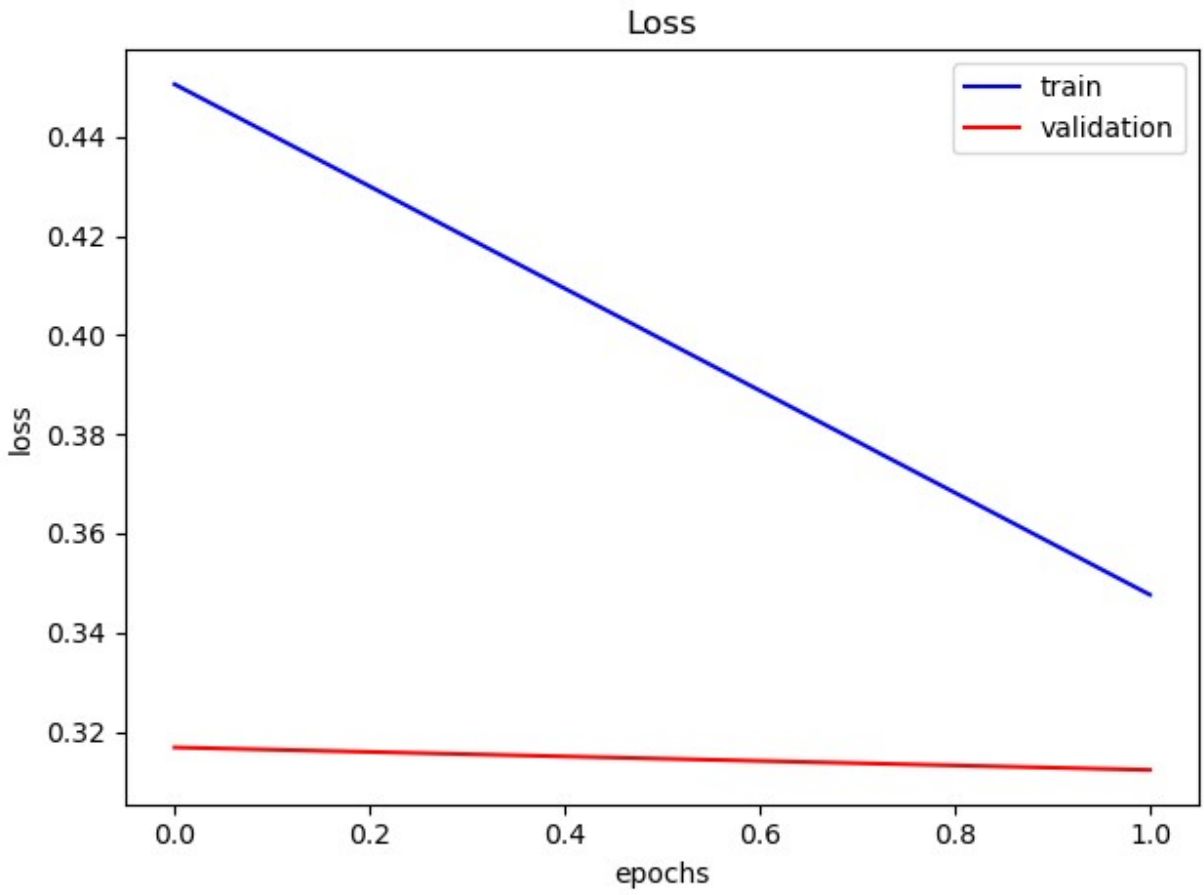

рис.4 – Потери НС при векторе 1000

Из рис.1-4 можно сделать вывод, что при уменьшении словаря точность будет падать, а ошибка будет расти. Это связано с тем, что при уменьшении данных нейронная сеть обучается не на всех возможных вариантах обзора и поэтому ей сложно проводить классификацию.

3. Напишем функцию для ввода пользовательского обзора.

```
def index(str, indx):
    lst = str.split()
    for i, w in enumerate(lst):
        lst[i] = indx.get(w)
    return lst


def user_text(str, indx):
    for i in range(len(str)):
        str[i] = index(str[i], indx)
    return str
```

и создадим массив обзоров

```
user_txt = [
    "The movie affects you in a way that makes it physically painful to
experience, but in a good way",
    "Seriously, unbelievable",
    "One of the best movies in these few years",
```

```
    "It is very boring film",
    "it's very good",
    "it's very boring",
    "fantastic film, wonderful casting, good job, creators",
    "beautiful picture, good scenario, it's amazing",
    "Absolute disappointment. I believed the hype like a fool"
]
user_y = [1., 1., 1., 0., 1., 0., 1., 1., 0.]
```

При помощи данной функции можно получить из массива 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 9 обзоров предоставлена на рис. 5.

