Міністерство освіти й науки
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

«Дослідження методології розробки рекомендаційних систем»

Текстова частина до курсової роботи
за спеціальністю «Інженерія Програмного Забезпечення» 121

Керівник курсової роботи:
старший викладач
Салата К. В.
_____(підпис)
«___» _____ 2022 р.
Виконала студентка ІПЗ-4
Шутяк Т.В.
«___» _____ 2022 р.

Київ 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

Доцент., к. ф.-м. н. С. С. Гороховський

(підпис)

„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці Шутяк Таїсії Володимирівні факультету інформатики 4-го курсу

ТЕМА Дослідження методології розробки рекомендаційних систем

Зміст ТЧ до курсової роботи:

1. Індивідуальне завдання
2. Календарний план
3. Анотація
4. Вступ
5. Загальна інформація про рекомендаційні системи
6. Колаборативна фільтрація
7. Типові проблеми при побудові рекомендаційних систем
8. Практична частина
10. Висновки
11. Перелік використаних джерел
12. Додатки

Дата видачі „___” _____ 2022 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи

№з/п	Назва етапу курсової роботи	Термін виконання	Примітка
1	Отримання завдання на курсову роботу	15.10.2021	
2.	Пошук та огляд літератури за темою роботи	5.11.2021	
3.	Дослідження алгоритмів обрахунку рекомендацій	20.12.2021	
4.	Написання демонстраційного застосунку для тестування роботи рекомендаційних алгоритмів	21.02.2022	
6.	Написання текстової частини курсової роботи	4.04.2022	
7.	Створення презентації	25.04.2022	
8.	Надання роботи на перевірку керівнику	11.05.2022	
9.	Коригування роботи відповідно до зауважень керівника	20.05.2022	
10.	Захист курсової роботи	27.05.2022	

Студент _____Шутяк Т.В._____

Керівник _____Салата К. В._____

“_____” _____

Зміст

Анотація

У даній роботі описуються підходи та алгоритми, які використовуються при побудові рекомендаційних систем. Розглядаються типові проблеми при формуванні рекомендацій та можливі їх рішення.

Курсова робота присвячена розгляду та аналізу основних типів рекомендаційних систем. В ході роботи проаналізовано способи збору інформації про користувача, підбір рекомендацій на основі колаборативної фільтрації. Досліджено різні методи обрахунку схожості. Розглянуто алгоритми, на основі яких працюють рекомендаційні системи.

Отримані знання застосовано для виконання практичної частини роботи.

Вступ

У часи розвитку інформаційних систем рекомендаційні системи є майже всюди. Людині постійно треба робити вибір: який фільм переглянути, яку пісню послухати, що придбати в магазині, яку гру завантажити. З появою Інтернету людство отримало не лише доступ до інформації з усього світу, а й необхідність обирати щось серед тисяч або навіть мільйонів варіантів.

Для прискорення процесу вибору було створено рекомендаційні системи – програми, метою яких є створення рекомендацій на основі інформації про вподобання користувача. Тепер алгоритми подумають, що людині було б цікаво замість неї, адже набагато простіше та приємніше, коли те, що вам потрібно, знаходять за вас. Збір інформації, її аналіз, пошук потрібних продуктів чи сервісів на основі наданої інформації – це та робота, яку виконують рекомендаційні системи.

Перед власне створенням рекомендацій система має отримати хоча б якусь інформацію про користувача та його вподобання, але якби цей процес був явним всюди – людям би набридло постійно вводити інформацію, тому системи навчилися збирати цю інформацію так, що користувач може цього навіть не помічати. Ви переглянули якийсь товар в магазині, поставили вподобайку на пісню, оцінили фільм, а потім рекомендаційні системи проаналізують всю цю інформацію та нададуть вам релевантні рекомендації. Точність рекомендацій залежить від алгоритмів, які використовує система.

Netflix, YouTube, Spotify – рекомендаційні системи, з якими стикався, напевно, майже кожен. Навіть реклама в Інтернеті часто підбирається на основі того, що користувач переглядав раніше. З часом рекомендаційні системи стали більш точними, алгоритми вдосконалилися, методи змінилися, впровадилися нові підходи.

Метою даної роботи є розгляд найпоширеніших типів рекомендаційних систем та аналіз алгоритмів, які в них використовуються. Також розглянуто методи збору інформації, на основі якої користувачу надаватимуться

рекомендації, способи підбору рекомендацій, за умов відсутності інформації про попередні вподобання користувача.

Також у ході курсової роботи для демонстрації роботи алгоритмів було реалізовано деякі з них, після чого було проведено порівняння отриманих рекомендацій із застосуванням кожного з реалізованих алгоритмів.

Розділ 1. Загальна інформація про рекомендаційні системи

Метою рекомендаційних систем є вирішення проблеми вибору, скорочення часу на пошук чогось, також дані системи допомагають постачальникам благ, оскільки вони можуть гарантувати, що потенційно зацікавлені користувачі матимуть змогу отримати інформацію про товари, які їх можуть зацікавити.

Від того, наскільки релевантні товари пропонуватиме система користувачу, залежить успішність платформи, яка дану систему використовує. Тому вкрай важливим є вибір правильного алгоритму підбору рекомендацій.

Рекомендації поділяють на персоналізовані та не персоналізовані.

Як приклад не персоналізованих рекомендацій можна запропонувати рекомендацію найпопулярніших товарів. Для прикладу, користуючись інтернет-магазином «Rozetka», після пошуку товару у вас є можливість встановити сортування товарів за популярністю. Система не може гарантувати, що найпопулярніший товар – це те, що вам потрібно, але оскільки люди беруть його частіше, ймовірність того, що цей товар зацікавить користувача, набагато вища, ніж ймовірність того, що його зацікавить товар, який, до прикладу, придбали 0 разів.

Персоналізовані рекомендації, в свою чергу, спираються на знання про взаємодію користувачів з системами, та, користуючись алгоритмами, деякі з яких будуть розглянуті у наступних розділах, формують свої рекомендації.

Часто для вибору алгоритму слід враховувати специфіку вимог до тієї чи іншої системи. Тобто які саме продукти має рекомендувати система, на яку кількість користувачів вона розрахована, на яку кількість товарів вона розрахована. Слід зауважити, що алгоритми, які показують гарні результати в одних системах, можуть бути зовсім непридатними для інших, зважаючи на дані, які система має аналізувати.

Проте мало лише обрати алгоритм, навіть система з найідеальнішим алгоритмом може не працювати у вашому конкретному випадку, якщо присутні проблеми за двома аспектами[1]:

- Повнота даних

Якщо компанія умовно взаємодіє з 3 користувачами, вподобання яких є абсолютно різними – колаборативна фільтрація у даному випадку не допоможе, оскільки схожих користувачів у вашій системі немає. У даному випадку часто легше використати працівника, який за допомогою логіки здатен визначити, що краще порекомендувати щось тому чи іншому покупцю.

- Глибина даних

Скільки даних про вподобання користувача є у вашій системі. Варіантами даних є взаємодія з власне з вашою системою, інші вподобання в Інтернеті, інформація про офлайн вподобання, деякі системи при формуванні рекомендацій спираються також на геолокацію користувача, час дня, вік, операційну систему, з якої було здійснено вхід. Чим більше інформації ви матимете, тим якісніші рекомендації ви зможете надати.

Серед основних сфер застосувань рекомендаційних систем можна виділити наступні:

- Електронна комерція – власне галузь, яка однією з перших почала використовувати рекомендаційні системи. Основною задачею рекомендаційних систем тут є підбір товарів, які найімовірніше захоче придбати користувач.
- Соціальні мережі – рекомендація людей, каналів, публікацій та іншого контенту, який наявний в конкретній соціальній мережі.
- Розваги – рекомендація ігор, фільмів, музики, книг тощо.
- Контент – рекомендація новин, статей, рекомендація вебсторінок при пошукових запитах тощо.

