Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |  |  |
| --- | --- | --- | --- |
| Институт: | ИВТИ | Кафедра: | УИТ |
| Направление подготовки: | | 27.04.04 Управление в технических системах | |

**ОТЧЕТ по практике**

|  |  |
| --- | --- |
| **Наименование практики:** | Учебная практика: ознакомительная практика |

**СТУДЕНТ**

|  |  |
| --- | --- |
|  | / Козлов И.А. / |
| *(подпись )* | (*Фамилия и инициалы*) |

|  |  |
| --- | --- |
| Группа | А-01м-23 |
|  | *(номер учебной группы)* |

**ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ**

|  |
| --- |
|  |
| *(отлично, хорошо, удовлетворительно, неудовлетворительно, зачтено, не зачтено)* |

|  |  |
| --- | --- |
|  | / / |
| *(подпись )* | (*Фамилия и инициалы члена комиссии*) |

|  |  |
| --- | --- |
|  | / / |
| *(подпись )* | (*Фамилия и инициалы члена комиссии*) |

**Москва**

**2023**

**Оглавление**

[**ПРОБЛЕМАТИКА РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ 3**](#_Toc153626278)

[**ОСНОВНЫЕ АЛГОРИТМЫ ПРИМЕНЯЕМЫЕ В РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМАХ 4**](#_Toc153626279)

[**КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ 6**](#_Toc153626280)

[**1) Сбор информации о предпочтениях 6**](#_Toc153626281)

[**2) Отыскание похожих пользователей 6**](#_Toc153626282)

[**2.1 Косинусное расстояние 6**](#_Toc153626283)

[**2.2 Коэффициент корреляции Пирсона 7**](#_Toc153626284)

[**3) Получения рекомендаций 8**](#_Toc153626285)

[**4) Оценка качества работы алгоритмов 8**](#_Toc153626286)

[**ПОСТРОЕНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ НА ОСНОВЕ КОЛЛАБОРАТИВНОЙ ФИЛЬТРАЦИИ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON 9**](#_Toc153626287)

[**Список литературы 14**](#_Toc153626288)

ПРОБЛЕМАТИКА РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

Рекомендательные системы – это технологии, которые предлагают пользователям персонализированные рекомендации, основанные на их предпочтениях и истории взаимодействия с платформой или сервисом. Эти системы широко применяются в различных областях, таких как электронная коммерция, стриминг мультимедиа, социальные сети, новостные порталы и другие.

Проблематика рекомендательных систем включает в себя ряд вызовов и проблем:

* Фильтрование информации: Рекомендательные системы помогают пользователям справляться с информационной перегрузкой, но они также могут уменьшить разнообразие информации, предоставляя рекомендации, которые соответствуют их существующим интересам. Это может привести к "пузырьковой изоляции", когда пользователи ограничиваются контентом, который соответствует их мнениям и интересам, и не видят разнообразных точек зрения.
* Приватность и безопасность данных: Рекомендательные системы собирают и анализируют большое количество личных данных пользователей. Это может вызвать опасения в отношении приватности и безопасности данных, особенно если эти данные попадают в руки злоумышленников или используются в недобросовестных целях.
* Байесовский выбор: Рекомендательные системы могут уделять слишком много внимания более популярным или монотонным элементам, так как они часто получают больше взаимодействий от пользователей. Это может привести к неразнообразию рекомендаций и игнорированию менее известных, но более интересных вещей.
* Холодный старт: Рекомендательные системы могут столкнуться с проблемой "холодного старта", когда новые пользователи или элементы не имеют истории взаимодействия. В этом случае системе трудно предоставить релевантные рекомендации.
* Фильтрация нежелательного контента: Рекомендательные системы должны также учитывать этические и законодательные нормы, чтобы избегать рекомендаций нежелательного или вредного контента, такого как фейковые новости, порнография, насилие и др.

ОСНОВНЫЕ АЛГОРИТМЫ ПРИМЕНЯЕМЫЕ В РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМАХ

**Коллаборативная фильтрация:**

На сегодняшний день, этот метод можно считать самым распространенным.

