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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: «Прогноз успеха фильмов по обзорам»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Порядок выполнения работы.

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

Требования.

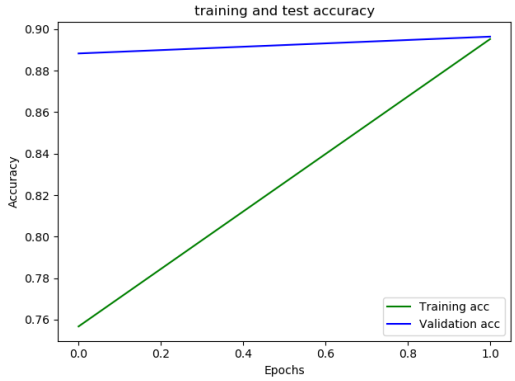
- Построить и обучить нейронную сеть для обработки текста
- Исследовать результаты при различном размере вектора представления текста
- Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте)

Ход работы.

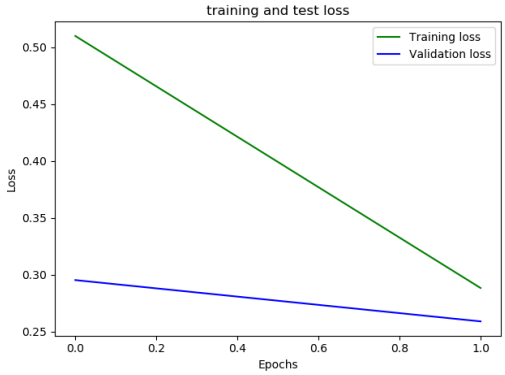
Набор данных IMDB – множество из 50 000 самых разных отзывов к кинолентам в интернет-базе фильмов (Internet Movie Database). Набор разбит на 25 000 обучающих и 25 000 контрольных отзывов, каждый набор на 50 % состоит из отрицательных и на 50 % из положительных отзывов.

1. Найдем наилучшую модель. В ходе исследования было выяснено, что такие манипуляции с сетью, как увеличение/уменьшение количества нейронов, добавление скрытых слоёв и добавление слоёв прореживания не дают особый выигрыш в точности, поэтому остановимся на наиболее простой модели, которая даёт точность 89.5%.
2. Была построена нейронная сеть со следующей архитектурой:
    - Оптимизатор – adam
    - `batch_size=512`
    - `loss='binary_crossentropy'`
    - `epochs=2`

Точность ~89.5%



а

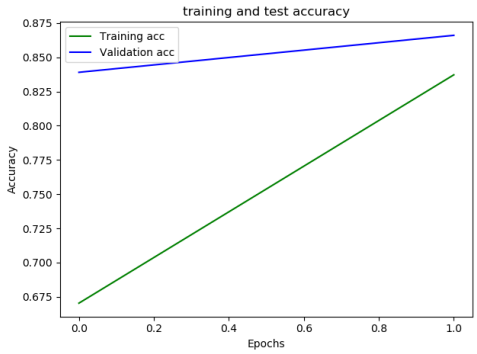


б

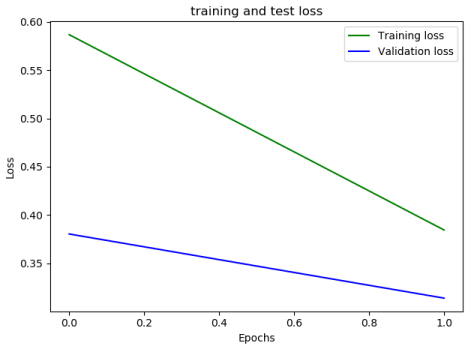
Рисунок 1 – Графики точности и потерь данной архитектуры

3. Исследуем результаты при различном размере вектора представления текста

Графики при размере 1000 образов, точность 86.6%:



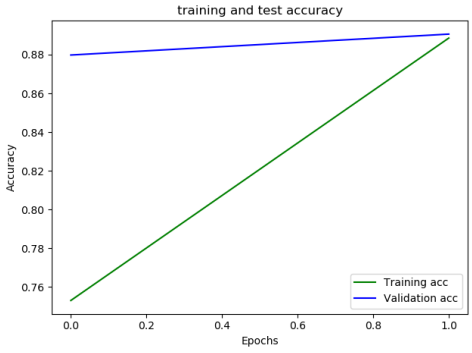
а



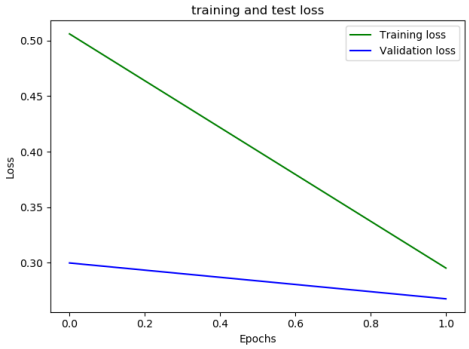
б

Рисунок 2 – Графики точности и потерь при 1000 образах

Графики при размере 5000 образов, точность 89%:



а



б

Рисунок 3 – Графики точности и потерь при 5000 образах

Как видим, точность уменьшается с уменьшением вектора. Это происходит из-за того, что мы выкидываем слова из обзоров, оставляя лишь самые часто употребляемые, с которыми сеть очевидно теряет в точности.