Загалом, алгоритми, які застосовуються у рекомендаційних системах, можна поділити на 3 типи:

- Колаборативна фільтрація (collaborative filtering) – ідея даного сімейства рекомендаційних алгоритмів полягає в тому, що схожим користувачам можуть подобатися схожі продукти.
- Фільтрація на основі вмісту (content-based) – у даному сімействі алгоритмів рекомендації надаються шляхом аналізу контенту (товарів) на відповідність вподобанням користувача.
- Гібридна фільтрація – передбачає сумісне використання методів колаборативної фільтрації та фільтрації на основі вмісту.

1.1 Огляд існуючих рекомендаційних систем

Навіть найпримітивніший інтернет-магазин може використовувати рекомендаційні алгоритми, але чим більше специфічних даних, критеріїв для рекомендацій, а також бажання дати найточніші рекомендації, тим складнішими стають рекомендаційні системи. Великі компанії розширюють вже існуючі або створюють власні алгоритми рекомендаційних систем. Як приклад можна надати такі команії як Netflix, Amazon, Google, LinkedIn, Facebook, Instagram, Twitter та багато інших.

- Рекомендаційна система GroupLens

GroupLens, створена дослідницькою лабораторією GroupLens, була одним з першопрохідців серед рекомендаційних систем. Система спеціалізувалася на рекомендації новин. GroupLens збирала оцінки від читачів Usenet і використовувала їх, щоб передбачити, чи сподобається іншим читачам стаття, перш ніж вони її прочитають. Одні з перших алгоритмів колаборативної фільтрації були розроблені саме у рамках системи GroupLens. Загальні ідеї розроблені дослідницькою групою згодом були адаптовані до інших типів контенту, таких як: книги (BookLens), фільми (MovieLens)[2]. Окрім досліджень у галузі рекомендаційних систем, дослідницька команда також видала кілька дата-сетів, зокрема 3 дата-сети з системи MovieLens.

- Рекомендаційна система Amazon.com

Рекомендаційна система Amazon.com також була одним з першопрохідців у галузі рекомендацій, які спеціалізувалися на комерції. Усвідомивши користь використання рекомендацій, бізнес з продажу електронних книг з часом розширився настільки, що зараз на Amazon.com можна знайти майже всі види існуючої продукції. Рекомендації тут надаються на основі явно наданих користувачем оцінок товарів, а також неявному зборі даних (купівельна поведінка користувача та його перегляди). Оцінка користувача ставиться за шкалою від 1 до 5. Інформація щодо переглядів та оцінок користувача збирається, коли користувач входить до системи за допомогою механізму аутентифікації.

- Рекомендаційна система Facebook

У соціальних мережах людям часто рекомендують потенційних друзів або контент, що може зацікавити. Цей вид рекомендацій має дещо інші цілі, ніж рекомендація продуктів. Якщо рекомендація та продаж продуктів дозволяє компанії збільшити розмір прибутку, то рекомендація друзів та контенту покращує користувацький досвід у даній мережі, що, по-перше, змусить користувача проводити більше часу у даній мережі, по-друге, привабить більше користувачів у мережу. Чим більше людей проглядає мережу, тим більше доходів буде отримано від реклами.

Такі типи рекомендацій базуються не на оцінках, а на зв’язках між користувачами, користувачем та групою тощо. Тому у даних системах використовуються інші алгоритми, які базуються на обробці зв’язків.

1.1.1 Рекомендаційна система Spotify

Для кращого розуміння власне роботи алгоритмів, а також їх комбінацій розглянемо один з найпопулярніших зараз додатків – стрімінговий сервіс Spotify.

Spotify було засновано 2006 року, спочатку планувалося використовувати додаток як музичну бібліотеку, проте з розвитком

платформи розробники зрозуміли, що персоналізація позитивно вплине на додаток. Тому у систему почали впроваджуватися алгоритми, які аналізували вподобання користувача та видавали йому рекомендації на основі прослуханих треків. З часом алгоритми удосконалювалися та система все краще і краще надавала рекомендації.

Для пошуку рекомендацій система використовує внутрішні та зовнішні дані:

- До внутрішніх даних відносять дані музичного треку (виконавець, зображення, назва, опис, жанр, текст, файл треку) та дані про користувача (історія прослуханих пісень, тривалість прослуховування, кількість повторних прослуховувань, пропущені треки, збережені пісні, створені списки відтворення, взаємодія з іншими користувачами).
- Зовнішні дані – аналіз статей, повідомлення в блогах та інші дані, пов’язані з треками, жанрами та виконавцями, які можна знайти в мережі «Інтернет».

Загалом дані, які використовує у своїх алгоритмах Spotify можна поділити на 3 групи:

- Аналіз поведінки поточного користувача – збережені списки відтворення, пропущені треки, улюблені жанри, треки та виконавці, спільні плейлисти, з якою частотою прослуховується певний список відтворення або пісня, залежність того, що користувач прослуховує у певний час доби.
- Аналіз інших (користувачів та виконавців) – списки відтворення, вподобані іншими користувачами, що зараз в тренді, що подобається слухати користувачам, подібним до поточного, нові треки виконавців, які подобаються поточному користувачу.
- Аналіз треку – аналіз спектрограми треку, тривалість, гучність, ритм, темп та інші характеристики.

Для рекомендацій Spotify використовується система BaRT («Bandits for Recommendations as Treatments»). Дана система має 2 режими Exploitation (експлуатація) та Exploration (дослідження)[3].

Експлуатація – це режим, у якому система використовує інформацію, яку вона зібрала про користувача, це є стандартний режим, у якому, зазвичай, працюють системи, засновані на колаборативній фільтрації. Даний режим є ідеальним, за умови достатньої кількості наявної інформації про користувача у системі. Але однією з проблем, яка виникає для подібних систем, є відсутність даних про користувача чи елемент. Наприклад, у системі є новий користувач або трек, який ще ніхто не прослуховував або прослуховували нечасто. У даному випадку у системи можуть виникати проблеми з наданням релевантних рекомендацій. Саме для уникнення подібного типу проблем використовується режим Exploration.

Exploration – це режим, у якому система рекомендує контент, щодо якого у неї немає достатньо даних, які дозволили б точно сказати сподобається даний трек користувачеві чи ні. Контент рекомендується з метою збору додаткової інформації. Перехід у цей режим дає гарантію, що навіть якщо трек щойно додали, він все одно має шанс бути рекомендованим. Рекомендація вважається успішною, якщо користувач слухає рекомендований трек довше ніж 30 секунд.

Задачею системи BaRT є максимальний підбір рекомендацій, які є релевантними. Система навчається на власних помилках та вдосконалюється з кожним отриманим відгуком. Проте на цьому команда Spotify не зупинилася та додала до алгоритму контексти. Тепер задачею алгоритму є не просто видавати рекомендації, а також враховувати додаткові фактори, за якими трек може бути рекомендовано (наприклад, залежно від дня тижня, від нових релізів, від попередніх прослуховувань користувача, від популярності треку тощо).

Проте, якщо у систему додався новий виконавець, тут з’являється так звана проблема холодного запуску. У даному випадку використовується аналіз аудіофайлів з використанням методів машинного навчання. Доріжки конвертуються у спектрограми (частотно-часові графіки звуку), які подаються на вхід нейронної мережі, яка починає їх аналіз та намагається зрозуміти, як мелодія буде сприйнята слухом користувача.