Коллаборативная фильтрация делится на два основных типа:

*User-Based Collaborative Filtering*: Рекомендации основаны на схожести пользователей. Если два пользователя оценили похожие элементы высоко, то система может рекомендовать элементы, оцененные одним пользователем, другому.

*Item-Based Collaborative Filtering*: Рекомендации основаны на схожести элементов. Если пользователь положительно оценил определенный элемент, система может рекомендовать элементы, похожие на тот, который был положительно оценен.

**Фильтрация на основе контента:**

Этот метод рекомендует элементы на основе их сходства с предпочтениями пользователя. Например, для рекомендации фильмов, этот метод учитывает жанры, актеров, режиссеров и другие характеристики фильмов, которые пользователь предпочел ранее.

**Матричные разложения:**

Этот метод разбивает матрицу взаимодействий между пользователями и элементами на несколько матриц более низкого ранга, что позволяет заполнить пропущенные значения и предсказать, какие элементы пользователь может предпочесть.

**Гибридные методы:**

Это комбинация различных методов, таких как фильтрация на основе контента и коллаборативная фильтрация, чтобы улучшить качество рекомендаций.

**Глубокое обучение:**

С использованием нейронных сетей и глубокого обучения можно разрабатывать более сложные модели рекомендательных систем, которые могут учесть более сложные зависимости и контекст.

КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ

1. **Сбор информации о предпочтениях**

Первое, что нам нужно, сбор предпочтений пользователей, в датасете должны быть представлены оценки пользователей на некоторые продукты, такие как: фильмы, книги, музыка, товары в магазине и т.д. Оценки могут быть выражены в виде разных характеристик, пример Таблица 1.

Таблица **Ошибка! Текст указанного стиля в документе отсутствует.**. Примеры оценки продукта.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Онлайн покупка |  | Статьи в интернете |  | Просмотр фильма онлайн |  |
| Купил | 3 | Время, потраченное на статью | T | Выставленный рейтинг | Оценка от 1 до 5 |
| Добавил в избранное | 2 | Открыл | 1 | Смотрел | 1 |
| Смотрел | 1 | Не открыл | 0 | Не смотрел | 0 |
| Не купил | 0 |  |  |  |  |

На этом этапе определяется какая информацию о действиях пользователя будет собрана, это важно для дальнейшей работы.

1. **Отыскание похожих пользователей**

Здесь, исходя из датасета, определяется по каким, из уже собранных критериев, мы будем определять похожесть пользователей (см. Таблицу 1.). Так же определяется метрика похожести, например косинусное расстояние или корреляция Пирсона.

* 1. **Косинусное расстояние**

Косинусное расстояние между двумя векторами измеряет угол между ними и может служить метрикой сходства между этими векторами. Чем ближе значение косинусного расстояния к 1, тем более похожи векторы, а чем ближе к -1, тем более они противоположны.

, где A и B – вектора признаков

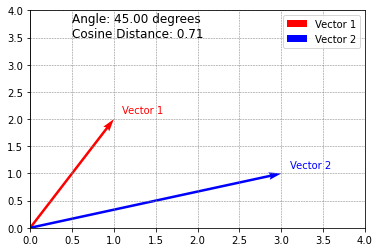


Рисунок . Визуализация косинусного расстояния.

* 1. **Коэффициент корреляции Пирсона**

Коэффициент корреляции Пирсона – это мера статистической взаимосвязи между двумя переменными, измеряемая от -1 до 1. Он используется для оценки степени линейной зависимости между двумя случайными величинами. Когда коэффициент равен 1, это означает идеальную положительную корреляцию, когда одна переменная растет, другая тоже растет. Когда коэффициент равен -1, это означает идеальную корреляцию, когда одна переменная растет, другая убывает. Если коэффициент равен 0, это указывает на отсутствие корреляции между переменными.

1. **Получения рекомендаций**

Выбор коэффициента похожести не так сильно влияет на результаты, как правило, выбор происходит эмпирическим путем. После чего происходит расчет схожести пользователей, то есть считается расстояние каждый с каждым.

