НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

До захисту допущено
Завідувач кафедри
___Оксана ТИМОЩУК
«__»___2025 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему «Побудова рекомендаційної системи фільмів з урахуванням вподобань користувачів та мета-інформації про фільм»

Виконав: студент IV курсу, групи КА-12
Дроздов Нікіта Андрійович _____

Керівник: доцент, к.ф.-м.н.
Подколзін Гліб Борисович _____

Консультант з економічного розділу: доцент, к.е.н.
Рощина Надія Василівна _____

Консультант з нормоконтролю: доцент, к.ф.-м.н.
Статкевич Віталій Михайлович _____

Рецензент: доцент, к.ф.-м.н.
Орловський Ігор Володимирович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент_____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Оксана ТИМОЩУК
«___»_____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту
Дроздову Нікіті Андрійовичу

1. Тема роботи «Побудова рекомендаційної системи фільмів з урахуванням вподобань користувачів та мета-інформації про фільм», керівник роботи Подколзін Гліб Борисович, кандидат фізико-математичних наук, доцент кафедри ММСА, затверджені наказом по університету від «26» 05 2025 р. № 1759-с.
2. Термін подання студентом роботи 13.06.2025.
3. Вихідні дані до роботи: датасет MovieLens ml-latest-small, метаінформація про фільми, рейтинги користувачів, методи колаборативної та контентної фільтрації, бібліотеки Python (scikit-learn, pandas, surprise), метрики оцінки якості (RMSE, MAE), матеріали переддипломної практики.
4. Зміст роботи: порівняльний аналіз сучасних підходів до побудови рекомендаційних систем, розробка рекомендаційних систем, оцінювання показників ефективності при реалізації, функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): графічні схеми застосованих методів, формальний опис обраних методів, порівняльні таблиці метрик точності, візуалізація результатів роботи програми, презентація до захисту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна, доц., к.е.н.		

7. Дата видачі завдання: 17.03.2025

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Дослідження предметної області та визначення цілей і завдань роботи	31.03.2025 - 06.04.2025	виконано
2	Опрацювання наукової літератури	07.03.2025 - 13.04.2025	виконано
3	Розробка математичних моделей і написання програмного коду	14.04.2025 - 18.05.2025	виконано
4	Робота над розділами диплому в ході переддипломної практики	14.04.2025 - 18.05.2025	виконано
5	Узгодження пояснювальної записки та розділів роботи	19.05.2025 - 01.06.2025	виконано
6	Попередній захист дипломної роботи	02.06.2025 - 15.06.2025	виконано
7	Захист дипломної роботи	16.06.2025 - 22.06.2025	виконано

Студент Нікіта ДРОЗДОВ

Керівник Гліб ПОДКОЛЗІН

РЕФЕРАТ

Дипломна робота: 82 с., 22 рис., 7 табл., 2 додатки, 17 джерел

Ключові слова: РЕКОМЕНДАЦІЙНА СИСТЕМА, ПРОГРАМНА РЕАЛІЗАЦІЯ, КОНТЕНТНА ФІЛЬТРАЦІЯ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ, МАТРИЧНА ФАКТОРИЗАЦІЯ.

Об’єктом дослідження є платформи та сервіси для перегляду фільмів і серіалів з можливістю персоналізованих рекомендацій.

Предметом дослідження є алгоритми рекомендацій, їхні методи та можливість оптимізації.

Метою роботи є дослідження побудови рекомендаційних систем, найефективніших підходів до роботи з різними обсягами даних, та реалізація власної рекомендаційної системи.

У роботі розглянуто сучасні підходи до побудови рекомендаційних систем, реалізовано модель колаборативної фільтрації за допомогою алгоритму SVD та контентну модель на основі векторного представлення фільмів із використанням TF-IDF. Запропоновано метод поєднання результатів моделей у вигляді гібридної системи. Система тестувалася на наборі даних MovieLens ml-latest-small.

Для оцінки якості побудованих моделей використано метрики RMSE та MAE. Програмну реалізацію виконано мовою Python.

Методологія дослідження базується на застосуванні методів машинного навчання, аналізу латентних факторів, обробки текстових даних.

Практичне значення роботи полягає у створенні універсального та розширюваного підходу до побудови рекомендаційних систем, який поєднує сильні сторони різних підходів — точність колаборативної фільтрації та гнучкість контентного аналізу. Отримані результати можуть бути використані для впровадження в стримінгові сервіси.

ABSTRACT

Thesis: 82 pages, 22 tables, 7 figures, 2 appendices, 17 references.

Keywords: RECOMMENDATION SYSTEM, SOFTWARE IMPLEMENTATION, CONTENT FILTERING, COLLABORATIVE FILTERING, FORECASTING, MACHINE LEARNING, MATRIX FACTORIZATION.

The object of the study is platforms and services for watching movies and TV series with the possibility of personalized recommendations.

The subject of the study is recommendation algorithms, their methods and the possibility of optimization.

The purpose of the work is to study the construction of recommendation systems, the most effective approaches to working with different amounts of data, and the implementation of our own recommendation system.

The work considers modern approaches to building recommendation systems, implements a collaborative filtering model using the SVD algorithm and a content model based on a vector representation of movies using TF-IDF. A method for combining the results of models in the form of a hybrid system is proposed. The system was tested on the MovieLens ml-latest-small dataset.

To assess the quality of the constructed models, the RMSE and MAE metrics were used. The software implementation was performed in Python.

The research methodology is based on the application of machine learning methods, latent factor analysis, and text data processing.

The practical significance of the work lies in creating a universal and extensible approach to building recommender systems, which combines the strengths of different approaches - the accuracy of collaborative filtering and the flexibility of content analysis. The results obtained can be used for implementation in streaming services.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

РС – Рекомендаційна Система

MBF – Metadata-based filtering (фільтрація на основі мета-даних)

CF – Collaborative filtering (колаборативна фільтрація)

MF – Матрична факторизація

LLM – Large Language Model

ВСТУП

В наш час, коли кількість інформації в мережі Інтернет зростає з неймовірною швидкістю, люди постійно стикаються з проблемою споживання — що переглянути, прочитати чи купити. Часто процес вибору продукту займає навіть більше часу, ніж його споживання (особливо фільмів та музики). Саме через це зростає попит на рекомендаційні системи — інструменти, що допомагають проаналізувати, відсортувати та впорядкувати інформацію, та подати людині саме те, що їй сподобається.

Рекомендаційні системи (скорочено РС) створені для того, щоб фільтрувати надвеликі динамічні потоки даних, і на основі вподобань користувача чи інших користувачів з подібними вподобаннями, пропонувати персоналізований контент [1]. Завдяки алгоритмам платформи та сервіси можуть запропонувати саме те, що сподобається людині.

Також зазначимо, що рекомендаційні системи є фінансово вигідними для платформ та сервісів, адже допомагають утримувати чи залучати аудиторію [2]. Так наприклад, часта та влучна рекомендація легко робить клієнта “постійним”, бо він не втрачає зайвий час та завжди задовольняє свій запит. До того ж, РС продемонстрували свою ефективність у допомозі з прийняттям рішень — зокрема там, де вибір великий або складний [3].

В e-com сфері рекомендаційні системи займають роль персонального інтелектуального помічника: вони знають вподобання користувача і підказують, що саме йому сподобається. У випадку нового клієнта (далі – холодного старту) РС виконують соціальну функцію, пропонуючи рішення із урахуванням колективного досвіду [4].

Усе вищезгадане суттєво зменшує інформаційне навантаження на користувача, робить його взаємодію з платформою чи сервісом зручнішою, а контент — більш влучним та персоналізованим.

Разом з технологіями, світ розвиває рекомендаційні системи та шукає нові методи для їхнього покращення. Насьогодні найбільш поширеними є підходи, засновані на фільтрації на основі мета-даних (metadata-based filtering), колаборативній фільтрації (collaborative filtering) або їх комбінаціях у вигляді гібридних моделей [5].

Саме тому тема дипломної роботи — «Побудова рекомендаційної системи фільмів з урахуванням вподобань користувачів та мета-інформації про фільм» — надзвичайно актуальна, адже кількість фільмів зростає, а увага та час обмежені.

Мета дослідження — дослідити як будуються РС, які підходи є найефективнішими при роботі з різними обсягами даних, та реалізувати власну рекомендаційну систему.

Основні завдання роботи:

1. Проаналізувати існуючі методи побудови рекомендаційних систем.
2. Визначити, які з методів найкраще підходять для роботи з різними за обсягом даними.
3. Реалізувати та протестувати власну РС.

Методи дослідження базуються на поєднанні теорії фільтрації, машинного навчання та основ побудови веб-систем.

Об’єктом дослідження є платформи та сервіси для перегляду фільмів і серіалів з можливістю персоналізованих рекомендацій.

