федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

О Т Ч Е Т
по исследовательскому проекту

«Кино-граф: актёры, фильмы и связи между ними»

Содержание

1. Общее описание проекта

1.1. Инициатор, заказчик проекта

Инициатором и заказчиком проекта является образовательная программа НИУ ИТМО в рамках учебной дисциплины.

1.2. Тип проекта

Проект является исследовательским, так как направлен на анализ реальных данных, построение модели и получение выводов на основе обработки и интерпретации результатов.

1.3. Состав команды

Проект выполнен студентами НИУ ИТМО:

- Сорокин Даниил Манхалевич
- Чесноков Егор Дмитриевич
- Малых Михаил Алексеевич
- Куранов Леонид Витальевич
- Цветков Даниил Вячеславович

2. Содержательная часть

2.1. Цели и задачи проекта

Цель проекта – построение и анализ графа связей между актёрами и фильмами для выявления ключевых участников и структурных особенностей киноиндустрии.

Задачи проекта:

- Сбор и предварительная обработка данных из датасета.
- Проектирование структур данных для представления графа.
- Построение графа с двумя типами узлов.
- Анализ центральности и связности.
- Визуализация результатов.
- Подготовка аналитического отчёта.

Источник данных: датасет *IMDB Top 1000*

2.2. Описание данных и их обработка

Исходный CSV-файл содержит сведения о фильмах из рейтинга IMDB Top 1000, включая название, год выпуска, жанры, рейтинг и четырёх главных актёров.

2.2.1. Производные файлы

Для решения поставленных задач была реализована совокупность Python-скриптов.

- EdgesCreate.py – формирует рёбра графа на основе исходного датасета. Для каждого фильма создаются рёбра «фильм–актёр» для всех звёзд съёмок, а также рёбра «актёр–актёр» для всех пар актёров, участвовавших в одном фильме. Результат сохраняется в файл edges.csv, где узлы идентифицируются префиксами f_(фильмы) и a_(актёры).
- ActorScore.py – на основе файла рёбер вычисляет для каждого актёра количество фильмов, в которых он снялся, и количество других актёров, с которыми он работал. Результаты записываются в файл stats.csv, который используется для выбора наиболее значимых вершин при визуализации.
- JSONtable.py – преобразует список рёбер в JSON-структуру данных, где каждому узлу сопоставлен список смежных узлов. Полученный файл gra.json используется для быстрого доступа к соседям узла в процессе визуализации.
- YearSpread.py – анализирует распределение фильмов по годам выпуска, подсчитывая количество фильмов в исходной выборке для каждого года. Результат сохраняется в файл spri.csv для последующего построения временного ряда.
- DrawGraph.py – выполняет визуализацию подграфа, включающего наиболее влиятельных актёров (топ-2 по количеству фильмов и топ-2 по количеству актёрских связей). Использует библиотеку networkx для построения графа и matplotlib для отрисовки. Узлы-фильмы подписываются красным цветом, узлы-актёры – тёмно-синим.

2.2.2. Производные файлы

В результате работы скриптов автоматически формируются следующие файлы:

- edges.scv– список рёбер графа;
- stats.csv– статистика по актёрам;
- graph.json– граф в формате списка смежности;
- spri.csv– распределение фильмов по годам.

2.2.3. Исходный код скриптов

Для обеспечения воспроизводимости и прозрачности процесса обработки данных ниже представлен исходный код всех скриптов проекта.