Окрім аналізу спектрограм, Spotify також використовує NLP(обробку мови), з допомогою якої аналізується вміст десятків мільйонів вебсайтів, серед яких форуми, блоги та статті. Обробляється дана інформація з метою пошуку ставлення користувачів до певних пісень, аналізу того, що зараз в тренді, та іншої інформації, яка може допомогти у рекомендації того чи іншого треку.

Кожен метод рекомендацій має свої недоліки, проте сумістивши їх, додавши можливість дослідження, забезпечивши явний та неявний збір даних, які так чи інакше зможуть допомогти при генерації рекомендацій, дана система забезпечила високу якість наданих рекомендацій, що підтверджується кількістю користувачів сервісу Spotify.

Розділ 2. Колаборативна фільтрація

У цьому розділі описані основні методи та алгоритми, що використовуються при колаборативній фільтрації. Розглянуто memory-based та model-based алгоритми.

Основна ідея даного підходу до підбору рекомендацій базується на схожості між користувачами та/або елементами. Термін «колаборативний» передбачає необхідність деякої спільної діяльності різних користувачів у одній системі, інформацію про дану діяльність використовують для фільтрації великої кількості інформації і генерації рекомендацій. Наприклад, маємо двох користувачів A і B, які однаково оцінювали деякі елементи у минулому, на основі даної інформації користувачів можна вважати схожими, тому їх історія вподобань вплине на рекомендації, які кожен з них отримає, більше ніж попередні вподобання інших користувачів, які не мають з ними нічого спільного.

Для відображення того, наскільки елемент подобається або не подобається користувачу, можуть використовуватися різні системи оцінки елементів користувачем. Серед прикладів можна навести числовий рейтинг, бінарне та унарне оцінювання. Числовий рейтинг – оцінити щось за шкалою від мінімальної оцінки до максимальної (наприклад, рейтинг фільму чи книги). Бінарний рейтинг – можна обрати лише між двома варіантами – подобається та не подобається (наприклад, лайк чи дизлайк відео на Youtube). Унарний рейтинг – все що, може зробити користувач – позначити те, що йому подобається (наприклад, лайк поста в Інстаграм). Подібне оцінювання відноситься до явних методів збору інформації про вподобання користувача.

Перед початком розгляду алгоритмів слід ввести поняття user-item interaction matrix (матриця взаємодії користувачів та товарів):

позначимо U – як множину користувачів системи ($U = \{u_1, u_2, \ldots, u_n\}$), а також E – множина елементів ($E = \{e_1, e_2, \ldots, e_m\}$). R – множина оцінок, r_{ue} –

оцінка надана користувачем «u» елементу «e». Якщо позначити користувачів як рядки матриці, елементи – стовпчики, а перетин рядка та стовпчика – оцінка відповідним користувачем відповідного товару, то отримаємо матрицю взаємодії.

Слід зауважити, що, якщо у системі багато користувачів та багато товарів, то ця матриця є доволі розрідженою, оскільки кожен користувач не оцінюватиме кожен товар.

Дана матриця є основним елементом, на якому базуються алгоритми колаборативної фільтрації.

Алгоритми колаборативної фільтрації можна розділити на:

- memory-based – використовуються нескладні алгоритми, які потребують великих затрат пам’яті.
- model-based – використовуються алгоритми машинного навчання, які дозволяють значно скоротити об’єми використовуваної пам’яті.

2.1 Memory-based алгоритми

Memory-based алгоритми намагаються передбачити оцінку користувачем товару за допомогою обрахування зваженої середньої оцінки користувачами, які є схожими з даним користувачем, даного товару. Memory-based алгоритм завантажує всю матрицю взаємодії у пам’ять системи та робить прогноз базуючись на дані, що містяться у пам’яті.

Memory-based алгоритми в свою чергу поділяються на user-based та item-based фільтрацію. Дані підходи є доволі схожими, але видають різні результати. Результат отриманий з використанням user-based фільтрації можна описати як: «Користувачі, які схожі на тебе, також вподобали …» В свою чергу item-based фільтрація скоріше запропонує товари, для яких справедливим є висловлювання: «Користувачі, які вподобали даний товар, також вподобали …» Обидва методи у своїх алгоритмах спираються на матрицю взаємодії.

2.1.1 User-based filtration

User-based фільтрація складається з таких кроків:

1. Нормалізація даних.
2. Обчислення схожості.
3. Вибір сусідів.
4. Оцінювання товарів.
5. Вибір товарів.

Тепер зупинимося детальніше на кожному з цих етапів.

Поширеною проблемою є різні підходи визначення оцінки товару різними користувачами, таким чином користувачі зі схожими інтересами досі можуть оцінювати різні товари по-різному. Якщо ми маємо справу з системою на основі рейтингів, слід враховувати людський фактор, тобто те, що деякі люди можуть, в принципі, на все ставити доволі високі оцінки, через власну доброту, в той час як інші можуть оцінювати все доволі низько. Тобто припустимо 8/10 від однієї особи, може означати те ж саме, що 4/10 від іншої. Щоб людський фактор не заважав роботі алгоритму, першим етапом алгоритму є нормалізація оцінок.

Найпоширенішими методами нормалізації оцінок є метод нормалізації Ґаусса, MinMax та decoupling normalization (нормалізація шляхом відділення).

Нормалізація Ґаусса: метод вираховує 2 фактори, які можуть спричинити відхилення рейтингу серед користувачів зі схожими інтересами:

1. **Зміщення середніх оцінок.** Проблема, яка стосується того, що деякі користувачі можуть більш лояльно оцінювати товар, та давати вищу оцінку, ніж більш прискіпливі користувачі. Як результат середня оцінка, більш лояльного користувача буде вища, ніж середня оцінка більш прискіпливого. Щоб прибрати вплив вказаного фактору на обрахунок рекомендацій виконується віднімання від оцінок кожного користувача від середньої оцінки відповідного користувача.

2. **Різне розходження шкали оцінок.** Ця проблема стосується того, що більш консервативні користувачі мають менший діапазон оцінювання, в той час як більш ліберальні користувачі більший. Щоб врахувати цей фактор, оцінка кожного користувача ділиться на розбіжність (дисперсію) його оцінок [4]. (Дисперсія — міра розсіяння значень випадкової величини відносно середнього значення розподілу [5].)

Скомбінувавши ці ідеї, отримуємо формулу для вирахування нормалізованої оцінки для товару x користувачем y $R_y(x)_{avg}$

$$R_y(x)_{avg} = \frac{R_y(x) - \bar{R}_y}{\sqrt{\Sigma_x\left(R_y(x) - \bar{R}_y\right)^2}}$$

У даній формулі $R_y(x)$ – оцінка елемента *y* користувачам *x*, $\bar{R}_y$ – середня оцінка користувача *y* (сума всіх оцінок, виставлених даним користувачем, поділена на кількість оцінок, виставлених даним користувачем.

Тобто у знаменнику формули відбувається ліквідація першого людського фактору – більш та менш лояльних оцінок. А у чисельнику обраховується дисперсія оцінок користувача за формулою дисперсії дискретної випадкової величини.

Також для вирішення даної проблеми може застосовуватися MinMax нормалізація, значення якої у результаті будуть варіюватися від 0 до 1. Формула для обчислення нормалізованої оцінки у даному випадку виглядатиме:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Де x_{norm} – нормалізована оцінка, x_{min} – мінімальна оцінка користувача, x_{max} – максимальна оцінка користувача.

Однак у ході роботи над практичною частиною та впровадження нормалізації мною було виявлено, що застосування даних формул для оцінок

користувачів, які оцінили все однаково призводить до ділення на нуль. Для вирішення даної проблеми у ході практичної роботи було прийнято рішення при нульовому розходженні оцінок встановлювати оцінку користувача у середнє значення допустимої оцінки. Тобто $x_{norm}=r_{max}/2$, де r_{max}- максимальна допустима оцінка після нормалізації.