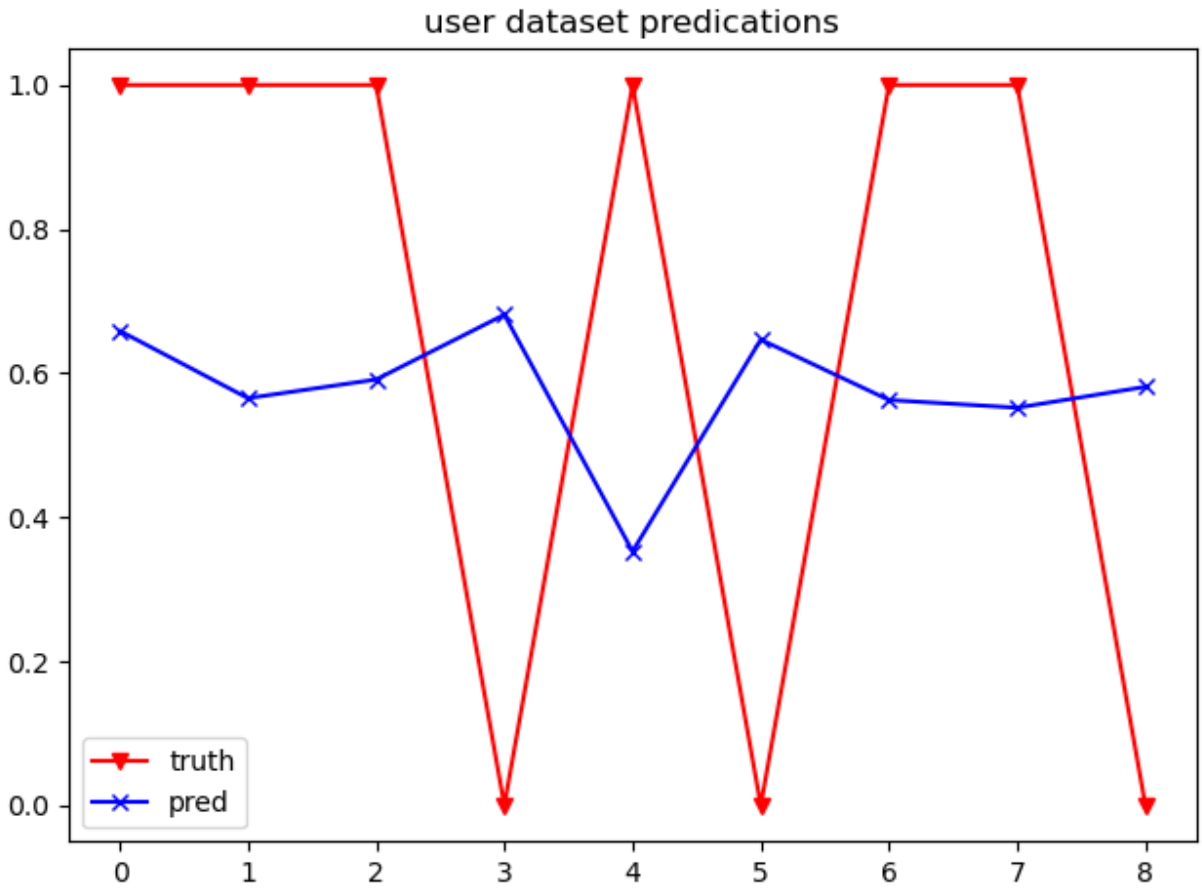

Рис.5 – прогноз пользовательских обзоров

Выводы.

В ходе работы была изучена задача классификация обзоров из датасета IMDB. Подобрана архитектура, дающая точность 89,1%. Было выяснено, что при уменьшении максимального размера словаря точность уменьшается, что логично, так как мы убираем часть «словарного запаса».

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД LAB6.PY

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

(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
print(training_data)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)
# print("Categories:", np.unique(targets))
# print("Number of unique words:", len(np.unique(np.hstack(data))))
length = [len(i) for i in data]
# print("Average Review length:", np.mean(length))
# print("Standard Deviation:", round(np.std(length)))
# print("Label:", targets[0])
# print(data[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)

user_txt = [
    "The movie affects you in a way that makes it physically painful to
experience, but in a good way",
    "Seriously, unbelievable",
    "One of the best movies in these few years",
    "It is very boring film",
    "it's very good",
    "it's very boring",
    "fantastic film, wonderful casting, good job, creators",
    "beautiful picture, good scenario, it's amazing",
    "Absolute disappointment. I believed the hype like a fool"
]
user_y = [1., 1., 1., 0., 1., 0., 1., 1., 0.]


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


def index(str, indx):
    lst = str.split()
    for i, w in enumerate(lst):
        lst[i] = indx.get(w)
    return lst


def user_text(str, indx):
```

```
    for i in range(len(str)):
        str[i] = index(str[i], indx)
    return str


def set_zero(str):
    for i, j in enumerate(str):
        for k, value in enumerate(j):
            if value is None:
                str[i][k] = 0
    return str


user_x = user_text(user_txt, imdb.get_word_index())
set_zero(user_txt)

data = vectorize(data)
targets = np.array(targets).astype("float32")
test_x = data[:10000]
test_y = targets[:10000]
train_x = data[10000:]
train_y = targets[10000:]
# print(train_x[:1000])
# Input - Layer
model = Sequential()
model.add(layers.Dense(50, activation="relu", input_shape=(10000,)))
# Hidden - Layers
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
model.add(layers.Dense(100, activation="relu"))
model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
# Output- Layer
model.add(layers.Dense(1, activation="sigmoid"))
model.summary()
model.compile(
    optimizer="adam",
    loss="binary_crossentropy",
    metrics=["accuracy"])
H = model.fit(
    train_x, train_y,
    epochs=2,
    batch_size=150,
    validation_data=(test_x, test_y)
)
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('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('accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend()
plt.show()
plt.clf()

user_x = vectorize(user_x)

user_loss, user_acc = model.evaluate(user_x, user_y)

print('user_acc:', user_acc)
preds = model.predict(user_x)
plt.title("user dataset predications")
plt.plot(user_y, 'r', marker='v', label='truth')
plt.plot(preds, 'b', marker='x', label='pred')
plt.legend()
plt.show()
plt.clf()
```