4. Напишем функцию для ввода пользовательского текста:

```
def gen_custom_x():
    def get_index(a, _index):
        new_list = a.split()
        for j, v in enumerate(new_list):
            new_list[j] = _index.get(v)
        return new_list

    for i in range(len(custom_x)):
        custom_x[i] = get_index(custom_x[i], 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
    return custom_x
```

Используем нашу функцию для следующих оценок:

```
custom_x = [
    "An excellent, well-researched film",
    "It is, perhaps, most playful and ambitious film to date",
    "This is the boring, stupid movie",
    "You don't really care who done it, you just want it to be over
with",
    "Great movie with a great story"
]
```

Получили следующие результаты:

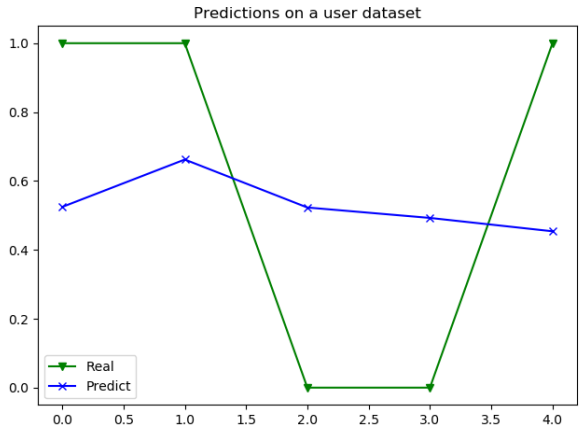


Рисунок 4 – Предсказания пользовательского датасета.

Точность составила 60%, то есть 3/5 оценок было угадано.

Выводы.

В ходе выполнения данной работы построили и обучили нейронную сеть для обработки текста, изучили способы представления текста для передачи в ИНС, а также исследовали результаты работы ИНС при различном размере вектора представления текста.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
import matplotlib.pyplot as plt
import numpy as np
from keras import layers, Sequential
from keras.datasets import imdb


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 = [
    "An excellent, well-researched film",
    "It is, perhaps, most playful and ambitious film to date",
    "This is the boring, stupid movie",
    "You don't really care who done it, you just want it to be
over with",
    "Great movie with a great story"
]
custom_y = [1., 1., 0., 0., 1.]


def gen_custom_x():
    def get_index(a, _index):
        new_list = a.split()
        for j, v in enumerate(new_list):
            new_list[j] = _index.get(v)
        return new_list

    for i in range(len(custom_x)):
        custom_x[i] = get_index(custom_x[i],
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
    return custom_x
```

```
    (training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=10000)
    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)

    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:]
    model = Sequential()

    model.add(layers.Dense(16, activation="relu",
input_shape=(10000,)))
    model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
    model.add(layers.Dense(16, activation="relu"))
    model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
    model.add(layers.Dense(16, activation="relu"))
    model.add(layers.Dense(1, activation="sigmoid"))

    model.compile(
        optimizer="adam",
        loss="binary_crossentropy",
        metrics=["accuracy"]
    )
    h = model.fit(
        train_x, train_y,
        epochs=2,
        batch_size=512,
        verbose=0,
        validation_data=(test_x, test_y)
    )

    print(np.mean(h.history["val_accuracy"]))
    score = model.evaluate(test_x, test_y, verbose=0)
    print('Test loss:', score[0])
```

```
print('Test accuracy:', score[1])

plt.title("training and test accuracy")
plt.plot(h.history['accuracy'], 'g', label='Training acc')
plt.plot(h.history['val_accuracy'], 'b', label='Validation acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.clf()
plt.title("training and test loss")
plt.plot(h.history['loss'], 'g', label='Training loss')
plt.plot(h.history['val_loss'], 'b', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()

X = vectorize(gen_custom_x())
custom_score = model.evaluate(X, custom_y)
print('custom_acc:', custom_score[1])
predict = model.predict(X)
plt.title("Predictions on a user dataset")
plt.plot(custom_y, 'g', marker='v', label='Real')
plt.plot(predict, 'b', marker='x', label='Predict')
plt.legend()
plt.show()
plt.clf()
```