Іншим методом нормалізації є decoupling normalization. Даний метод нормалізації конвертує оцінку товару користувачем у імовірність, що цей товар подобається користувачеві. Він використовується у багатокритеріальній колаборативній фільтрації (multi-criteria collaborative filtering) [6]. Ідея даного типу колаборативної фільтрації полягає у моделюванні переваг користувачів більш детально, на основі оцінок про різні аспекти продукту чи послуги. Слід додати, що для даного типу рекомендаційних систем використовується не двовимірна, а 3-мірна матриця взаємодії, : User×Item×Criteriamatrix (Un×m×k). Для обрахування ймовірності того, що користувачеві подобається певний продукт, на основі його оцінок продукту за певними аспектами, використовуються 2 наступних припущення:

1) Коли велика кількість товарів має найбільші оцінки від користувача за параметром R, є ймовірність, що користувачу найбільше сподобаються товари, які мають максимальну оцінку по параметру R.
2) Коли більшість товарів оцінені як R стає менш ймовірно, що користувач надає перевагу товарам з категорії R.

Пояснення: припустимо маємо товар «Взуття» та 3 категорії, за якими проводиться оцінка даного товару: зручність, довговічність, краса. Дивимося, яке взуття користувач купує. Бачимо, що найвища оцінка користувача у категорії «Зручність», отже, можна зробити припущення, що користувач надає перевагу зручному взуттю і може знехтувати красою та довговічністю – ця частина пояснення відображає ідею першого припущення. Припустимо, користувач все ще обирає зручність, але зробивши аналіз товарів ми бачимо,

що більшість взуття на сайті є зручною, тобто сайт просто не продає незручного взуття. Отже, оскільки все взуття зручне, з'являється ймовірність, що це не основний критерій користувача, йому головне довговічність, просто висока зручність є поширеною для даного класу товарів.

На основі наведених припущень побудовано формулу для перетворення рейтингу предмету на ймовірність того, що користувач уподобає даний предмет.

$$p_y(A) = p_y(Rating \leq R) - \frac{p_y(Rating = R)}{2}$$

У даній формулі обраховується ймовірність події А, де А – користувачу подобаються товари, які мають максимальну оцінку за параметром R.$p_y(Rating < R)$– відсоток елементів, оцінених не вище R, $p_y(Rating = R)$ – відсоток елементів, оцінених так само як R.

Виконавши необхідну нормалізацію, можна перейти до обчислення схожості. Обчислення схожості є одним з ключових факторів для колаборативної фільтрації, адже саме від того, які користувачі будуть вважатися схожими для даного користувача, залежить список отриманих рекомендацій.

Найпримітивніший метод визначення схожості користувачів полягає у тому, що вони вподобали один і той самий товар. Множини вподобань перетинаються, отже, у користувачів є щось спільне. Проте, базуючись на такому методі, є ймовірність продукування багатьох нерелевантних рекомендацій, оскільки недостатньо визначити просто схожих користувачів, потрібно обрати користувачів, максимально схожих на даного користувача. Тут вводиться термін «міра схожості», обчислення якої розглядатиметься далі. Для обрахування міри схожості, наприклад, між двома користувачами (також можливий обрахунок міри схожості між товарами) користувачів відображають у вигляді векторів-оцінок ними товарів, далі для двох векторів-

користувачів обраховують схожість, використовуючи певну формулу для міри схожості.

У рекомендаційних системах можуть використовуватися наступні міри схожості:

- Косинусна міра
- Коефіцієнт кореляції Пірсона
- Евклідова відстань
- Манхетенська відстань тощо.

Значення кожної з мір схожості залежить від деяких специфічних ознак, тому, яку саме міру схожості обирати у рекомендаційній системі, можна сказати шляхом експериментів та перевірки релевантності отриманих значень. Також при виборі слід враховувати формат даних.

Обирається формула та обраховується міра схожості користувача, для якого формується рекомендація з кожним користувачем у системі.

Після обрахунку міри схожості можна переходити до формування рекомендацій, але з'являється одна суттєва деталь. Чи потрібно враховувати думку кожного з користувачів при формуванні рекомендацій для користувача? Здавалося б, що для точності не завадить урахувати оцінки всіх користувачів, але якщо схожість користувача X з даним користувачем мала, то, швидше за все, і вплив оцінок користувача X для даного користувача буде малим. Чим більше даних слід обрахувати, тим довше отримуватиметься відгук від системи. Саме тому перед тим, як передбачати рейтинг, додається ще один етап: вибір сусідів. Вибрати сусідів означає вирішити, чиї саме оцінки враховуватимуться при побудові рекомендацій. Варто зазначити, що даний етап є необов'язковим та може пропускатися, за умови невеликої кількості користувачів та товарів у системі або ж за умови наявності достатньої кількості ресурсів для виконання необхідних обрахунків. Ідея

даного етапу полягає в тому, що якщо користувачі абсолютно не схожі, то немає сенсу враховувати їх при обчисленні рекомендацій.

Є 4 традиційних підходи до вибору користувачів, оцінки яких будуть враховані [7]:

- Всі користувачі – доречно використовувати лише за умови, якщо вибірка даних є малою і не дуже навантажить систему
- Поріг – встановлюється поріг для отриманого значення міри схожості, враховуються лише оцінки користувачів, які пройшли поріг схожості з даним користувачем.
- Алгоритм k найближчих сусідів – користувачі сортуються за мірою схожості, після цього береться k перших користувачів.
- Виконується кластеризація користувачів після чого враховуються лише оцінки користувачів, які у тому ж кластері (використання даного методу переведе даний алгоритм до гібридних, оскільки кластеризація є алгоритмом, який використовується у model-based фільтрації та буде розглянута у відповідному розділі).

На основі одного з даних методів обирається множина користувачів, які будуть враховані при формуванні прогнозованих оцінок.

Після отримання схожих користувачів можна починати оцінку ймовірності того, що даний товар буде вподобаний даним користувачем. Перш за все, слід зазначити, що недоцільно було б рекомендувати користувачу те, що він вже оцінив, крім того, зменшення кількості товарів, які слід оцінити, зменшить навантаження на систему та пришвидшить формування рекомендацій, за рахунок зменшення кількості необхідних обрахунків.

Тому перший крок при оцінюванні товарів – виключити з оцінки вже оцінені товари. Далі обраховується вихідна оцінка за формулою:

$$R_{u,i} = k \sum_{u' \in U} sim(u, u')\, R_{u',i}$$

Де $R_{u,i}$ – прогнозована оцінка користувачем u товару i, $sim(u, u')$ - схожість користувача u з користувачем u', $R_{u',i}$ - оцінка користувачем u' товару i, k – нормалізований коефіцієнт.

З формули видно, що схожість користувачів являє собою ваговий коефіцієнт для їх оцінок.

Нормалізований коефіцієнт обчислюється за формулою:

$$k = \frac{1}{\sum_{u' \in U} |sim(u, u')|}$$

Після застосування даних формул ми отримаємо прогнозовану оцінку даного користувача для кожного з неоцінених ним раніше товарів. Далі залишається лише посортувати товари за спаданням оцінки та видати користувачу перші n товарів, де n – необхідна кількість. У випадку занадто низьких оцінок також можна видати користувачу лише товари, оцінка яких перейшла певний поріг.