Предметом дослідження є алгоритми рекомендацій, їхні методи та можливість оптимізації.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна інформація та ознайомлення

Головним призначенням рекомендаційних систем є вирішення та допомога у проблемі вибору продукту та скороченні часу на пошук необхідних товарів чи послуг. РС пропонують взаємовигідну співпрацю споживачам, адже клієнти отримують саме те, що їм потрібно, а платформи – прибуток завдяки кращому таргету та збільшенню продажів як результат.

Також зазначимо, що успіх будь-якої платформи чи сервісу напряму залежить від того, наскільки якісно та влучно її РС підбирає товари для користувача. Грамотний аналіз та персоналізація тут є головними факторами. Саме через це вибір та роботоспроможність алгоритму для формування рекомендацій є критично важливим.

Насьогодні будь–які рекомендації можна класифікувати на дві основні групи: персоналізовані та неперсоналізовані.

Гарним прикладом неперсоналізованих рекомендацій є реклама найпопулярніших товарів. Так наприклад, користуючись будь–якою інтернет–платформою, після пошукового запиту ми маємо змогу впорядкувати товари за популярністю на даній платформі. Хоч система і не може гарантувати, що найбільш проданий товар ідеально відповідатиме нашим потребам, все ж логічним допущенням буде, що товар, який обирає більшість клієнтів, з вищою ймовірністю буде нам цікавим, ніж той, що поки не має жодного покупця.

Як приклад персоналізованих рекомендацій розглянемо будь–які соціальні мережі: Instagram, TikTok тощо. Система зчитує дії користувачів та на основі

переглядів, підписок та інших взаємодій пропонує рекомендації: за ким почати стежити, що подивитися тощо. Саме такі рекомендації є найбільш влучними.

Для вибору алгоритму варто враховувати усю специфіку та параметри платформи. Нерідко навіть відмінно злагоджений алгоритм певного сервісу буде неефективним чи навіть непотрібним для іншого. Через це кожна РС під кожну задачу є особливою [5].

1.2 Актуальність проблеми

В наш час, коли кількість інформації в мережі Інтернет зростає з неймовірною швидкістю, люди постійно стикаються з проблемою споживання — що переглянути, прочитати чи купити. Часто процес вибору продукту займає навіть більше часу, ніж його споживання (особливо фільмів та музики). Саме через це зростає попит на рекомендаційні системи — інструменти, що допомагають проаналізувати, відсортувати та впорядкувати інформацію, та подати людині саме те, що їй сподобається [1].

Надважливою задачею є надати клієнту влучну пропозицію ще на початку користування та підтримувати цей рівень протягом усієї комунікації. На жаль, так трапляється на завжди: клієнт втрачає час та емоції, а сервіс – гроші. Для запобігання цьому необхідно впроваджувати ефективні рекомендаційні системи, що унеможливлять такі негативні сценарії взаємодії

Також варто зазначити проблематику нових фільмів: без досвіду перегляду серед знайомих чи близьких складно отримати відкритий відгук щодо новинки. І це є однією з причин впровадження РС, адже система збирає користувацький досвід людей зі схожими смаками та прогнозує: сподобається Вам фільм чи ні.

1.3 Історія та класифікація рекомендаційних систем

1.3.1 Історія створення

Перші рішення для автоматизованого надання рекомендацій з’явились у 90–х роках XX століття у відповідь на потребу у впорядкуванні великих обсягів даних новин, онлайн – бібліотек і магазинів.

Однією з перших РС була система GroupLens від однойменної дослідницької лабораторії, створена у 1994. Система збирала відгуки читачів мережі Usenet, де користувачі спілкувались та обмінювались новинами. Саме завдяки цьому GroupLens змогла прогнозувати реакції читачів на новини перш, ніж вони опублікуються.

Цей механізм фільтрації став першим автоматизованим механізмом колаборативної фільтрації. В ній використовувалися алгоритми для автоматичного формування прогнозів на основі історичних даних. Загальна система отримала назву рекомендувача GroupLens, що є назвою як для цієї системи рекомендацій, так і для всієї науково–дослідницької лабораторії в Університеті Міннесоти, США.

Наразі GroupLens спеціалізується на соціальних обчисленнях, системах рекомендацій та поведінки користувачів на онлайн – платформах. Команда реалізувала багато проектів, які стали не тільки цінним джерелом даних, а й досі діючими сервісами, що приносять користь людям. Розглянемо декілька:

1. MovieLens – рекомендаційна платформа фільмів, що працює з 1997 року і до сьогодні використовується для оцінювання фільмів та отримання персоналізованих рекомендацій на основі колаборативної фільтрації.

Також платформа надає відкриті датасети, які будуть використані в цій роботі.

2. BookLens – рекомендаційна система книжок.

3. LensKit – відкрита бібліотека для створення РС.

4. Cyclopath – краудсорсингова карта велосипедних маршрутів.

1.3.2 Класифікація РС

Сучасні РС можна класифікувати за кількома критеріями, зважаючи на тип використовуваних даних, тип і механізм роботи та рівень персоналізації [6].

1. Колаборативні системи: базуються на аналізі вподобань груп користувачів, також розділяються на два підтипи: user-based, де система прогнозує вподобання на основі користувачів зі схожим смаком та item-based, де РС рекомендує фільми схожі на ті, які вже сподобались користувачеві.

2. Контентно – орієнтовані системи. Аналізують характеристики фільмів: жанри, актори, режисер тощо, та надають рекомендації на основі тих фільмів, що вже оцінені користувачем. Вимагають структуровану мета-інформацію.

3. Системи на основі моделей з глибоким навчанням: створені на основі нейронних мереж, що здатні відслідковувати складні патерни взаємодій користувачів із фільмами.

4. РС на основі великих мовних моделей (Large Language Model, скорочено LLM): забезпечують високий рівень персоналізації завдяки натренованим трансформерам для розуміння описів, відгуків та навіть запитів.

5. Гібридні системи: поєднують у собі декілька методів, часто колаборативний та контентно – орієнтований.

Зазвичай великі сервіси, наприклад, Amazon та Netflix, обирають саме комбіновані, або гібридні рішення, адже вони адаптуються до даних.

1.4 Особливості рекомендаційних систем

Кожна платформа чи веб–сервіс у потребі РС має виконати свою ціль. Отримати не просто алгоритм автоматизованої реклами, а власного інтелектуального помічника для кожного клієнта, що допоможе користувачеві з його запитами до платформи. Різноманіття платформ та пропонованих продуктів чи послуг створює задачу необхідності попереднього дослідження та врахування всіх особливостей рекомендаційних систем.

Розберемо найважливіші особливості РС, які необхідно враховувати перед початком планування їхньої розробки та впровадження до онлайн–платформи чи сервісу. Які критерії необхідно враховувати:

1. Мета платформи – збільшення прибутку, залучення нових користувачів чи утримання існуючої аудиторії.
2. Дані – тип, формат, розмірність тощо.
3. Специфіка платформи з точки зору запиту користувачів – що саме очікують клієнти від сервісу.
4. Обмеження платформи – технічні, ресурсні чи законодавчі.
5. Швидкість оновлення – чи є дані динамічними, як часто необхідно вносити корегування.
6. Інтерфейс взаємодії – окрема гілка з обмеженнями (який функціонал, простір для рекомендацій тощо).

7. Можливість тестування.

Коли вищезазначені критерії були враховані, можна перейти до особливостей роботи алгоритму, а саме:

8. Персоналізація – необхідність адаптації РС під кожного користувача.

9. Залежність від даних – система потребує дані з якнайвищим показником: точності, достовірності, повноти, обсягу.

10. Масштабованість – можливість масштабувати РС з розвитком платформи чи веб–сервісу.

11. Проблема холодного старту – необхідність вирішення проблеми надання якісних рекомендацій новим користувачам.

12. Безпека та стійкість – захист від штучно створених даних чи зловмисним просуванням певного товару чи послуги третьою стороною.

Врахувавши всі ці особливості, можна перейти до планування та створення рекомендаційної системи.

1.5 Існуючі рекомендаційні системи

На сьогоднішній день РС є невід’ємною складовою багатьох сервісів, якими ми користуємось щодня. Інколи вони непомітні, інколи навпаки надто агресивні, але так чи інакше ми знаходимось з ними в одному інформаційному полі та щоразу з ними стикаємось, коли беремо до рук смартфон чи інший сучасний медіа–пристрій.

Для кращого візуального розуміння рекомендаційних систем розглянемо декілька сервісів та платформ з наявними РС.

1. Spotify – один з найбільших у світі музичних стрімінгових сервісів, де доступні мільйони пісень, подкастів та аудіо лекцій (рис. 1.1). Заснований

у 2008 році та має понад 500 млн користувачів [9]. Сервіс використовує гібридну рекомендаційну систему, що поєднує в собі фільтрацію на основі мета-даних (metadata-based filtering), колаборативну фільтрацію (collaborative filtering) та глибоке навчання через нейронні мережі (DNN).

