

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

ЛАБОРАТОРНАЯ РАБОТА №6
по дисциплине «Искусственные нейронные сети»
Тема: «Прогноз успеха фильмов по обзорам»

Студент гр. 7381

Минуллин М.А.

Преподаватель

Жукова Н. А.

Санкт-Петербург

2020

Цели.

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.

1. Построить и обучить нейронную сеть для обработки текста.

Была выбрана следующая архитектура:

```
model = models.Sequential()  
model.add(layers.Dense(128, activation="relu",  
input_shape=(10000,)))  
model.add(layers.Dropout(0.3, noise_shape=None, seed=13))  
model.add(layers.Dense(64, activation="relu"))  
model.add(layers.Dropout(0.3, noise_shape=None, seed=15))  
model.add(layers.Dense(48, activation="relu"))  
model.add(layers.Dense(1, activation="sigmoid"))
```

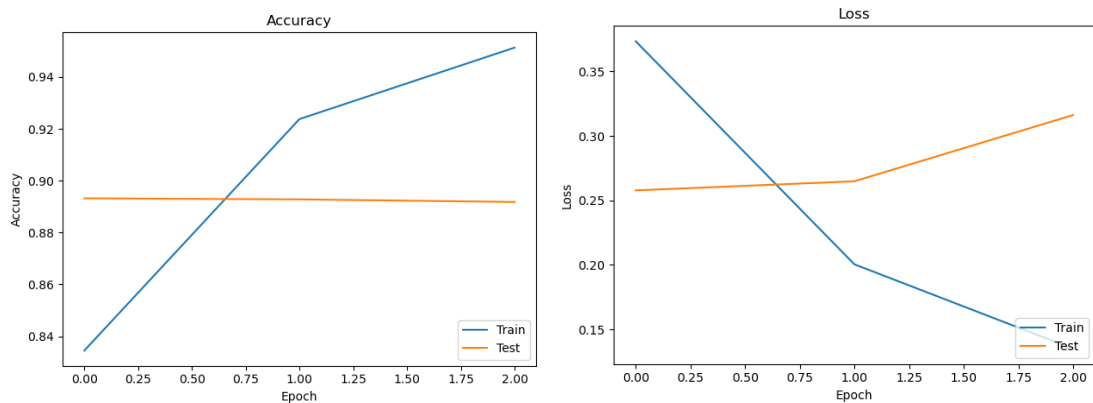


Рисунок 1 —Точность и потери выбранной модели.

Переобучение устранить не удалось, как и повысить точность прогноза.

2. Исследовать результаты при различном размере вектора представления текста.

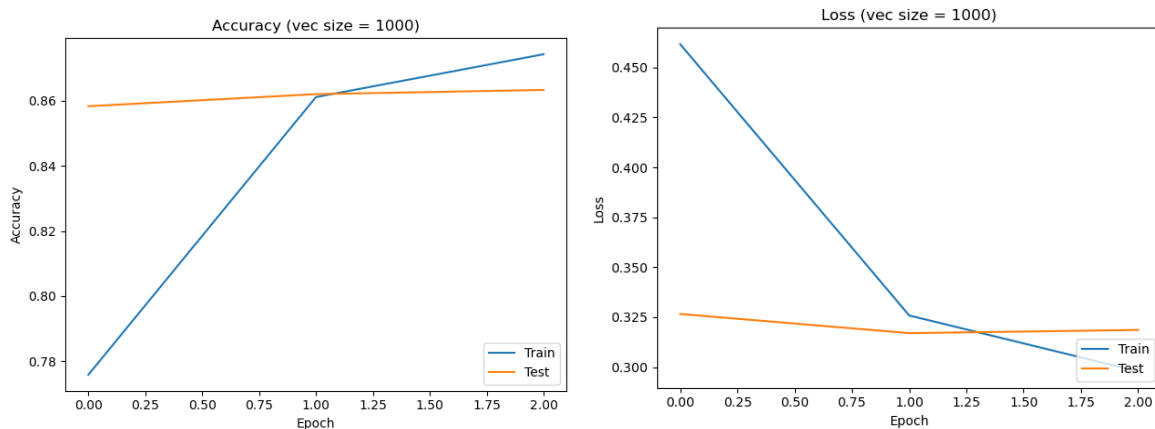


Рисунок 2 — Результаты при размере вектора представления текста равном 1000.

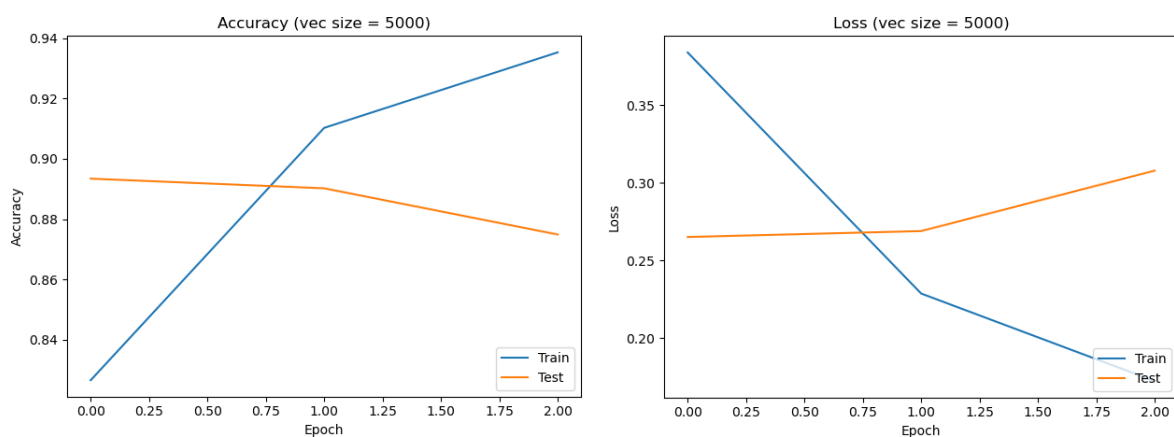


Рисунок 3 — Результаты при размере вектора представления текста равном 5000.