2.1.2 Item-based filtration

Даний алгоритм надання рекомендацій схожий на user-based алгоритм, але порівняння виконується не за векторами-користувачами, а за векторами-товарами. Здавалося б, оскільки методи схожі, і є висока ймовірність отримати схожі результати, проте це не так. Можна сказати, що item-based підхід є більш узагальненим і менш персоніфікованим, тому що кількість оцінок користувачами одного товару зазвичай вище, ніж кількість всіх оцінок одного користувача. Даний метод бере до уваги зв’язок послуг між собою.

Щоб краще зрозуміти відмінність item-based фільтрації повернемося до кроків алгоритму згаданих у user-based підході. Отже:

1. Нормалізація оцінок – крок залишається незмінним.

2. Обчислення схожості – обчислюється схожість не між користувачем та іншими користувачами, а між товарами, які вподобав користувач та іншими товарами.
3. Вибір сусідів – обираються товари, найбільш схожі на товари, вподобані користувачем.
4. Оцінювання товарів – оцінка формується на основі схожості товарів та попередніх оцінок користувача.
5. Вибір товарів – крок залишається незмінним.

Також item-based підхід має ряд переваг. Вподобання користувачів з часом можуть змінюватися, але опис товарів, навіть якщо може зазнати змін, це буде не так часто. Оцінка міри схожості товарів більш точна, ніж оцінка схожості користувачів, оскільки маємо більше інформації для прийняття рішення.

2.2 Model-based алгоритми

Основною проблемою memory-based алгоритмів є проблема використання великих обсягів пам’яті для завантаження повної матриці взаємодії, а також проведення та збереження значень додаткових обчислень, проведених на цій матриці. За великих обсягів товарів та користувачів, продуктивність такої системи знижуватиметься. Для вирішення таких проблем було створено model-based підхід до формування рекомендацій. Він використовує 5 загальних методів: кластеризація, класифікація, латентна модель, Марковський процес вирішування (Markov decision process) та факторизація матриць[9]. Розглянемо детальніше деякі з цих методів:

2.2.1Кластеризація

Даний метод базується на припущенні, що користувачі з однієї групи мають однакові інтереси, тому вони оцінюють продукти однаково. Користувачів ділять на кластери (підгрупи, найбільш схожих користувачів).

Для поділу кожен користувач відображається як вектор оцінок всіх продуктів. Несхожість двох користувачів – це міра відстані між ними. Дану відстань обчислюють за допомогою формул Евклідової відстані, Манхетенської відстані або відстані Мінковського. Чим менша відстань, тим більша схожість між користувачами. Використання даного підходу дозволяє скоротити обчислення, оскільки пошук рекомендацій для користувача X здійснюється з використанням лише інших користувачів з того ж кластеру, що і користувач X.

Перед описом алгоритму введемо поняття «центроїд». Центроїд — це вектор, елементи якого являють собою середні значення кожної змінної, обрахованих по всім записам кластеру[8].

Найбільш популярним алгоритмом кластеризації є алгоритм k-means, який складається з таких кроків:

1. Випадково обрати k користувачів, кожен з яких відображатиме середнє значення (центроїд) певного кластера.

2. Для кожного користувача обрахувати відстань між ним і кожним з кластерів, віднести користувача до того кластера, з центроїдом якого у нього найменша відстань.

3. Перерахувати нове середнє значення (центроїд) кожного кластера та повторити крок 2 для розподілу користувачів. Повторювати кроки 2 та 3, доки система не стабілізується. Система стабілізована, якщо центроїди кластерів не змінилися після переобрахунку або якщо кожен з центроїдів змістився на достатньо мале значення (попередньо визначений поріг).

Для кластеризації основною проблемою є розрідженість матриці взаємодії користувача, яка може призводити до невірних значень при обчисленні кластерів. Для вирішення даної проблеми було запропоновано алгоритм, який спочатку групує схожі товари, а потім використовує групи товарів, щоб на їх основі сформувати групи користувачів.

2.2.2 Факторизація матриць

Факторизація матриць є однією з найбільш популярних методик генерації рекомендацій. Ідеєю даного підходу є розбиття user interacion матриці на добуток матриць меншої розмірності.

Ідеєю даного метода є те, що існують певні фактори, за допомогою яких можна унікально описати продукти системи. Ці фактори можна використати і для опису інтересів користувачів. За допомогою машинного навчання система самостійно визначатиме дані фактори у процесі навчання. Система, яка використовує даний метод, орієнтована на знаходження зв’язків, які можна виразити математично. Для того щоб описати подібні фактори більш зрозуміло, наведемо приклад: користувач високо оцінив книги A B і C, але ми знаємо, що ці книги належать до сегменту науково-фантастичних книг, таким чином ми визначили фактор, за яким користувач вподобав ці книги. Це був доволі узагальнений приклад і система, що використовує метод може знайти набагато специфічніші зв’язки між оцінками користувача та товарами, до деяких з подібних зв’язків людина могла б навіть не додуматися, але вони мають математичне обґрунтування під собою. Можна сказати що даний алгоритм прибирає шуми та залишає корисні дані.

User interaction матриця (R) розмірності n×m (n – кількість користувачів, k – кількість товарів) виражається як добуток двох матриць. U–матриця розмірності n×k, що відображає приховані фактори користувачів, і I–матриця розмірності k×m, що відображає приховані фактори продуктів.

Матриці Uта I шукаються за алгоритмом:

- Ініціалізувати задані матриці випадковими значеннями.
- Перемножити їх та порівняти з матрицею R. Обрахувати похибки для кожної комірки та загальну похибку.

- Використовуючи методи мінімізації помилок (серед яких метод найменших квадратів, градієнтний спуск та ін.) максимально наблизити добуток U*I до матриці R.

Оцінка користувача обчислюватиметься за формулою:

$$\tilde{r}_{ui} = \sum_{f=0}^{k} U_{u,f} I_{f,i}$$

Обчислення прогнозованої оцінки, базуючись на матрицях, отриманих у процесі факторизації, значно зменшить кількість обрахунків у зв’язку з меншою розмірністю матриць, що дозволить виконати меншу кількість операцій.

Розділ 3. Типові проблеми при побудові рекомендаційних систем

Будь-яка рекомендаційна система стикається з рядом проблем при обрахуванні прогнозованих оцінок. Це пов’язано з нестачею даних, обмеженнями ресурсів та програмного забезпечення та ін. У цьому розділі буде розглянуто типові проблеми та методи їх вирішення.

3.1 Збір відомих оцінок користувачів

Дана проблема породжена людським фактором. Користувачі рекомендаційних систем не завжди оцінюють придбані ними товари. Припустимо, користувач А придбав товар Б, але він не оцінив його. Таким чином система не може отримати інформацію, чи сподобався товар користувачеві (а якщо це система з оцінками, то наскільки товар сподобався) чи ні, що може вплинути на формування подальших рекомендацій.

Є 2 методи подолання даної проблеми:

- Явний: після покупки певного товару запропонувати користувачеві оцінити його.
- Неявний: спробувати передбачити оцінку даного товару після покупки користувачем, базуючись на інформації про його попередні вподобання.

Також, повертаючись до попереднього розділу, можна згадати метод системи Spotify, який хоч і не використовує оцінки, проте, базуючись на тривалості прослуховувань та кількості відтворень, може зробити висновки щодо того, чи подобається даний трек користувачу.

3.2 Холодний старт

На відміну від попередньої проблеми, холодний старт - це проблема самої системи, а не щось, що базується на діях користувачів.

Є 3 варіації даної проблеми:

- Новий користувач приходить до системи. Система не має жодної інформації про нього, отже, вона не зможе визначити користувачів, схожих на нього, та сформувати рекомендації.
- Новий товар додається до системи. Аналогічно до попереднього пункту ніхто не оцінював даний товар, тому він нікому не буде рекомендований. Далі даний товар стає непопулярним.
- Перший запуск системи. Матриця взаємодії у даній системі абсолютно порожня, оскільки ніхто з нею ще не взаємодіяв.