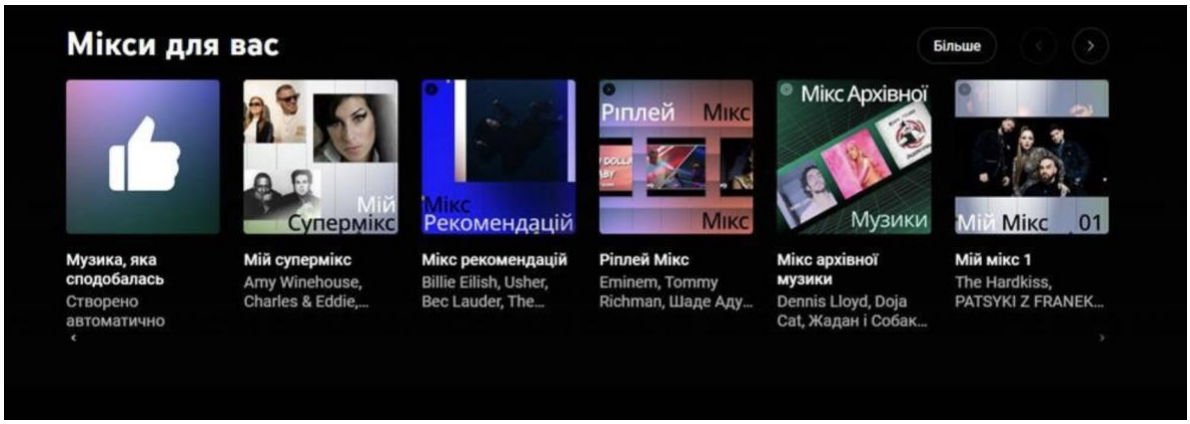

Рисунок 1.1 Рекомендаційна система сервісу Spotify [10].

2. YouTube – найпопулярніший у світі відеохостинг, заснований у 2005 році (рис. 1.2). Сервіс надає можливість користувачам дивитися, оцінювати, коментувати та викладати власні відеоматеріали, а також отримувати персоналізовані рекомендації для перегляду потенційно цікавих користувачеві відео. Має понад 2 млрд користувачів та охоплює всі тематики: від новин та навчання до розважальних та повсякденних відео [11]. Платформа має складну гібридну рекомендаційну систему, що складається з глибинних нейронних мереж (DNN), модулю кандидування (Candidate Generation), модулю ранжування (Ranking Model) та контекстно–обумовлених рекомендацій.

Рисунок 1.2 Рекомендаційна система сервісу YouTube.

3. Netflix – один з найбільших у світі провайдерів медійних послуг та продюсерська компанія, що надає доступ до великої бібліотеки фільмів, серіалів, документалістики тощо (рис. 1.3). Була заснована у 1997 році у Каліфорнії, США. Наразі сервіс має понад 230 млн підписників. Платформа має одну з найкращих РС, яка є гібридною та заснована на контентній фільтрації (content-based filtering), колаборативній фільтрації (collaborative filtering), глибокому навчанні та векторизації (Deep Learning & Embedding), системі ранжування (Ranking Model) і контекстно–адаптивній рекомендації. Netflix є проривною компанією, яка відзначилась також цікавою подією. У 2006 році Netflix оголосили Netflix Prize – відкритий конкурс задля покращення їхньої РС з призом в 1 млн доларів за

перевершення їхньої рекомендаційної системи на 10%. Це значно простимулювало розвиток машинного навчання, PC та, звісно, компанії.

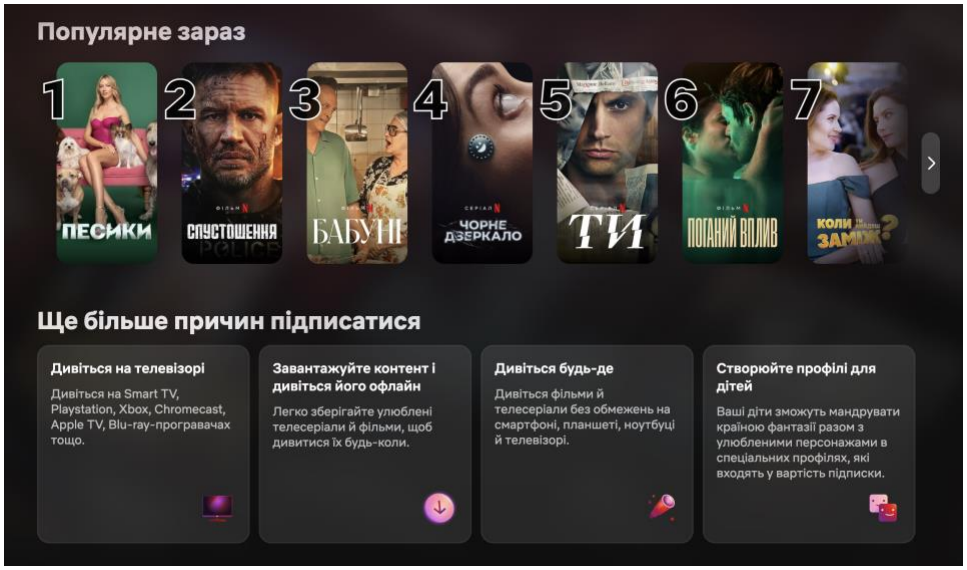

Рисунок 1.3 Рекомендаційна система сервісу Netflix [12].

4. Instagram – одна з найбільших соціальних мереж та платформ для обміну та публікацій фото та відео, що належить американській технологічній корпорації Meta Platforms, Inc (рис. 1.4). Платформа була запущена у 2010 році і наразі має понад 2 мільярди активних користувачів щомісяця. Платформа надає можливість як публікувати власні медіа-матеріали, так і слідкувати за контентом інших людей, а також спілкуватися всередині платформи.

Instagram використовує гібридну рекомендаційну систему, яка побудована на комбінації контентної фільтрації (content-based filtering), колаборативній фільтрації (collaborative filtering), глибокому навчанні (deep learning) та системі ранжування (Ranking Model). На рис. 1.4 продемонстровано PC Instagram за цільовим запитом.

Рисунок 1.4 Рекомендаційна система платформи Instagram. Робота з чітким запитом.

5. Rozetka – один з найбільших інтернет-магазинів України, засновано у 2005 році та налічує мільйони користувачів щомісяця (рис. 1.5). Оскільки платформа велика та пропонує різноманітні товари, то, звісно, має відмінну адаптивну РС, побудовану на гібридній моделі, що поєднує контентну фільтрацію (content-based filtering), колаборативну фільтрацію (collaborative filtering) та машинне навчання.