EdgesCreate.py

```
# Poster_Link, 0
# Series_Title, 1
# Released_Year, 2
# Certificate, 3
# Runtime, 4
# Genre, 5
# IMDB_Rating, 6
# Overview, 7
# Meta_score, 8
# Director, 9
# Star1, 10
# Star2, 11
# Star3, 12
# Star4, 13
# No_of_Votes,
# Gross

import csv

edges = set()


def addMovieAndActor(row):
    global edges
    for i in range(1, len(row)):
        edges.add(tuple([f"f_{row[0]}", f"a_{row[i]}"]))
    return


def addActorAndActor(row):
    global edges
    tmp = sorted(row[1:])
    for i in range(0, len(tmp)):
        for j in range(i + 1, len(tmp)):
            edges.add(tuple([f"a_{tmp[i]}", f"a_{tmp[j]}"]))


file = open("imdb_top_1000.csv", "r", newline="")
reader = csv.reader(file)
k = []
next(reader)
```

```
for row in reader:
    k.append([row[1], row[10], row[11], row[12], row[13]])
file.close()
for k_row in k:
    addMovieAndActor(k_row)
    addActorAndActor(k_row)

res = open("edges.csv", "w", newline="")
writer = csv.writer(res)
writer.writerow(["first", "second"])
writer.writerows(sorted(list(edges)))
res.close()
```

ActorScore.py

```
import csv

fop = {}


def addToList(key, value):
    global fop
    if key[0] == "a":
        if key not in fop.keys():
            fop[key] = {"films": 0, "actors": 0}
    else:
        return
    if value[0] == 'f':
        fop[key]["films"] += 1
    else:
        fop[key]["actors"] += 1


file = open("edges.csv", "r", newline="")
reader = csv.reader(file)
next(reader)
for row in reader:
    addToList(row[0], row[1])
    addToList(row[1], row[0])
file.close()

valid = []
for i in fop.items():
    valid.append([i[0][2:], i[1]["films"], i[1]["actors"]])

data = open("stats.csv", "w", newline="")
writer = csv.writer(data)
```

```
writer.writerow(["actor", "films_cnt", "actors_cnt"])
writer.writerows(sorted(valid))
data.close()
```

JSONtable.py

```
import json
import csv

data = {}


def addToJson(key, value):
    global data
    if key not in data:
        data[key] = [value]
    else:
        data[key].append(value)


file = open("edges.csv", "r", newline="")
reader = csv.reader(file)
next(reader)

for row in reader:
    addToJson(row[0], row[1])
    addToJson(row[1], row[0])
file.close()

file = open("graph.json", "w")
json.dump(data, file)
file.close()
```

YearSpread.py

```
import csv

file = open("imdb_top_1000.csv", "r", newline="")
reader = csv.reader(file)
k = {}
next(reader)
for row in reader:
    if row[2].isdigit():
        if row[2] in k.keys():
            k[row[2]] += 1
        else:
            k[row[2]] = 1
```

```
file.close()
res = open("spri.csv", "w", newline="")
writer = csv.writer(res)
data = []
for i in k.items():
    data.append([int(i[0]), i[1]])
writer.writerow(["date", "count"])
writer.writerows(sorted(data))
res.close()
```

DrawGraph.py

```
import csv
import json

import networkx as nx
import matplotlib.pyplot as plt

graph = nx.Graph()

file = open("stats.csv", "r", newline="")
reader = csv.reader(file)
next(reader)
data = []
actors = set()
for row in reader:
    data.append([row[0], int(row[1]), int(row[2])])
data = sorted(data, key=lambda x: -x[1])
for i in range(2):
    actors.add(data[i][0])
data = sorted(data, key=lambda x: -x[2])
for i in range(2):
    actors.add(data[i][0])
actors = list(actors)
file.close()

rows = open("graph.json", "r")
data = json.load(rows)
nodes = set()
edges = []
addis = []
for act in actors:
    nodes.add("a_"+act)
    for k in data["a_"+act]:
        nodes.add(k)
nodes = list(nodes)
```

```
goyda = open("edges.csv", "r", newline="")
redis = csv.reader(goyda)
next(redis)
for row in redis:
    if row[0] in nodes and row[1] in nodes:
        edges.append(row)

red_colors = []
blue_colors = []
i = 0
for node in nodes:
    if node[0] == 'f':
        graph.add_node(node[2:])
        red_colors.append(node[2:])
    else:
        graph.add_node(node[2:])
        blue_colors.append(node[2:])
    i += 1
for edge in edges:
    graph.add_edge(edge[0][2:], edge[1][2:])
    graph.add_edge(edge[1][2:], edge[0][2:])

plt.figure()
pos2 = nx.spring_layout(graph, seed=25, k=0.3)
nx.draw(graph,
        pos2,
        with_labels=False,
        node_size=0,
        node_color='none',
        edge_color='black',
        width=0.1,
        font_size=8
        )
for node, (x, y) in pos2.items():
    if node in red_colors:
        plt.text(x, y, node,
                 fontsize=8,
                 color="red",
                 ha='center',
                 va='center')
    else:
        plt.text(x, y, node,
                 fontsize=8,
                 color="darkblue",
                 ha='center',
                 va='center')

plt.show()
```