Методи вирішення:

- Коли новий користувач доєднується до системи, можна явно попросити його оцінити деякі товари/категорії – зібрати інформацію про вподобання користувача, яка дозволить знайти схожих на нього користувачів та сформувати рекомендації. Часто, реєструючись на тих чи інших сайтах, ви бачите, як у вас просять вказати жанри фільмів, пісень чи книг, улюблених авторів та ін. Всі ці дані дають системі можливість зібрати інформацію про користувача ще до залишення оцінок. Також можна використати інформацію про геолокацію користувача або просто рекомендувати товари, які зараз є найпопулярнішими.
- При додаванні нового товару слід звернутися до методів рекомендацій на основі вмісту. Новий товар оцінюється за певними атрибутами, які є вагомими у рекомендаційній системі, та отримує свою гіпотетичну оцінку.
- Щодо останнього варіанту, тут можливими рішеннями є застосування методів машинного навчання. Знову ж таки, повертаючись до прикладу Spotify, так чи інакше можна сформувати рейтинг популярності тих чи інших продуктів, спираючись на інформацію, яку можна знайти в мережі «Інтернет». А щодо

користувачів, то для збору інформації про них підійдуть вище зазначені методи.

3.3 Масштабованість

Чим більше у системі користувачів, тим більше обрахунків їй доведеться виконувати. Важливо усвідомлювати, що система яка дає швидкий та релевантний результат, коли нею користується 1000 користувачів, може погіршити свою роботу, якщо кількість користувачів збільшиться до кількох сотень тисяч або мільйону.

Дану проблему можна вирішувати одразу у двох напрямах:

- Удосконалювати обладнання (використовувати більш потужні сервери, процесори, збільшувати кількість серверів тощо)
- Програмне забезпечення (удосконалення використовуваних алгоритмів, перехід до методів, які краще працюють на великих кількостях користувачів при цьому менше навантажуючи систему)

3.4 Надмірна спеціалізація

Ця проблема виникає, коли продукти, що рекомендуються занадто схожі один на одного. Наприклад, користувач постійно купує бакалію на певному вебсайті. Після цього йому рекомендується лише цукор (різних брендів, різних типів, але лише одна категорія товарів), базуючись на найчастіших покупках.

Одним з можливих рішень даної проблеми є рекомендація користувачеві товарів, які його можуть зацікавити, але не схожі на те, що він зазвичай купує (recommendation diversification).

Розділ 4. Практична частина

У якості практичної частини було вирішено порівняти рекомендації, що будуть видаватися з використанням колаборативної фільтрації. Для порівняння відстань між користувачами обраховувалася на основі косинусної міри та евклідової відстані. Також була реалізована можливість виконувати або не виконувати нормалізацію даних перед обрахунком відстані. Для нормалізації було обрано алгоритм MinMax наведений (див. розділ 2).

Для реалізації алгоритму рекомендацій було обрано мову Python, оскільки вона містить засоби, що дозволяють спростити обробку матриць, що є важливим, оскільки алгоритм працює з user interaction матрицею.

Робота алгоритму була протестована на дата-сеті MovieLens від дослідницької групи GroupLens (див. розділ 1). Для тестування алгоритму було використано частину дата-сету, оскільки робота проводилася з використанням оперативної пам’яті, у яку не вдалося завантажити повну матрицю для цілого дата-сету.

Першим кроком було завантажено дані до системи (див. Додаток 1). Дані у файлах зберігалися у форматі csv.

У файлі movies.csv значення movieId – ідентифікатор фільму, title – заголовок фільму, genres – список жанрів, розділених знаком «|» (див. Додаток 1). Всього у файлі можуть міститися дані про 9742 фільми.

У файлі ratings.csv userId (ідентифікатор користувача), movieId (ідентифікатор фільму), rating (оцінка), timestamp (дата оцінювання) (див. Додаток 1). Всього у файлі міститься 100836 оцінок користувачів.

Після конвертації інформації, що зберігається у даних файлах було отримано user interaction матрицю (рис. 4.1). Як видно з рисунку, матриця є доволі розрідженою. Це пов’язано, по-перше, з тим, що було взято не повний дата-сет, а його частину. По-друге, як було згадано вище (розділ 2)

розрідженість матриці є наслідком великої кількості користувачів та великої кількості товарів, зрозуміло, що у реальних умовах ситуація з заповненістю матриці буде приблизно такою ж.

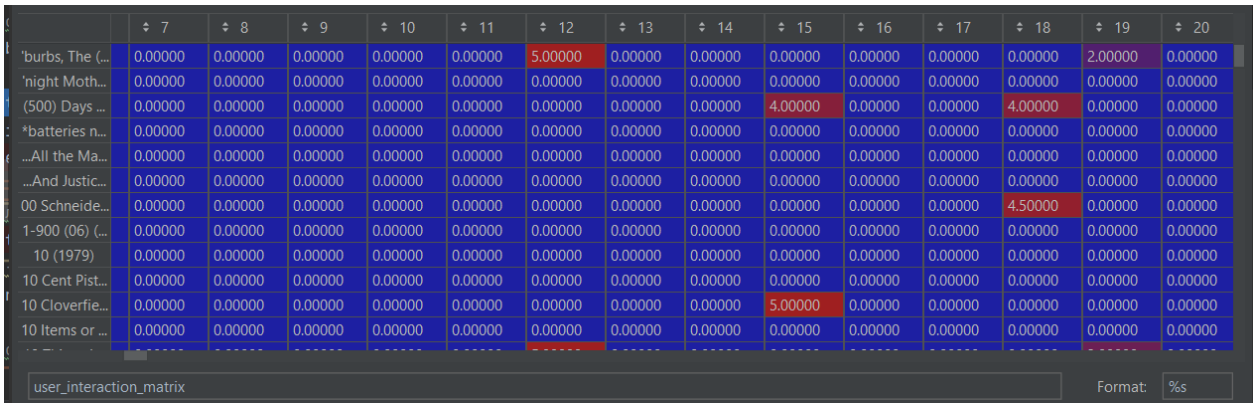

Рисунок 4.1. User interaction матриця

Щодо алгоритму, що був застосований до матриці, оскільки у розділі 2 детально розглядалася user-based колаборативна фільтрація, у практичній частині було вирішено детальніше розглянути item-based фільтрацію.

Точкою входження до обчислень є функція count_recommendations (див. Додаток 1), яка приймає список параметрів: user (ідентифікатор користувача), num_neighbors (кількість сусідів, що враховуються при обрахунку рекомендацій), num_recommendation (бажана кількість отриманих рекомендацій), distance_metric (яку формулу обраховувати при обрахунку відстаней між фільмами), do_normalization (Boolean змінна, яка визначає виконувати нормалізацію методом MinMax чи ні).

Першим кроком при обрахунках є виконання нормалізації (функція normalize_ratings). Під час роботи програми мною було помічено, що незалежно від того використовується для нормалізації метод Ґаусса чи MinMax, якщо оцінки користувача не мали розходження, то буде отримано нуль у знаменнику. Тому у функції нормалізації було додатково перевірено дану умову (див. Додаток 1).

Наступним етапом з алгоритму, наведеного у підрозділі 2.1.1, є обрахування схожості. У даному алгоритмі для кожного з фільмів, переглянутих заданим користувачем, обраховується його схожість з іншими фільмами (див. Додаток 1). Варто враховувати, що у випадку роботи з розрідженою матрицею, фільми, у яких доволі мало оцінок, можуть бути оцінені як схожі, оскільки у них багато спільних неоцінених комірок.