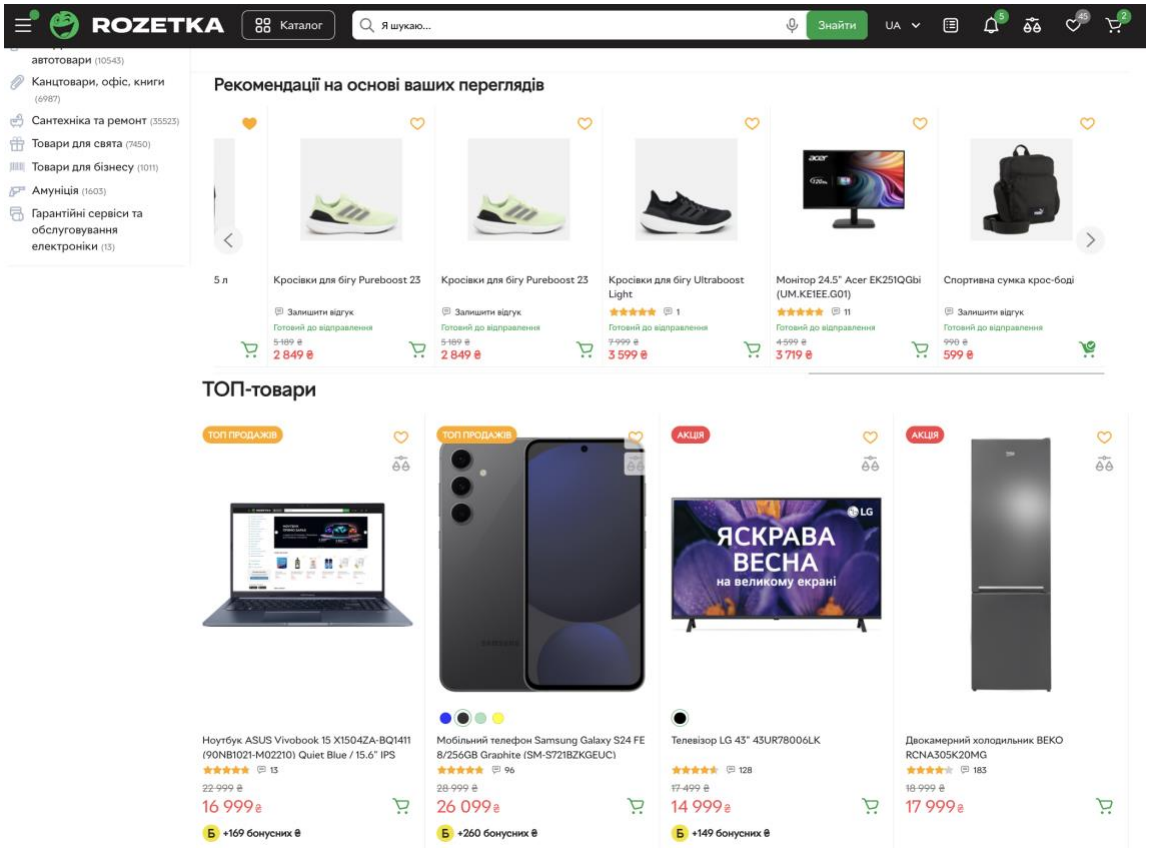

Рисунок 1.5 Рекомендаційна система маркетплейсу Rozetka

Дана РС є влучним прикладом поєднання як персоналізованих рекомендацій (у верхній частині), так і загальних рекомендацій на основі найбільш проданих товарів (у нижній частині рисунку).

Є ще надвелика кількість прикладів РС великих компаній: Amazon, Google, Facebook (Meta) тощо, але найяскравіші та найкорисніші приклади вже були наведені.

1.6 Висновки до розділу 1

Отже, в цьому розділі було розглянуто поняття рекомендаційна система, їхні види, функції, класифікація та практичне застосування, наведені візуальні приклади. А також вказані їхні відмінності та особливості.

Також наведено коротку історію створення і розвитку та проблематику рекомендаційних систем.

Можна зробити висновок, що рекомендаційні системи – це необхідність сьогодення, що допомагає як великим компаніям, так і користувачам із запитами один до одного. РС – не просто алгоритми, а цілі системи, націлені на результат та взаємодопомогу.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Вступ до основ побудови рекомендаційних систем

Процес побудови рекомендаційної системи вимагає комплексного підходу не тільки до алгоритму, а й загалом до системи. Існує надвелика кількість змінних, яку необхідно врахувати задля того, щоб розробити дійсно актуальний та , що найголовніше, ефективний продукт.

Адже РС – складна та індивідуальна система ще з етапу планування, бо кожна РС створюється під конкретний запит та платформу чи веб-сервіс.

Саме розуміння цього допомагає підійти до створення рекомендаційної системи так, щоб кінцевий продукт був достатньо результативним, актуальним та відповідав усім вимогам задачі.

У найсучасніших РС система працює з користувачем, що наявно чи неявно демонструє свої вподобання, а також показує альтернативи свого вибору. Системи це зчитують та враховують. У результаті система працює через певний метод фільтрації та усі зібрані дані про вподобання користувача, щоб зарекомендувати саме той набір варіантів, які він вважатиме найбільш доцільними.

Для побудови власної РС розглянемо детальніше існуючі алгоритми та методи фільтрації.

2.2 Детальний огляд основних методів фільтрації

2.2.1 Контентно–орієнтована фільтрація (content–based filtering)

Даний метод фільтрації використовує дані про активності та існуючі вподобання користувача та на основі них створює рекомендації (рис. 2.1). Наприклад, метод використовує інформацію про те, які фільми найбільше сподобались користувачеві, які блоги він найчастіше читає чи просто залишає коментарі (будь-яка напрямлена активність). Ця інформація у найбільш сучасних системах зчитується автоматично за допомогою альтернативних методів подібності [5].

У випадку рекомендаційної системи фільмів ми можемо виділити наступні дані, що враховуються при контентно–орієнтованій фільтрації: жанр, ключові слова (теги), опис фільму, головний актор, другорядні актори, рік випуску, режисер, тривалість, автор сценарію, країна походження, мова оригіналу, середня оцінка, кількість оцінок, бюджет.

Як можемо побачити, ці дані включають в себе мета–інформацію про фільм.

Метод ґрунтується на аналізі даних про фільм та індивідуальних вподобань користувача, а рекомендації створюються через порівняння нових об’єктів (за умови достатньої кількості даних) з тими, що вже були позитивно оцінені користувачем. Наприклад, якщо користувачеві подобаються усі фільми з актором А у головній ролі, то система рекомендуватиме йому нові фільми з цим актором. Через це існує проблема “холодного старту”, коли дані про вподобання користувача відсутні. Про це детальніше згодом.

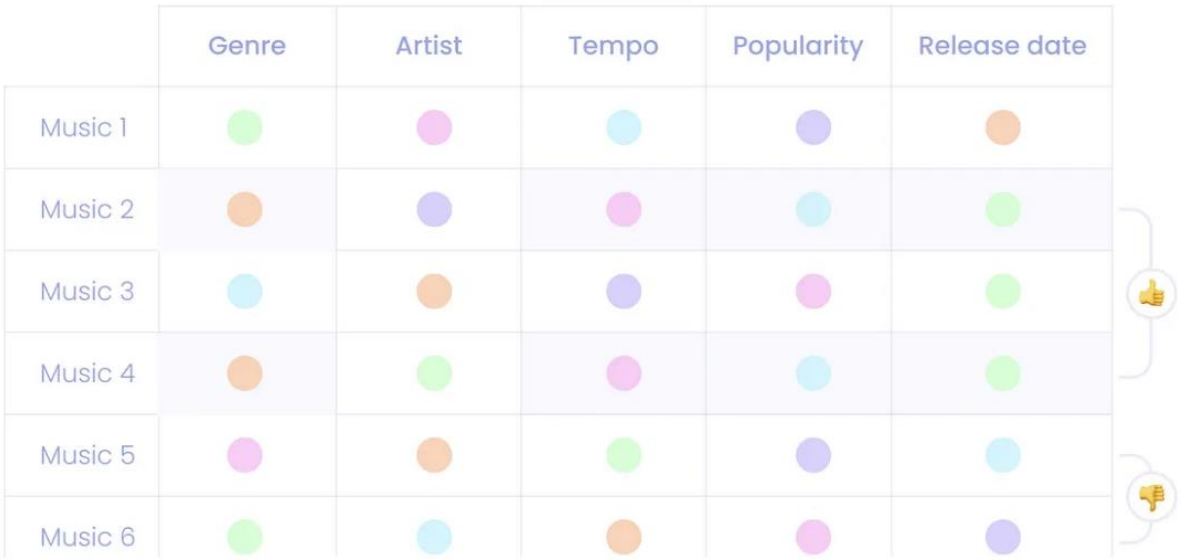

Рисунок 2.1 Приклад даних для контентно-орієнтованої фільтрації [13].

Далі розглянемо як такий вид фільтрації працює на прикладі коротких відеокліпів в соціальній мережі TikTok. На рис. 2.2 наведено матрицю взаємодій, де кожен рядок — це вектор ознак (embedding vector), який представляє окремий відеокліп, кожен з яких описується за допомогою характеристик (тривалість, категорія, автор, теги тощо). Горизонтальний вектор знизу відображає користувача, який також описується вищезгаданими характеристиками. Задля того, щоб визначити схожість між користувачем та кожним відеокліпом, використовується скалярний добуток цих векторів — кількість ознак, які наявні одночасно в обох векторах. Відповідно, чим більше спільних ознак, тим більшим буде скалярний добуток і, як результат, вищим рівень схожості.

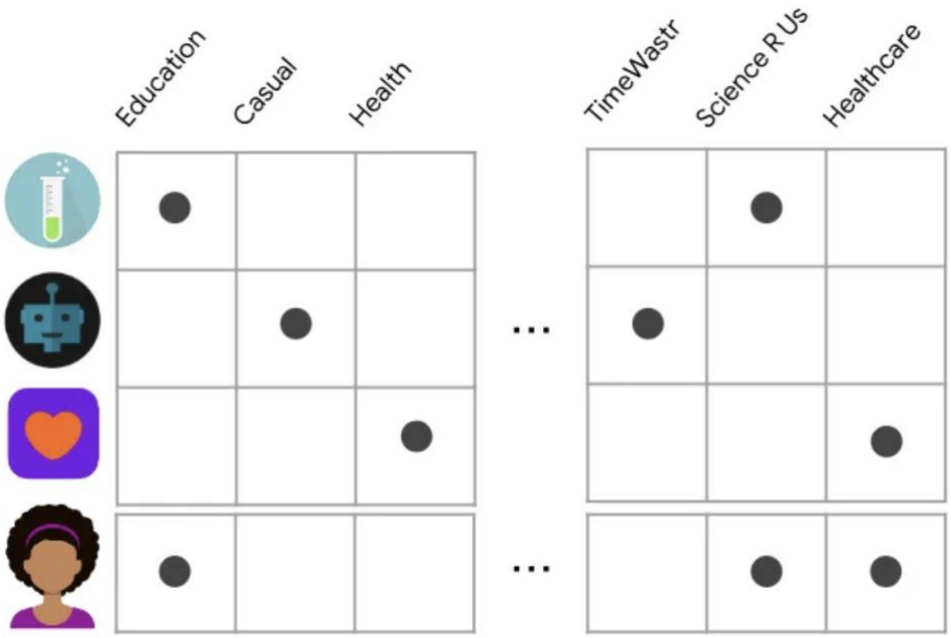

Рисунок 2.2 Матриця взаємодій користувача [14].