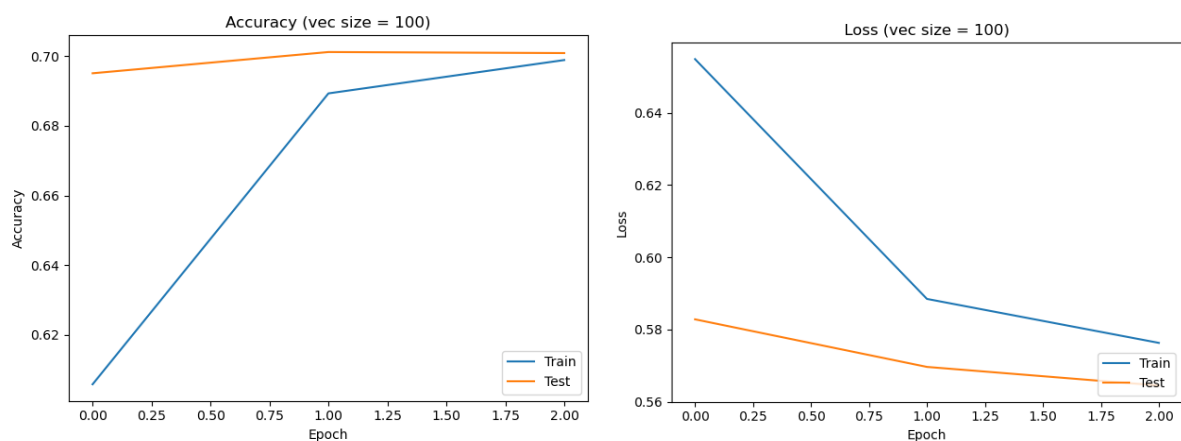
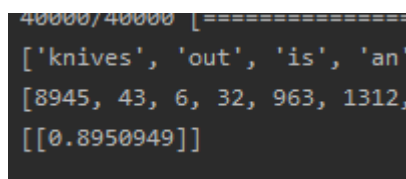


Рисунок 4 — Результаты при размере вектора представления текста равном 100.

Как видно, чем меньше размер вектора представления текста, тем меньше точность.

3. Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте). Для тестирования была написана функция `test()`. Для ревью «Knives Out is an incredibly complex whodunit that skillfully subverts expectations, switches between multiple genres easily and effectively and only gets better with multiple viewings. It's also surprisingly extremely funny and expertly paced. All of it's incredible cast are excellent but Chris Evans, Daniel Craig and Ana De Armas give the standout performances. Rian Johnson's direction is incredible and the film is extremely well filmed. The music by Nathan Johnson is fantastic.» Был получен результат



```
40000/40000 [=====]  
['knives', 'out', 'is', 'an'  
[8945, 43, 6, 32, 963, 1312,  
[[0.8950949]]
```

Рисунок 5 — Результаты теста.

Действительно, отзыв о фильме очень положительный.

Вывод.

В ходе выполнения данной работы было произведен прогноз успеха фильмов по обзорам, ознакомление с задачей регрессии и изучены способы представления текста для передачи в ИНС.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import re

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

def compile_model():
    model = models.Sequential()
    model.add(layers.Dense(128, activation="relu",
input_shape=(10000,)))
    model.add(layers.Dropout(0.3, noise_shape=None, seed=13))
    model.add(layers.Dense(64, activation="relu"))
    model.add(layers.Dropout(0.3, noise_shape=None, seed=15))
    model.add(layers.Dense(48, activation="relu"))
    model.add(layers.Dense(1, activation="sigmoid"))
    model.summary()
    model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])
    return model

def test(text):
    (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)

    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 = compile_model()

    model.fit(
```

```

        train_x, train_y, epochs=3, batch_size=400,
validation_data=(test_x, test_y),
    )
    output_str = ''
    with open(text, 'r') as f:
        for input_str in f.readlines():
            output_str += re.sub('[^A-Za-z0-9 ]+', '',
input_str).lower()
            indexes = imdb.get_word_index()
            encode = []
            text = output_str.split()
            #print(text)
            for index in text:
                if index in indexes and indexes[index] < 10000:
                    encode.append(indexes[index])
            #print(encode)
            text = vectorize([encode])
            result = model.predict(text)
            print(result)

((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)

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)

def vectorize(sequences, dimension=10000):

```

```

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

data = vectorize(data, 10000)
targets = np.array(targets).astype("float32")

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

model = compile_model()
history = model.fit(train_x, train_y, epochs=3, batch_size=400,
                    validation_data=(test_x, test_y),)

plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.title("Accuracy (vec size = 10000)")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Train", "Test"], loc="lower right")
plt.show()

plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Loss (vec size = 10000)")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend(["Train", "Test"], loc="lower right")
plt.show()

test("test.txt")

```