Після обрахунку відстаней між фільмами починається прогнозування оцінок на неоцінені фільми (рис 7). Для кожного з фільмів, не переглянутих даним користувачем, беруться індекси фільмів, найближчих до нього. Далі відстань конвертується у схожість (чим більша відстань, тим менша схожість) (див. Додаток 1).

Далі обраховується прогнозована оцінка як сума добутків міри прогнозованого схожості фільму з даним фільмом на оцінку даного фільму. І прогнозована оцінка записується у результуючу матрицю.

Останнім кроком є взяти з результуючої матриці фільми, які ще не були оцінені користувачем, посортувати їх за спаданням та взяти перші n, де n–необхідна кількість фільмів для рекомендацій.

Далі представлена порівняльна таблиця схожості результатів для різних алгоритмів у відсотках (див. Додаток 2). Порівнювалися перші 1000 рекомендованих фільмів.

З отриманих результатів можна зробити висновок про залежність отриманих результатів від використовуваних алгоритмів. Також можна висунути припущення про те, що на значну різницю при прогнозуванні впливає не лише обраний алгоритм, а й те, що матриця є розрідженою. Розрідженість матриці підсилює ефект того, що різні міри відстані спираються на різні фактори. Слід зауважити, що результати одного й того самого алгоритму з застосуванням нормалізації та без є доволі близькими, що можна пояснити незначним розходженням користувацьких оцінок.

Для того, щоб продемонструвати прогнозовані оцінки було створено тестові дані з десятьма користувачами та десятьма фільмами та виконано обчислення на тестових даних (див. Додаток З).

Висновок

У роботі було досліджено методологію розробки рекомендаційних систем. Обґрунтовано необхідність впровадження даних систем. Досліджено основні переваги, недоліки та причини для застосування необхідних кроків кожного зі згаданих алгоритмів.

Крім того, для усвідомлення того, як все працює у реальному житті, було розглянуто існуючі рекомендаційні системи та детально проаналізовано підхід до побудови рекомендацій кожної з них.

Рекомендаційні системи досі розвиваються, впровадження методів машинного навчання, нейронних мереж та ускладнення алгоритмів триває. На прикладі рекомендаційної системи, проаналізованої у розділі 1, було продемонстровано як комбінації існуючих алгоритмів допомагають покращувати результат.

Також було проаналізовано проблеми, з якими стикається рекомендаційна система, серед яких збір даних користувача, розрідженість матриць, нестача ресурсів програмного забезпечення, холодний старт та власне вибір алгоритму, який буде підходити для продуктів, які має рекомендувати система. Розглянуто можливі рішення даних проблем, серед яких неявний збір даних, використання всіх можливих даних, робота у кількох режимах, збільшення кількості обчислювальних ресурсів.

У практичній частині реалізовано алгоритм рекомендацій, проаналізовано причини відмінностей між рекомендаціями для різних алгоритмів та обґрунтовано необхідність підбору алгоритму саме орієнтуючись на конкретну систему, тип контенту, кількість даних.

Для подальших досліджень можна виконати аналіз model-basedалгоритмів, а також їх комбінацій з memory-based.

Перелік використаних джерел

[1] - Recommendation Systems: Applications and Examples in 2022 [Електронний ресурс] https://research.aimultiple.com/recommendation-system/

[2] - C.C. Aggarwal: RecommenderSystems: TheTextbook – Springer, 2016

[3] -UncoveringHowtheSpotifyAlgorithm Works[Електронний ресурс] https://towardsdatascience.com/uncovering-how-the-spotify-algorithm-works-4d3c021ebc0

[4] - A StudyofMethodsforNormalizingUserRatingsin CollaborativeFiltering, RongJin, LuoSi [Електронний ресурс] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.3197&rep=rep1&type=pdf

[5] –Variance [Електронний ресурс]https://mathworld.wolfram.com/Variance.html

[6] -Improving Accuracyof Multi-Criteria Collaborative Filteringby Normalizing User Ratings,Alper BİLGE, Alper YARGIÇ, [Електронний ресурс] https://dergipark.org.tr/tr/download/article-file/288637

[7] -User-UserCollaborativeFilteringForJokesRecommendation, [Електронний ресурс]https://towardsdatascience.com/user-user-collaborative-filtering-for-jokes-recommendation-b6b1e4ec8642

[8] -InterpretallstatisticsandgraphsforCluster K-Means, [Електронний ресурс] https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/multivariate/how-to/cluster-k-means/interpret-the-results/all-statistics-and-graphs/

[9] - Model-basedapproachforCollaborativeFiltering, August 2010[Електронний ресурс] https://www.researchgate.net/publication/321753015_Model_based_approach_for_Collaborative_Filtering

Додатки

Додаток 1. Реалізація алгоритму

Завантаження даних у систему:

```
# завантаження таблиць фільмів і рейтингів від користувачів у систему
ratings = pd.read_csv('ratings.csv', usecols=['userId', 'movieId', 'rating'])
movies = pd.read_csv('movies.csv', usecols=['movieId', 'title'])
```

Структура файлу movies.csv:

movies.csv

```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
6,Heat (1995),Action|Crime|Thriller
7,Sabrina (1995),Comedy|Romance
8,Tom and Huck (1995),Adventure|Children
9,Sudden Death (1995),Action
```

Структура файлу ratings.csv

ratings.csv

```
userId,movieId,rating,timestamp
1,1,4.0,964982703
1,3,4.0,964981247
1,6,4.0,964982224
1,47,5.0,964983815
1,50,5.0,964982931
1,70,3.0,964982400
1,101,5.0,964980868
1,110,4.0,964982176
1,151,5.0,964984041
```

Функція обрахунку рекомендацій. Нормалізація і отримання відстаней:

```
def count_recommendations(user, num_neighbors, num_recommendation, distance_metric, do_normalization):
    # кількість сусідів, обраних за найвищою схожістю, що враховуватиметься при прогнозуванні
    number_neighbors = num_neighbors
    # копія user interaction matrix для обчислення в ній predicted рейтингів
    user_interaction_matrix_copy = user_interaction_matrix.copy()
    if do_normalization:
        normalize_ratings(user_interaction_matrix_copy)

    # пошук найближчих фільмів-сусідів з використанням косинусної міри схожості
    knn = NearestNeighbors(metric=distance_metric, algorithm='brute').fit(user_interaction_matrix_copy.values)
    # distances - відстань між фільмами (на перетині рядка x і стовпчика y відстань між фільмами), indices - індекси
    # (на перетині рядка x і стовпчика y індекс фільму з яким обраховувалася схожість)
    distances, indices = knn.kneighbors(user_interaction_matrix_copy.values, n_neighbors=number_neighbors)

    # індекс поточного користувача у списку користувачів
    user_index = user_interaction_matrix.columns.tolist().index(user)
```

Нормалізація оцінок:

```
def normalize_ratings(matrix_to_normalize):
    for user_index in matrix_to_normalize.columns:
        movie_names = matrix_to_normalize[user_interaction_matrix[user_index] != 0].index.tolist()
        user_ratings = matrix_to_normalize[user_index].values.tolist()
        min_rating = min(user_ratings)
        max_rating = max(user_ratings)
        max_min_difference = max_rating - min_rating
        for movie_index in movie_names:
            if max_min_difference == 0:
                matrix_to_normalize.at[movie_index, user_index] = 0.5
            else:
                matrix_to_normalize.at[movie_index, user_index] = (matrix_to_normalize.loc[movie_index][
                                                                   user_index] - min_rating) / max_min_difference
```