Поняття міри подібності вимірюється одним з трьох найпопулярніших методів: косинус, скалярний добуток (найпоширеніший), евклідова відстань.

Косинусна подібність – найпопулярніша з цих метрик. Вона складається з рядка клієнта та стовпця товару у геометричному вигляді. Подібність двох з них розраховується як косинус між їх відповідними векторами [6] за формулою (2.1):

$$cosine\ similarity\ (A, B) = \cos(A, B) = \frac{A \cdot B}{||A|| \cdot ||B||}, \quad (2.1)$$

де A і B — вектори інтересів користувача та ознак товару відповідно.

Також на рис 2.3 наведемо схему роботи контентно-орієнтованої фільтрації для складних систем

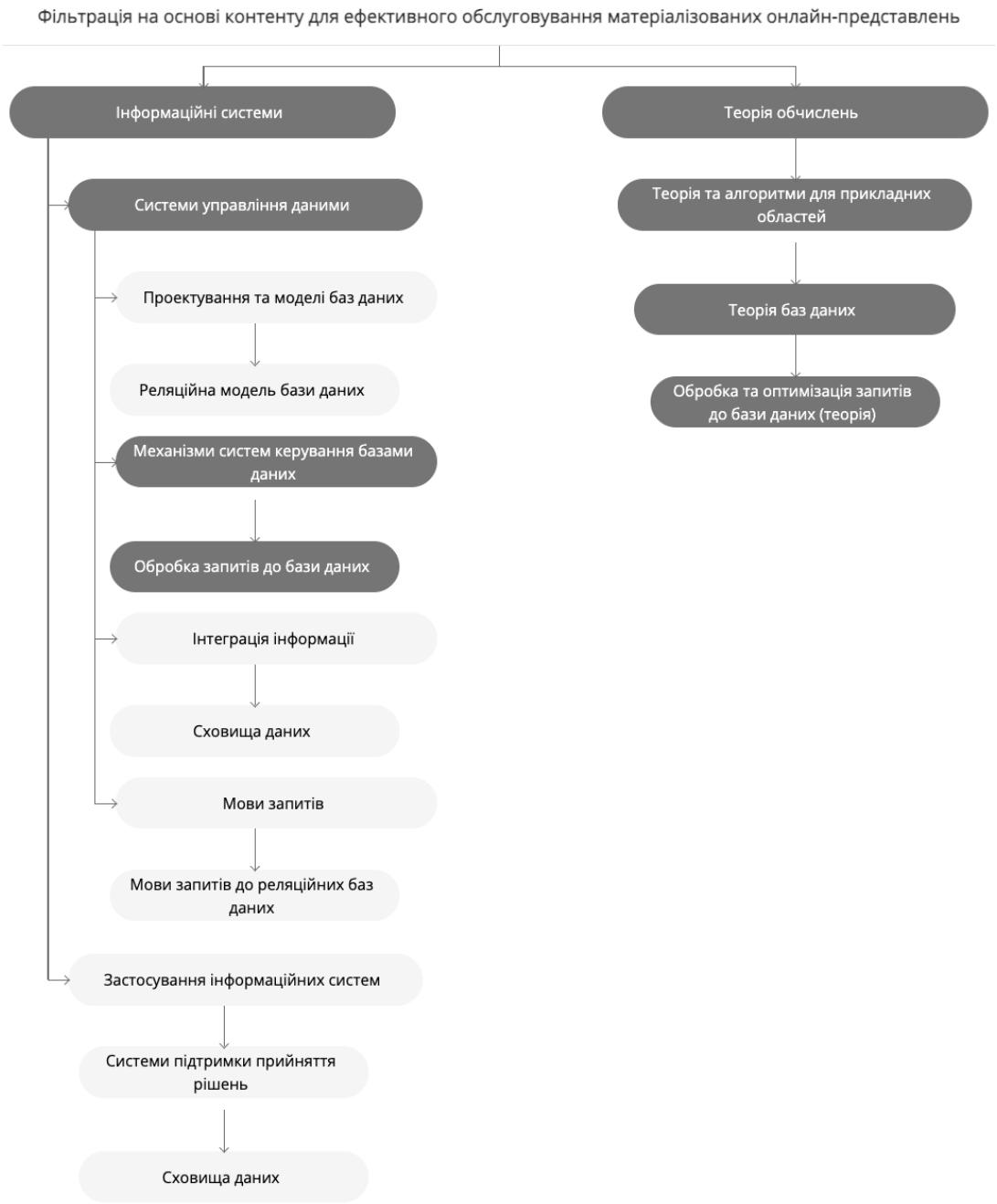

Рисунок 2.3 Схема роботи контентно-орієнтованої фільтрації для складних систем [15].

2.2.2 Колаборативна фільтрація (collaborative filtering)

Задля подолання обмежень методів, заснованих виключно на контентно-орієнтованій фільтрації, використовується метод колаборативної фільтрації. Його головна перевага — побудова рекомендацій з урахуванням подібностей як між самими користувачами, так і між об’єктами (у нашому випадку – фільмами) одночасно. Це дозволяє створювати неочевидні, але якісні рекомендації — наприклад, система може запропонувати користувачеві А фільм, яким цікавився користувач В, якщо їхні вподобання мають збіг.

До того ж, такий підхід дозволяє уникати необхідності ручного створення характеристик — модель здатна самостійно навчитися ефективним рекомендаціям з часом.

Відгуки в такій системі бувають двох основних типів:

1) явні — користувач ставить оцінку фільму;

2) неявні — система фіксує сам факт перегляду як індикатор зацікавленості або залучення.

Коли людина заходить на сторінку сервісу, РС формує персоналізовані рекомендації, спираючись на два головних фактори:

1) подібність до контенту, який вже сподобався цьому користувачеві;

2) уподобання інших користувачів зі схожими смаками.

Далі на рис. 2.4 наведемо наявне порівняння методів контентно-орієнтованої та колаборативної фільтрації.

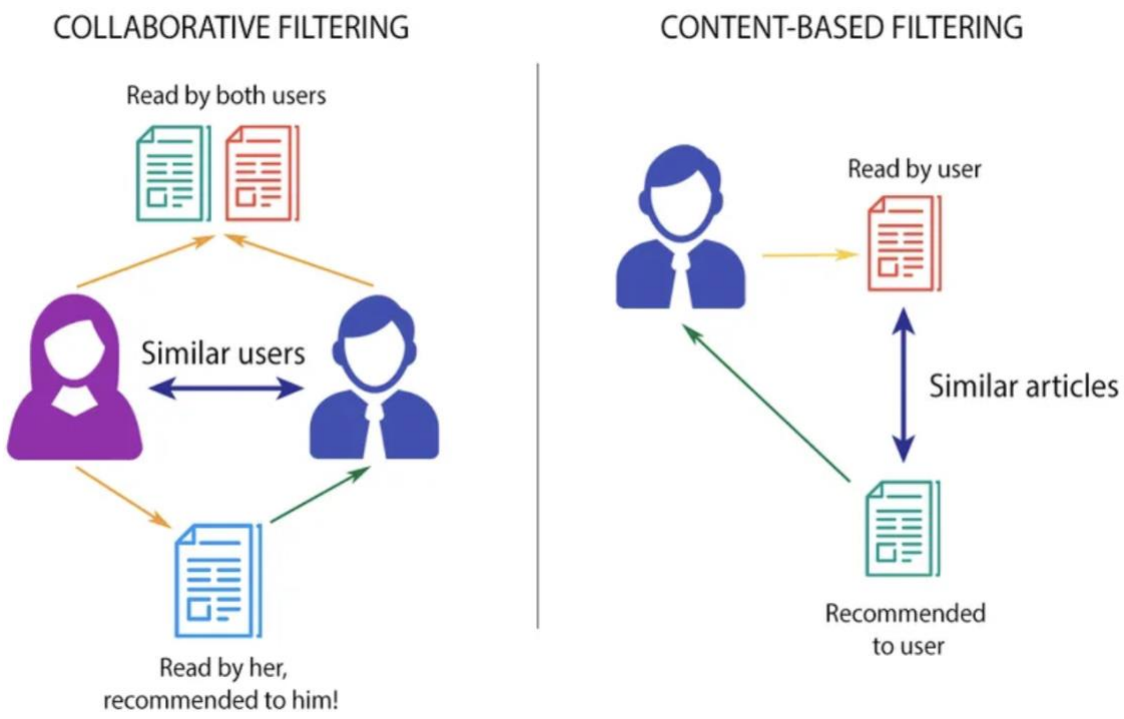

Рисунок 2.4 Наявне порівняння методів контентно-орієнтованої та колаборативної фільтрації [14].