Чтобы получить рекомендации для конкретного пользователя ,

мы смотрим на одного или на несколько наиболее близких к нему пользователей, и берем все продукты, с которыми эти пользователи взаимодействовали, отсекаем от них те, которые уже использовал . Оставшиеся продукты и будут рекомендациями для текущего пользователя. Подробнее с примерами рассмотрим в следующей главе.

1. **Оценка качества работы алгоритмов**

Для оценки качества работы используются метрики, в основном это метрики точности предполагаемого значения или реального, если таковой имеется.

MAE (Mean Absolute Error, пер. средняя абсолютная ошибка) – оценивается как разница между реальным и предсказанным значением.

RMSE (Root Mean Squared Error, пер. средняя квадратичная ошибка) – ошибка вычисляется как корень из суммы квадратов разниц между предсказываемым значением и реальным значением.

Accuracy – метрика, которая основывается при наличии реальных значений. T – множество всех релевантных рекомендаций, N – длина всего списка рекомендаций.

ПОСТРОЕНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ НА ОСНОВЕ КОЛЛАБОРАТИВНОЙ ФИЛЬТРАЦИИ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

В этом разделе мы построим систему рекомендаций литературы на основе датасета с Kaggle. Датасет содержит информацию о пользователях, поставивших оценку книгам, которые они купили в интернет-магазине. Есть названия, описания этих книг, жанры и авторы, ссылки на книги.

Исходный датасет состоит из 5 таблиц, таблицы включают в себя следующую информацию.

* book\_id – ID книги
* title – название книги
* image\_url – ссылка на обложку
* url – ссылка на сайт с описанием книги
* num\_pages – число страниц в книге
* ratings\_count – число оценок поставленных книге
* description – описание
* genre – список жанров
* name – автор
* book\_id\_mapping – сопоставление идентификаторов книг в разных

таблицах

* user\_id\_mapping - сопоставление идентификаторов пользователей в

разных таблицах

* user\_id – ID пользователя
* Predicted Rating – рейтинг присужденный пользователем книге
* Actual Rating – общий рейтинг книги

Исходный датасет содержит почти 200 тысяч записей, примерно о 60 тысяч пользователях. Эмпирическим путём было получено, что такой объем вычислений не произвести без специального оборудования (отдельного сервера). Поэтому было решено взять только пользователей, купивших 6 и более книг. Таким образом остается 3897 пользователей. Так же такой подход решает проблему холодного старта, так как мы имеем достаточно информации о пользователях, в реальной жизни ситуации может быть другой.

Для решения данной задачи удобно использовать встроенные библиотеку python: Numpy и Pandas.

# Данные представлены в формате CSV, считываем с помощью встроенных методов pandas

call\_book = pd.read\_csv('datasets/books/collaborative\_books\_df.csv')

# Получаем словарь, где по id юзера указаны id прочитанных им книг

read\_by\_user = {}

user\_id\_mapping = set(call\_book['user\_id\_mapping'])

for i in user\_id\_mapping:

    temp\_df = call\_book.loc[call\_book['user\_id\_mapping'] == i]

    read\_by\_user.update({i: sorted(list(temp\_df['book\_id']))})

# Сокращаем число пользователей, оставляем тех, кто прочитал 6 и более книг

list\_keys = []

for key in read\_by\_user.keys():

    if len(read\_by\_user[key]) > 6:

        list\_keys.append(key)

len(list\_keys)

Далее можно считать косинусное расстояние, для этого используется библиотека Scipy. В качестве координат будут оценки пользователей на купленные книги, оценки ставятся от 1 до 5.

В файл 'output\_key\_index\_3000.txt' записаны индексы пользователей, между которыми расстояние получается не нулевое, это сделано для уменьшения вычислительных затрат.

import scipy.spatial.distance as ds

with open('output\_key\_index\_3000.txt') as file:

    with open('cosine\_dim.txt', 'w') as write\_file:

        for line in tqdm(file):

            i, j = list(map(int, line.split()))

            cosine = (ds.cosine(grade\_list[i], grade\_list[j]))

            write\_file.write(str(i) + ' ' + str(j) + ' ' + str(cosine) + '\n')

cosine\_list = []

with open('cosine\_dim.txt') as file:

    for line in tqdm(file):

        temp = line.split()

        temp = int(temp[0]), int(temp[1]), float(temp[2])

        cosine\_list.append(temp)

cosine\_dim = pd.DataFrame(columns=['x', 'y', 'cosine'], data=(cosine\_list))