2.3. Модель графа

2.3.1. Узлы

- **Актёры** – вершины вида a_Имя_Фамилия.
- **Фильмы** – вершины вида f_Название_фильма.

2.3.2. Рёбра

- **актёр** – **фильм**: участие актёра в съёмках;
- **актёр** – **актёр**: совместное участие в одном фильме.

2.4. Используемые структуры данных и алгоритмы

2.4.1. Структуры данных

- list – хранение списков фильмов и актёров;
- set – хранение уникальных рёбер;
- dict – представление графа в виде списка смежности;
- networkx.Graph – анализ и визуализация графа.

2.4.2. Алгоритмы

- генерация рёбер для каждого фильма;
- подсчёт степени вершин актёров;
- анализ компонент связности;
- визуализация с использованием алгоритма spring_layout.

2.5. Результаты анализа

2.5.1. Топ-актёры по связности

Анализ выполнен на основе файла stats.csv, содержащего автоматически рассчитанные показатели количества фильмов и уникальных актёрских связей.

Актёр	Кол-во фильмов	Кол-во связей
Robert De Niro	17	45
Tom Hanks	14	38
Brad Pitt	12	36
Al Pacino	13	35
Clint Eastwood	12	33

Вывод: наиболее центральными вершинами графа являются Robert De Niro и Tom Hanks, обладающие максимальными значениями по количеству фильмов и числу уникальных

совместных актёрских связей. Это указывает на их высокую степень интеграции в крупнейший компонент связности графа.

2.5.2. Кластеризация и связность

На основе анализа графа и визуализации выявлено 8 компонент связности. Наиболее крупная компонента включает преимущественно голливудских актёров и фильмы. Остальные компоненты соответствуют региональным киноиндустриям (Индия, Япония и др.).

2.5.3. Распределение фильмов по годам выпуска

Анализ файла spri.csv показал, что пик количества фильмов в рейтинге IMDB Top 1000 приходится на 2014 год – 32 фильма, что является максимальным значением за весь период наблюдений.

2.6. Визуализация

Граф связей построен с использованием библиотек networkx и matplotlib. Узлы актёров выделены синим цветом, узлы фильмов – красным.

2.7. Интерактивная аналитическая панель в Yandex DataLens

В рамках проекта была создана интерактивная аналитическая панель с использованием облачного сервиса Yandex DataLens. Дашборд обеспечивает наглядное представление ключевых метрик проекта и позволяет проводить динамический анализ данных.

Основные компоненты дашборда:

- Динамика выпуска фильмов – график распределения фильмов по годам
- Рейтинг актёров – таблица топ-актёров по количеству связей и фильмов
- Сводная статистика – агрегированные показатели по всему датасету

Ссылка для доступа к дашборду:

https://datalens.yandex/ynfxys4cehl4i

3. Заключение

В результате выполнения проекта были достигнуты поставленные цели. В ходе работы были развиты навыки анализа данных, построения графовых моделей и интерпретации результатов. Проект способствовал формированию компетенций в области работы со структурами данных, алгоритмами и инструментами визуализации.

4. Результат проекта

Результатом проекта являются:

- аналитический отчёт;
- граф связей актёров и фильмов;
- набор автоматически сформированных CSV- и JSON-файлов;
- визуализации и материалы для защиты проекта.