Перш ніж переходити до аналізу алгоритмів, необхідно ввести базове поняття матриці взаємодії користувачів і елементів (user-item interaction matrix).

Нехай U – множина всіх користувачів системи ($U = \{u_1, u_2, u_2 \ldots, u_n\}$), а I – множина доступних елементів (фільмів) $I = \{i_1, i_2, \ldots, i_m\}$. Тоді $R_{(u \times i)}$ позначає множину оцінок, де $r_{(u \times i)}$ – це оцінка, яку користувач u залишив для фільму i.

Якщо уявити користувачів як рядки, фільми – як стовпці, а на перетині цих двох вимірів розмістити відповідні оцінки, то отримаємо ту саму матрицю взаємодії.

Варто підкреслити, що в реальних системах, де кількість користувачів і фільмів може сягати тисяч або навіть мільйонів, ця матриця, як правило, буде розрідженою – адже очевидно, що жоден користувач не здатен оцінити всі існуючі фільми. Через це використовувані датасети сягають надвеликих розмірів

і розробникам доводиться зменшувати область дослідження чи тестової вибірки, це розглянемо згодом.

Саме ця матриця взаємодії лежить в основі роботи алгоритмів колаборативної фільтрації, які й використовуються для побудови персоналізованих рекомендацій. У таблиці 2.1 наведемо простий приклад матриці взаємодій.

Таблиця 2.1 Приклад матриці взаємодій

	Фільм 1	Фільм 2	Фільм 3	Фільм 4
Користувач 1	-	-	-	2
Користувач 2	-	-	3	-
Користувач 3	-	-	-	-
Користувач 4	1	4	-	-
Користувач 5	-	2	-	-

Оскільки, як було сказано вище, матриця взаємодії, особливо в реальних умовах з великою кількістю користувачів і об’єктів, є надмірно розрідженою — більшість комірок є порожніми, тобто користувачі взаємодіють лише з малою частиною доступного їм контенту. Задля того, щоб виконати прогнозування щодо вподобань користувача по відношенню до невідомих об’єктів, необхідно відновити відсутні дані матриці взаємодій.

Одним з найбільш ефективних способів вирішення цієї проблеми є метод матричної факторизації — метод, що дозволяє представити велику матрицю взаємодій у вигляді добутку двох менших матриць, що відображають приховані характеристики користувачів та об’єктів. Це дає змогу не лише зменшити розмірність та ресурсне навантаження задачі, а і виявити глибинні закономірності в поведінці користувачів.

Матрична факторизація (MF) – один з найефективніших підходів до побудови великих РС. На практиці матрична факторизація показує високу ефективність у створенні моделей прихованих факторів (latent factor models). Основна ідея методу полягає в тому, щоб на основі шаблонів оцінювання фільмів побудувати вектори ознак, що характеризують як користувачів, так і самі елементи. Надалі рекомендації формуються через міру схожості між вищезгаданими векторами.

Розглянемо матрицю взаємодій $R_{u\times i}$ розміром u×i відповідно, де u — кількість користувачів, а і — кількість фільмів. Отже, матриця відображає наявні взаємодії між користувачами та фільмами. Її можна апроксимувати добутком двох матриць меншого розміру за формулою (2.2):

$$R_{u\times i} = P_{u\times k} \cdot Q^{T}_{i\times k}. \tag{2.2}$$

У моделі факторизації є гіперпараметр k, який визначає розмірність векторів ознак — ембедингів (embedding vector). Отримуємо, що кожен користувач і кожен елемент представлені у вигляді дійсного вектора довжини k.

Взаємодія між користувачем та елементом моделюється як скалярний добуток відповідних векторів. На рис. 2.5 та рис. 2.6 наведено наявний алгоритм матричної факторизації у графічному вигляді за формулою (2.2) [7].

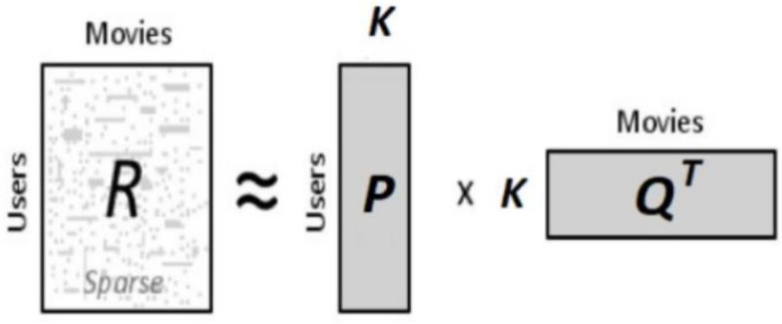

Рисунок 2.5 Алгоритм матричної факторизації у графічному вигляді [16].

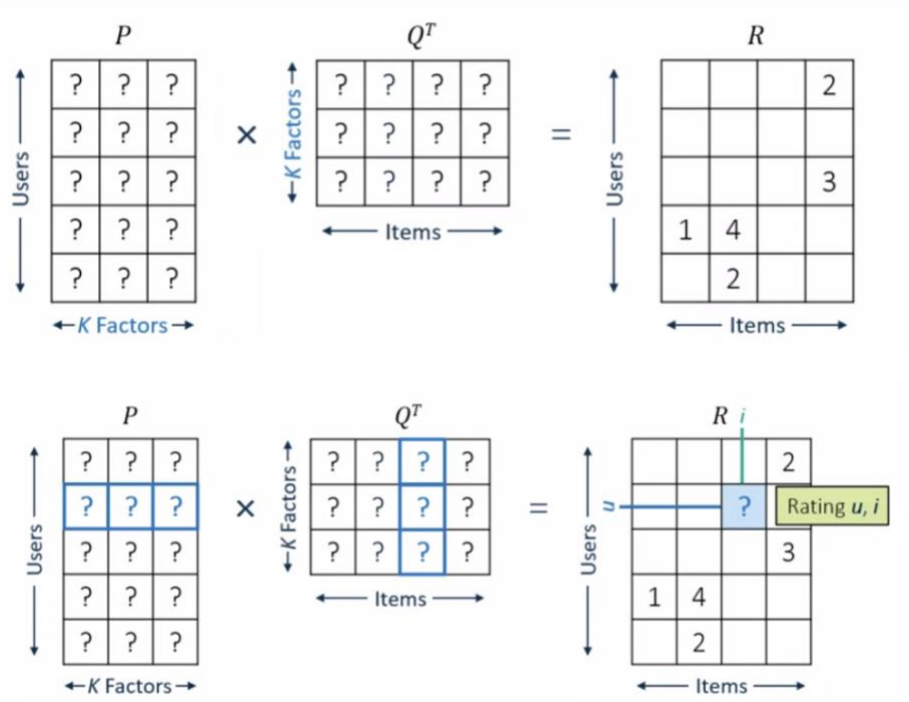

Рисунок 2.6 Приклад роботи алгоритму матричної факторизації [16].

Далі використовуємо формулу (2.3) прогнозування оцінки користувача u для фільму i:

$$\widehat{r_{ui}} = p_u \cdot q_i^T = \sum_{k=1}^{K} p_{uk} \cdot q_{ki} \tag{2.3}$$

Використовуємо формулу (2.4) для мінімізації похибок (різниці між реальною оцінкою r та прогнозованої $\hat{r}$) під час навчання моделі:

$$e_{ui}^2 = (r_{ui} - \widehat{r_{ui}})^2 = (r_{ui} - \sum_{k=1}^{K} p_{uk} \cdot q_{ki})^2 \tag{2.4}$$

Таким чином отримуємо, що після навчання матриця вподобань користувача до фільмів буде заповнена. Щоб мінімізувати помилку, продиференціюємо вищезгадане рівняння за змінними p_{uk} та q_{ki} за формулою (2.5):

$$\frac{\partial}{\partial p_{uk}} e_{ui}^2 = -2(r_{ui} - \widehat{r_{ui}}) q_{ki} = -2e_{ui} q_{ki},$$
$$\frac{\partial}{\partial q_{ki}} e_{ui}^2 = -2(r_{ui} - \widehat{r_{ui}}) p_{uk} = -2e_{ui} p_{uk}. \quad (2.5)$$

