МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искуственные нейронные сети»

Тема: Прогноз успеха фильмов по обзорам

Студент гр. 8382 _____ Щемель Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель

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

Задачи

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

Требования

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)

Ход работы

1. Построение нейронной сети

Модель сети:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               1500100
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
```

Оптимизатор: `rmsprop`

Функция потерь: `binary_crossentropy`

Метрика: `accuracy`

Такая архитектура сети позволила добиться точности в 0.8980999886989594

2. Влияние размера входного вектора на результат

Результаты работы НС на разных размерах входного вектора

Размер	Результат
10000	0.8980999886989594
12000	0.897599995136261
15000	0.8958999812602997

Из полученных результатов можно сделать вывод, что увеличение размера вектора почти не влияет на результат.

3. Определение рейтинга по пользовательскому тексту

С помощью индекса слов, содержащиеся в введённом тексте приводятся к массиву встречающихся слов и передаются на вход НС.

Рейтинг автора	Результат работы НС
10	0.9429237
8	0.9994276
4	0.8413594
2	0.2819546
1	0.06345126

Отсутствие и первого результата значение 10 можно объяснить характером передаваемого текста (наличие слов, описывающих “плохие” вещи).

“Ошибочную” оценку третьего результата так же можно объяснить содержимым текста (который начинается с “This film is absolutely brilliant”).

Вывод

В ходе лабораторной работы была написана программа, прогнозирующая успех фильма по отзывам.

```
import sys
from pathlib import Path
from typing import Tuple

import numpy as np
from keras import layers
from keras import models
from keras.datasets import imdb

INDEX = imdb.get_word_index()


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

    return results


def get_data() -> Tuple[np.array, np.array]:
    (training_data, training_targets), (testing_data, testing_targets) = imdb.load_
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets), axis=0)

    return data, targets


def covert_text(text: str, max_size=10000) -> np.array:
    result = []

    for word in text.split():
```

```
            index = INDEX.get(word.lower())
            if index is None or index + 3 > max_size:
                continue
            result.append(index + 3)

        reverse_index = dict([(value, key) for (key, value) in INDEX.items()])
        decoded = " ".join([reverse_index.get(i - 3, "#") for i in result])
        print(decoded)

        return vectorize([result], max_size)


def main(file_path: Path):
    capacity = 10000
    input_data = covert_text(file_path.read_text())
    data, targets = get_data()

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

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

    model = models.Sequential()

    # Input - Layer
    model.add(layers.Dense(100, activation="relu", input_shape=(capacity,)))
    # Hidden - Layers
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(50, activation="relu"))
    model.add(layers.Dropout(0.2))
    model.add(layers.Dense(50, activation="relu"))
    # Output- Layer
    model.add(layers.Dense(1, activation="sigmoid"))
```

```
    model.summary()

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

    results = model.fit(
        train_x, train_y,
        epochs=2,
        validation_data=(test_x, test_y)
    )

    print(np.mean(results.history["val_accuracy"]))

    print(model.predict(input_data))


if __name__ == '__main__':
    main(Path(sys.argv[1]))
```