Функція обрахунку рекомендацій. Обрахунок прогнозованої оцінки:

```
for movie_index, movie_title in list(enumerate(user_interaction_matrix.index)):
    # якщо на перетині фільму і користувача немає оцінки - потрібно спрогнозувати оцінку заданого фільму
    if user_interaction_matrix.iloc[movie_index, user_index] == 0:
        # за індексом фільму дістаємо індекси схожих на нього фільмів
        sim_movies = indices[movie_index].tolist()
        # за індексом фільму дістаємо індекси між ним та схожими фільмами
        movie_distances = distances[movie_index].tolist()

        #обрахунок схожості на основі відстані (чим більша відстань тим менша схожість
        movie_similarity = convert_to_similarity(movie_distances, distance_metric)
        # копіюємо таблицю схожості для обрахунків
        movie_similarity_copy = movie_similarity.copy()
        nominator = 0

        # для кожного фільму з подібних фільмів
        for s in range(0, len(movie_similarity)):
            # якщо рейтинг фільму на який схожий ще неоцінений фільм - 0 видаляємо його зі списку порівнянь
            if user_interaction_matrix.iloc[sim_movies[s], user_index] == 0:
                movie_similarity_copy.pop(s - (len(movie_similarity) - len(movie_similarity_copy)))
            # інакше додаємо добуток схожості фільму на оцінку користувачем схожого фільму
            else:
                nominator += movie_similarity[s] * user_interaction_matrix.iloc[sim_movies[s], user_index]
        # встановлюємо нову передбачену оцінку для фільму схожого на один з фільмів, що оцінив користувач
        if len(movie_similarity_copy) > 0:
            if sum(movie_similarity_copy) > 0:
                user_interaction_matrix_copy.iloc[movie_index, user_index] = nominator / sum(movie_similarity_copy)
```

Обрахунок схожості:

```
def convert_to_similarity(movie_distances, distance_metric):
    if distance_metric == 'cosine':
        return [1 - x for x in movie_distances]
    return [1 / (1 + x) for x in movie_distances]
```

Додаток 2. Порівняльна таблиця схожості рекомендацій

	Косинусна міра схожості	Косинусна міра схожості з попередньою нормалізацією	Евклідова відстань	Евклідова відстань з попередньою нормалізацією
Косинусна міра схожості		99.2%	61.2%	61.1%
Косинусна міра схожості з попередньою нормалізацією	99.2%		60.9%	60.8%
Евклідова відстань	61.2%	60.9%		99.8%
Евклідова відстань з попередньою нормалізацією	61.1%	60.8%	99.8%	

Додаток 3. Рекомендації отримані на тестових даних

Вхідні оцінки користувачів:

	1	2	3	4	5	6	7	8	9	10
Father of the Bride Part II (1995)	4.00000	0.00000	0.00000	0.00000	5.00000	5.00000	5.00000	4.00000	0.00000	5.00000
Grumpier Old Men (1995)	4.00000	0.00000	5.00000	4.00000	0.00000	5.00000	4.00000	4.00000	3.00000	0.00000
Jumanji (1995)	0.00000	5.00000	4.00000	5.00000	0.00000	3.00000	5.00000	0.00000	5.00000	0.00000
Toy Story (1995)	4.00000	3.00000	5.00000	0.00000	4.00000	5.00000	0.00000	4.00000	5.00000	4.00000
Waiting to Exhale (1995)	0.00000	5.00000	0.00000	4.00000	0.00000	4.00000	5.00000	0.00000	5.00000	0.00000

Оцінки отримані з використанням косинусної міри схожості без нормалізації:

	1	2	3	4	5	6	7	8	9	10
Father of th...	4.00000	3.70601	5.00000	4.00000	5.00000	5.00000	5.00000	4.00000	4.27928	5.00000
Grumpier O...	4.00000	4.30324	5.00000	4.00000	4.00000	5.00000	4.00000	4.00000	3.00000	4.00000
Jumanji (19...	4.00000	5.00000	4.00000	5.00000	4.00000	3.00000	5.00000	4.00000	5.00000	4.00000
Toy Story (...	4.00000	3.00000	5.00000	4.43406	4.00000	5.00000	4.63401	4.00000	5.00000	4.00000
Waiting to ...	4.00000	5.00000	4.54218	4.00000	4.00000	4.00000	5.00000	4.00000	5.00000	4.00000

Оцінки отримані з використанням Евклідової відстані

	1	2	3	4	5	6	7	8	9	10
Father of th...	4.00000	3.87509	5.00000	4.00000	5.00000	5.00000	5.00000	4.00000	4.31325	5.00000
Grumpier O...	4.00000	4.33307	5.00000	4.00000	4.00000	5.00000	4.00000	4.00000	3.00000	4.00000
Jumanji (19...	4.00000	5.00000	4.00000	5.00000	4.00000	3.00000	5.00000	4.00000	5.00000	4.00000
Toy Story (...	4.00000	3.00000	5.00000	4.44705	4.00000	5.00000	4.63767	4.00000	5.00000	4.00000
Waiting to ...	4.00000	5.00000	4.47930	4.00000	4.00000	4.00000	5.00000	4.00000	5.00000	4.00000

Вхідні оцінки після виконання нормалізації

	1	2	3	4	5	6	7	8	9	10
Father of th...	1.00000	0.00000	0.00000	0.00000	1.00000	1.00000	1.00000	1.00000	0.00000	1.00000
Grumpier O...	1.00000	0.00000	1.00000	0.80000	0.00000	1.00000	0.80000	1.00000	0.60000	0.00000
Jumanji (19...	0.00000	1.00000	0.80000	1.00000	0.00000	0.00000	1.00000	0.00000	1.00000	0.00000
Toy Story (...	1.00000	0.60000	1.00000	0.00000	0.80000	1.00000	0.00000	1.00000	1.00000	0.80000
Waiting to ...	0.00000	1.00000	0.00000	0.80000	0.00000	0.50000	1.00000	0.00000	1.00000	0.00000

Оцінки отримані з використанням косинусної міри схожості та нормалізації (враховувати при перегляді лише нові оцінки):

	1	2	3	4	5	6	7	8	9	10
Father of th...	1.00000	3.59751	5.00000	4.00000	1.00000	1.00000	1.00000	1.00000	4.22809	1.00000
Grumpier O...	1.00000	3.87651	1.00000	0.80000	4.46496	1.00000	0.80000	1.00000	0.60000	4.46496
Jumanji (19...	4.00000	1.00000	0.80000	1.00000	4.00000	0.00000	1.00000	4.00000	1.00000	4.00000
Toy Story (...	1.00000	0.60000	1.00000	4.36517	0.80000	1.00000	4.60704	1.00000	1.00000	0.80000
Waiting to ...	4.00000	1.00000	4.51650	0.80000	4.00000	0.50000	1.00000	4.00000	1.00000	4.00000

Оцінки отримані з використанням Евклідової відстані та нормалізації (враховувати при перегляді лише нові оцінки):

	1	2	3	4	5	6	7	8	9	10
Father of th...	1.00000	3.88114	5.00000	4.00000	1.00000	1.00000	1.00000	1.00000	4.30670	1.00000
Grumpier O...	1.00000	3.94611	1.00000	0.80000	4.47789	1.00000	0.80000	1.00000	0.60000	4.47789
Jumanji (19...	4.00000	1.00000	0.80000	1.00000	4.00000	0.00000	1.00000	4.00000	1.00000	4.00000
Toy Story (...	1.00000	0.60000	1.00000	4.00000	0.80000	1.00000	4.63618	1.00000	1.00000	0.80000
Waiting to ...	4.00000	1.00000	4.54347	0.80000	4.00000	0.50000	1.00000	4.00000	1.00000	4.00000