Завдяки вводу швидкості навчання α отримуємо змогу ітеративно оновлювати необхідні приховані фактори p_{uk} та q_{ki} за формулою (2.6):

$$\mathrm{p'_{uk}} = \mathrm{p_{uk}} + \alpha \frac{\partial \mathrm{e_{ui}^2}}{\partial \mathrm{p_{uk}}} = \mathrm{p_{uk}} + 2\alpha \mathrm{e_{ui} q_{ki}},$$
$$\mathrm{q'_{ki}} = \mathrm{q_{ki}} + \alpha \frac{\partial \mathrm{e_{ui}^2}}{\partial \mathrm{q_{ki}}} = \mathrm{q_{ki}} + 2\alpha \mathrm{e_{ui} p_{uk}}. \quad (2.6)$$

Датасети для РС мають вбудовані упередження через те, що різні користувачі по-різному оцінюють фільми або бувають більш вимогливими при виставленні оцінок, тому деякі елементи в середньому оцінюються вище, ніж інші. Саме тут з'являється потреба у модифікованій матричній факторизації з урахуванням упереджень. У рівнянні нище попередню модель розширено завдяки урахуванню упередженості, а також додано регуляризаційні члени, які забезпечують стабільність і коректність значень прихованих факторів.

$$\widehat{r_{ui}} = \mu + b_u + b_i + p_u \cdot q_i^T, \quad (2.7)$$

де μ, b_u, b_i — це середнє упередження, упередження користувача та упередження елемента відповідно.

Також розглянемо другий варіант з точки зору реалізації. Однією з головних проблем при побудові моделей матричної факторизації є запобігання перенавчанню. Дана ситуація виникає, коли модель чрезмірно підлаштовується під дані навчальної вибірки, у тому числі і під шум, і через це демонструє погану узагальнюючу здатність на нових даних. Задля уникнення цього, у функцію втрат

додаються регуляризаційні члени, що "штрафують" модель за надмірно великі значення прихованих факторів [7].

Функція помилки (втрат) у цьому випадку рахується за формулою (2.8):

$$J = \sum_{(u,i)\in\kappa}(r_{ui} - \widehat{r_{ui}})^2 \quad (2.8)$$

де к — множина пар (користувач – фільм), для яких відомі оцінки.

Як було сказано вище, тут додаються регуляризаційні члени, які штрафують великі значенн векторів P та Q. Це запобігає надмірному збільшенню ваг, що може призводити до поганої узагальненості моделі.

Регуляризована функція втрат рахується за формулою (2.9):

$$J = \sum_{(u,i)\in\kappa}(r_{ui} - P_u \cdot Q_i^{\top})^2 + \lambda(||P_u||^2 + ||Q_i||^2) \quad (2.9)$$

де $\lambda > 0$ — коефіцієнт регуляризації, $||P_{u|}|^2 = \sum_{k=1}^{K} p_{uk}^2$ — квадратична норма вектора користувача, $||Q_i^{\top}||^2 = \sum_{k=1}^{K} q_{ik}^2$ — квадратична норма вектора фільму.

Який результат регуляризації отримуємо:

1) коли λ занадто мале, тоді модель перенавчається (враховує зайві шуми даних);
2) коли λ занадто велике, тоді модель недонавчається (недостатньо адаптується).

Параметр λ добирається за допомогою крос-валідації (методу оцінювання моделі на різних підмножинах даних, що дозволяє ефективно обирати гіперпараметри).

Отже, регуляризація – важливий метод при побудові моделей матричної факторизації. Вона дозволяє моделі залишатися стійкою до шуму в даних, запобігає перенавчанню та забезпечує кращий результат на нових даних, а вибір параметра регуляризації λ суттєво впливає на якість рекомендацій.

Розглянемо обробку неявних дій. У більшості моделей колаборативної фільтрації використовується лише явна інформація, тобто оцінки фільмів користувачами. Проте при реальних сценаріях використання таких РС часто часто виникає проблема нестачі явних даних або вони є неповними. Саме через це доцільно враховувати неявні відгуки — дані про взаємодію користувача з фільмом без прямої оцінки. Розглянемо приклади таких взаємодій:

1. Факт повного перегляду фільму.
2. Поставлене вподобання фільму (без оцінки) або додавання у список обраних.
3. Тривалість перегляду.

Для обробки неявної взаємодії використовується неявний метод чергування найменших квадратів (iALS), адаптований для роботи з бінарними або зваженими сигналами взаємодії. У цій моделі кожна взаємодія користувача $r_{ui} \in \{0,1\}$ перетворюється на позитивний сигнал з вагою: $c_{ui}=1+\alpha \cdot n_{ui}$, де c_{ui} – ступінь довіри до взаємодії користувача u з об'єктом i, α – гіперпараметр, n_{ui} — бінарне значення наявності взаємодії (1 — є взаємодія, 0 — немає).

Далі апроксимувати матрицю взаємодії користувачів з об'єктами за допомогою двох низьковимірних матриць: матриці латентних факторів користувачів та матриці латентних факторів фільмів.

Завдання моделі полягає в мінімізації наступної цільової функції за формулою (2.10):

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \sum_{u,i} c_{ui}\left(n_{ui} - P_u\, Q_i^T\right)^2 + \lambda(||P_{\boldsymbol{u}}||^2 + ||Q_{\boldsymbol{i}}||^2), \qquad (2.10)$$

де c_{ui} — ступінь довіри до взаємодії користувача u з фільмом i, n_{ui} — бінарне значення наявності взаємодії (1 — є взаємодія, 0 — немає); λ — коефіцієнт регуляризації. Модель працює так, щоб скалярний добуток $P_u\,, Q_i^T$ якомога

точніше відображав ймовірність неявної взаємодії між користувачем і фільмом, з урахуванням впевненості у даній взаємодії [6].

2.2.3 Гібридні моделі

Контентно–орієнтована та колаборативна фільтрації мають як переваги, так і недоліки. Для мінімізації недоліків та максимізації переваг варто використовувати саме гібридні моделі, в яких враховуються слабкі та сильні сторони обох методів. Гібридні моделі поєднують обидва методи, щоб компенсувати недоліки кожного з них. Такі системи враховують як інформацію про взаємодії: кліки, перегляди тощо, так і мета-інформацію про фільм: жанри, акторів, ключові слова, опис сюжету тощо. Існує декілька стратегій побудови гібридних моделей. Розглянемо дві найпоширеніші з них.

1. Model-based. Побудова єдиної моделі, яка одночасно враховує як колаборативну інформацію, так і контентні ознаки. Реалізується завдяки тому, що вектори фільмів формуються не лише на основі взаємодій, але й на основі їхніх мета-даних.

2. Weighted. Це комбінація результатів кількох окремих моделей шляхом зваженого сумування через ваговий коефіцієнт. Наприклад, якщо r_{ui}^{CF} — прогноз на основі колаборативної фільтрації, а $\widehat{r_{ui}^{CB}}$ — прогноз на основі content-based фільтрації, то остаточний результат буде обчислюватись за формулою (2.11):

$$\widehat{r_{ui}} = \alpha \cdot \widehat{r_{ui}^{CF}} + (1 - \alpha) \cdot \widehat{r_{ui}^{CB}}, \tag{2.11}$$

де $\alpha \in [0,1]$— ваговий коефіцієнт, що визначає важливість та відповідно вагу кожної складової. Можливість вибору α надає змогу регулювати баланс між

двома підходами. Наприклад, при α = 0.7, тобто 70% довіри (ваги) надається колаборативній частині, а 30% — контентній.

Контентні ознаки фільмів можна подати у вигляді багатовимірного вектору ознак. Для цього можна застосувати TF-IDF для текстових описів та one-hot-encoding для категоріальних змінних. Розберемо це детальніше.

TF-IDF — статистична міра, що оцінює важливість терміну в документі (у нашому випадку – фільмі) відносно всієї колекції документів. Широко застосовується для обробки текстів (наприклад, описів фільмів, оглядів, ключових слів), особливо при побудові контентно-орієнтованих рекомендацій. TF-IDF для терміну t у документі d визначається як добуток двох компонентів за формулою (2.12):

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t) \tag{2.12}$$

де $\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$ — частота терміна t у документі d,

$\text{IDF}(t) = \log\left(\frac{N}{1+n_t}\right)$, де N— загальна кількість документів, n_t— кількість документів, що містять термін t.

Серед важливих тез щодо TD-IDF можна виділити декілька наступних.

1. Враховує частоту терміна в документі та його унікальність.

2. Допомагає виділяти саме релевантні слова.

3. Проте векторна модель має розмірність, що відповідає словнику, часто тисячі слів.

4. Не враховує порядок слів або контекст (але це може бути вирішено нейронними мережами).

One-hot encoding — метод представлення категоріальних ознак у вигляді бінарних векторів. Необхідно зазначити, що кожне можливе значення категоріальної змінної представляється окремим елементом вектора.