Таким образом, получаем таблицу с индексами пользователей и косинусным расстоянием между ними, таблица имеет 796274 уникальных записей:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **x** | **y** | **cosine** |
| **0** | 0 | 1 | 0,866865 |
| **1** | 0 | 32815 | 0,743842 |
| **2** | 0 | 49220 | 0,883996 |
| **3** | 0 | 8288 | 0,866469 |
| **4** | 0 | 49250 | 0,703843 |
| **5** | 0 | 24695 | 0,730879 |
| **6** | 0 | 16512 | 0,955926 |
| **7** | 0 | 8344 | 0,932982 |
| **8** | 0 | 24785 | 0,83639 |
| **9** | 0 | 32988 | 0,869538 |
| **10** | 0 | 238 | 0,777684 |

Напишем функцию для выделения наиболее похожих пользователей.

def getting\_closes(i: int):

    """Функция выдает наиболее похожих пользователей для заданного пользователя i"""

    if not i in all\_users:

        return (-1, f"Пользователя {i} не существует")

    reco\_df = cosine\_dim.loc[((cosine\_dim.x == all\_users[i]) | (cosine\_dim.y == all\_users[i]))].sort\_values('cosine', ascending=False)

    quant = reco\_df.quantile(0.90).cosine

    reco\_df.loc[reco\_df.cosine >= quant]

    result = reco\_df[reco\_df['cosine'] >= quant]

    return result

def getting\_recommendations(i: int):

    """Выдает рекомендованные id книг для пользователя i"""

    if not i in all\_users:

        return (-1, f"Пользователя {i} не существует")

    read = read\_by\_user[i]

    closes\_val = getting\_closes(i)['y']

    reco\_book = []

    for j in range(len(closes\_val)):

            reco\_book.extend(read\_by\_user[closes\_val.iloc[j]])

return set(reco\_book)

i = int(input("Введите id пользователя: "))

if i in all\_users:

reco\_set = getting\_recommendations(i) - set(read\_by\_user[i])

filtered\_df = call\_meta[call\_meta['book\_id'].isin(reco\_set)]['title']

print(f"{len(reco\_set)} рекомендаций для пользователя {i}")

filtered\_df.to\_excel(f'reco\_result/reco-{i}.xlsx')

else:

print(f"Пользователя {i} не существует")

В переменную all\_users записаны все id пользователей, выбираем любого, например 196, получаем файл с таблицей, в которой указаны id и названия книг, которые система рекомендует текущему пользователю.

|  |  |
| --- | --- |
|  | **title** |
| **1** | Blue Bloods Blue Bloods 1 |
| **3** | Sharp Objects |
| **11** | The Ultimate Hitchhikers Guide to the Galaxy |
| **15** | Living Dead in Dallas Sookie Stackhouse 2 |
| **28** | The TellTale Heart and Other Writings |
| **31** | Freakonomics A Rogue Economist Explores the Hidden Side of Everything Freakonomics 1 |
| **35** | The Firm Penguin Readers Level 5 |
| **37** | Where the Sidewalk Ends |
| **55** | Free Four Tobias Tells the Divergent KnifeThrowing Scene Divergent 15 |
| **74** | The Amber Spyglass His Dark Materials 3 |
| **85** | Marley and Me Life and Love With the Worlds Worst Dog |
| **87** | A Child Called It Dave Pelzer 1 |
| **88** | Oedipus Rex The Theban Plays 1 |
| **90** | Maus I A Survivors Tale My Father Bleeds History Maus 1 |

Вывод: таким образом, мы получили 14 рекомендаций для пользователя, если пройтись по всему списку пользователей, то можно получить рекомендации для каждого из них.

Список литературы

1. Tobi Segaran. Programming Collective Intelligence, 2007.
2. Р.И. Ролгин. Метрики оценки качества работы систем коллаборативной фильтрации.