**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 _____ Дорох С.В.

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. Была построена модель с тремя скрытыми слоями по 48 нейронов на каждом, с уменьшением нейронов наблюдалось уменьшение точности. В качестве функции активации использовалась relu. Также присутствует два слоя разреживания с вероятностями 0.3 и 0.2. Модель была скомпилирована со следующей архитектурой:
    - Оптимизатор – adam
    - batch_size=512
    - loss='binary_crossentropy'
    - epochs=3

У модели с текущими параметрами была получены следующие параметры точности и потери, на тестовых данных результат показал точность в 88,5%.

```
40000/40000 [==============================] - 5s 118us/step - loss: 0.1590 - accuracy: 0.9425
Test-Accuracy: 0.8853333393732706
```

Рисунок 1 – результат обучения модели

2. В ходе исследования зависимости размера вектора от результатов обучения было установлено, что с увеличением размера вектора точность повышалась. Подробнее можно увидеть на следующих рисунках:

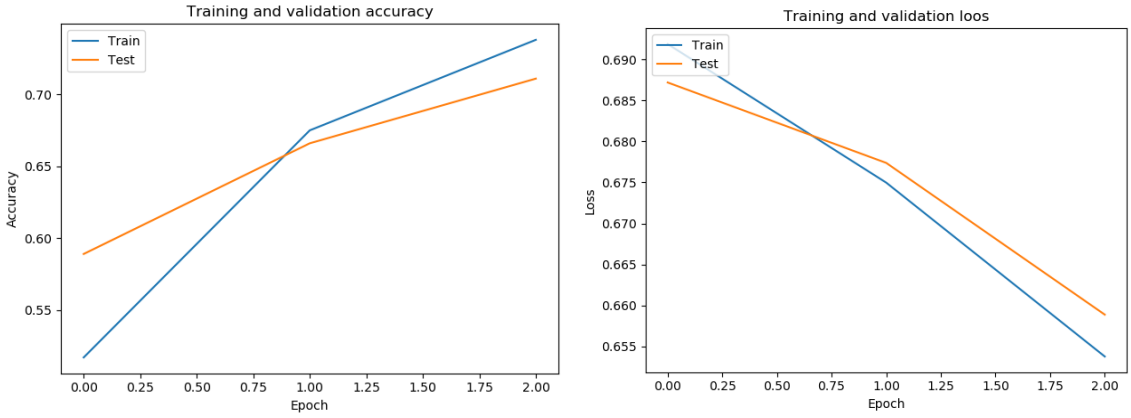


Рисунок 2 – графики точности и потерь при векторе в 1000 образцов

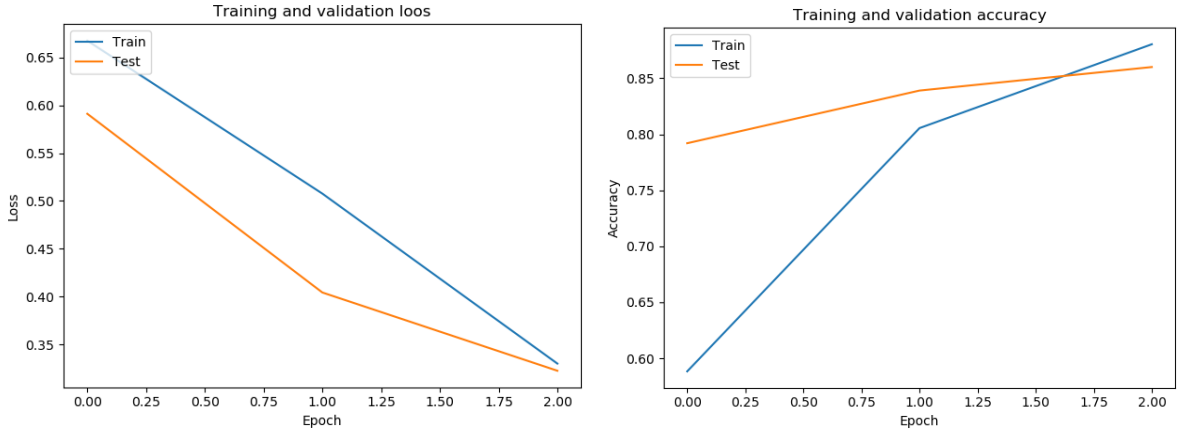


Рисунок 3 – графики точности и потерь при векторе в 5000 образцов

3. Был написан декоратор, позволяющий считывать данные из текста и подготавливать их для корректного обучения модели.

   Для текста «quality action with amazing and exciting stunt work , as in 1999's the matrix , can be a real gem.» результат работы составил 59,6%,

то есть в большей мере положительный, о чём и говорится в тектсе.

Код функций и приложения в целом можно увидеть в приложении.

**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

## КОД ПРОГРАММЫ

```
import matplotlib.pyplot as plt
import numpy as np
from keras import layers, models
from keras.datasets import imdb
from keras.preprocessing.text import text_to_word_sequence

((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)
targets = np.array(targets).astype("float32")

test_x = data[:1000]
test_y = targets[:1000]
train_x = data[10000:]
train_y = targets[10000:]

model = models.Sequential()
model.add(layers.Dense(48, activation="relu", input_shape=(10000,)))
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
```

```
model.add(layers.Dense(48, activation="relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
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"])
history = model.fit(train_x, train_y, epochs=3, batch_size=512,
validation_data=(test_x, test_y),)

print("Test-Accuracy:", np.mean(history.history["val_accuracy"]))

plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.title("Training and validation accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Train", "Test"], loc="upper left")
plt.show()

plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Training and validation loos")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend(["Train", "Test"], loc="upper left")
plt.show()

review = (
    "the american action film has been slowly drowning to death in a
sea of asian wire-fu copycats",
    "it's not a pretty death , and it's leaving the likes of
schwartznager , stallone , and van damme wearing cement "
    "galoshes at the bottom of a kung fu sea",
    "sometimes , the mix results in a mind-blowing spectacle unlike
any other",
    "quality action with amazing and exciting stunt work , as in
1999's the matrix , can be a real gem",
    "but too often hollywood gets it wrong , even when they pay off
chinese directors",
    "flying ninjas and floating karate masters have been replaced by
soaring bronx detectives and slow motion kicking "
    "scientists",
    "mostly it's laughable",
)


def prepare_text(func):
    def wrapper(*args, **kwargs):
        with open(args[0]) as f:
            text = f.read().lower()
```

```
            text = text_to_word_sequence(text)
        indexes = imdb.get_word_index()
        text_indexes = []
        for item in text:
            if item in indexes and indexes[item] < 10000:
                text_indexes.append(indexes[item])
        return func(text_indexes, **kwargs)

    return wrapper


@prepare_text
def predict_for_text(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, 10000)
    targets = np.array(targets).astype("float32")

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

    model = models.Sequential()
    model.add(layers.Dense(48, activation="relu",
input_shape=(10000,)))
    model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
    model.add(layers.Dense(48, activation="relu"))
    model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
    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"])
    model.fit(
        train_x, train_y, epochs=3, batch_size=512,
validation_data=(test_x, test_y),
    )
    text = vectorize([text])
    result = model.predict(text)
    print(result)


predict_for_text("review.txt")
```