Нехай маємо множину унікальних значень категорії $C = \{c_1, c_2, \dots, c_k\}$. Тоді кожне значення c_i кодується вектором $\boldsymbol{x}^{(i)} = \left[0,0, \dots, \underbrace{1}_{i\text{-та позиція}}, \dots, 0\right]$, де всі елементи дорівнюють нулю, крім і-го, який дорівнює 1.

Серед важливих тез щодо one-hot encoding можна виділити:

1. Відносна простота реалізації.

2. Добре працює для невеликої кількості унікальних значень.

3. Висока розмірність: при великій кількості можливих значень ознаки (наприклад, тисячі акторів або ключових слів) вектор стає надрозрідженим.

4. Відсутність семантичної близькості: «драма» і «мелодрама» будуть такими ж далекими один від одного як «драма» і «жахи», бо кодування не враховує подібності [8].

2.3 Метріки оцінки якості рекомендацій

В побудові РС необхідно впроваджувати метрики, адже оцінка ефективності системи є критично важливою для аналізу її придатності у реальному застосуванні. Залежно від задачі використовуються різні метрики: для рейтинг-прогнозування (коли модель передбачає оцінки користувача) та для top-N рекомендацій (коли система повертає список із N релевантних фільмів).

Розглянемо декілька метрик [17].

1. RMSE (Root Mean Squared Error), що оцінює відхилення передбачених рейтингів від фактичних за формулою (2.13):

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|}\sum_{(u,i)\in\mathcal{T}}(r_{ui} - \widehat{r_{ui}})^2}\,, \tag{2.13}$$

де $|\mathcal{T}|$ — множина пар користувач–фільм у тестовій вибірці, r_{ui} реальна оцінка, $\widehat{r_{ui}}$— передбачена оцінка.

2. Середня абсолютна похибка (MAE), що вимірює середнє абсолютне відхилення між передбаченими і реальними оцінками рахується за формулою (2.14). Варто додати, що є менш чутливою до аномальних даних (викидів):

$$\text{MAE} = \frac{1}{|\mathcal{T}|}\sum_{(u,i)\in\mathcal{T}}|r_{ui} - \widehat{r_{ui}}| \tag{2.14}$$

де $|\mathcal{T}|$ — множина пар користувач–фільм у тестовій вибірці, r_{ui} реальна оцінка, $\widehat{r_{ui}}$— передбачена оцінка.

3. Повнота (Recall@K), що вимірює, яку частину всіх релевантних фільмів було в результаті рекомендовано і рахується за формулою (2.15):

$$\text{Recall@K} = \frac{\text{релевантні порекомендовані}}{\text{усі релевантні}}. \tag{2.15}$$

4. Точність (Precision@K), що вимірює частку релевантних фільмів серед top-N рекомендованих і рахується за формулою (2.16):

$$\text{Precision@K} = \frac{\text{релеватні}}{top-N}. \tag{2.16}$$

2.4 Висновки до розділу 2

Отже, в цьому розділі були детально розглянуті основні методи фільтрації: контентно-орієнтований, колаборативний, а також гібридні моделі.

Крім цього, було наведені графічні представлення та приклади, а також розглянуто метрики для оцінки результативності роботи рекомендаційних систем.

Можна зробити висновок, що рекомендаційні системи – це сучасні системи, в основі яких закладено математичний апарат та алгоритми роботи з даними.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

3.1 Початок роботи

Для реалізації програми було обрану мову програмування Python та середовище VSCode, оскільки таким чином ми отримуємо оптимізовану роботу з матрицями, що використовуються в алгоритмах роботи.

У практичній частині вирішено порівняти результати та оцінки роботи алгоритмів, що будуть автоматично виведені програмою. У якості дата-сету був обраний дата-сет MovieLens від дослідницької лабораторії GroupLens. Для тестування алгоритму використано менший дата-сет ml-latest-small через технічні причини, а саме обмеження обсягом пам’яті при завантаженні дата-сету та обчислювальною потужністю комп’ютера.

3.2 Аналіз даних

Після підключення необхідних бібліотек було завантажено дата-сет з офіційного сайту MovieLens у форматі csv.

У файлі movies.csv зберігається інформація про фільми: поле movieId відповідає за унікальний ідентифікатор фільму, title — назву, genres — за перелік жанрів, розділених символом «|». Загально у файлі представлено дані щодо 9742 фільмів.

Файл ratings.csv містить інформацію про оцінювання фільмів користувачами: userId — унікальний ідентифікатор користувача, movieId — ідентифікатор фільму, rating — виставлена оцінка, timestamp — час оцінювання (рис. 3.1). Загальна кількість оцінок – 100836.

```
movies
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9742 non-null   int64
 1   title    9742 non-null   object
 2   genres   9742 non-null   object
dtypes: int64(1), object(2)
memory usage: 228.5+ KB
Дубльованих записів: 0
------------
ratings
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100836 non-null  int64
 1   movieId    100836 non-null  int64
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
...
dtypes: int64(3), object(1)
memory usage: 115.2+ KB
Дубльованих записів: 0
------------
```

Рисунок 3.1 Результат аналізу даних

З інформації з вищенаведених файлів після попередньої обробки було сформовано матрицю взаємодії користувачів з фільмами. Отримана матриця є досить розрідженою. Це зумовлено тим, що для дослідження було використано не повний обсяг даних, а лише частину, і, як зазначалося раніше, розрідженість матриці обумовлена великою кількістю користувачів та фільмів.

3.3 Вибір моделей та результати

Було прийнято рішення розробити та порівняти результати моделей, заснованих на:

1) контентно-орієнтованій фільтрації (content-based з використанням TF-IDF та NearestNeighbors);
2) колаборативній фільтрації (модель SVD з бібліотеки surprise);
3) гібридному методі з двох попередніх.

Відповідно було отримано наступні результати (рис. 3.2-3.4):

```
За жанрами "Drama Crime Fantasy Romance" рекомендовано такі фільми:
------------------------------------------------------------------
Silence of the Lambs, The (1991)
Dances with Wolves (1990)
Hudsucker Proxy, The (1994)
Dave (1993)
Birdcage, The (1996)
Pretty Woman (1990)
Interview with the Vampire: The Vampire Chronicles (1994)
Batman (1989)
House Arrest (1996)
Ransom (1996)
```

Рисунок 3.2 Результати роботи моделі 1.

```
Для користувача з ID 13 рекомендовано (SVD):
--------------------------------------------
Brazil (1985) 4.81
Wild Bunch, The (1969) 4.80
The Artist (2011) 4.80
Day of the Doctor, The (2013) 4.79
American Beauty (1999) 4.78
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964) 4.78
Paths of Glory (1957) 4.75
Lawrence of Arabia (1962) 4.75
Black Mirror: White Christmas (2014) 4.74
Sophie Scholl: The Final Days (Sophie Scholl – Die letzten Tage) (2005) 4.74
```

Рисунок 3.3 Результати роботи моделі 2.

```
Для користувача з ID 13 рекомендовано (Hybrid):
------------------------------------------------
Rosemary's Baby (1968) 4.50
Ginger Snaps (2000) 4.33
Split (2017) 4.25
Misery (1990) 4.23
What Ever Happened to Baby Jane? (1962) 4.20
Babadook, The (2014) 4.10
[REC] (2007) 4.10
Thesis (Tesis) (1996) 3.92
Spiral (2007) 3.92
Dead Ringers (1988) 3.89
```

Рисунок 3.4 Результати роботи моделі 3.

У контентно-орієнтованому методі РС формує результати на основі подібності між фільмами, використовуючи жанри. Оскільки така модель не враховує оцінки користувачів, її основна мета полягає не в передбаченні числових рейтингів, а у виявленні фільмів зі схожими жанрами. Через це застосування класичних метрик регресії (RMSE, MAE) або класифікації (Precision@k, Recall@k), які передбачають наявність цільових значень, не є релевантним для оцінки якості представленної контентної моделі. Наприклад, у випадках, коли система рекомендує фільми, які ще не були оцінені користувачем, метрики точності автоматично занижуються, хоча самі рекомендації мають цінність.

Таким чином, контентна модель у даній роботі використовується як доповнюючий елемент гібридної системи, головна ціль якого – покращення рекомендацій та врахування ознак, недоступних у колаборативних методах, а також незалежна система підбору фільмів за заданими жанрами. Її ефективність оцінюється шляхом аналізу поліпшення метрик у гібридній моделі.

Для аналізу якості рекомендацій колаборативного та гібридного методів були використані метрики регресії – RMSE та MAE, розглянуті у розділі 2.

Для моделі 2, що реалізовує колаборативний метод було отримано (рис. 3.5):

```
RMSE: 0.8519
MAE:  0.6500
```

Рисунок 3.5 Метрики моделі 2.

Результат свідчить про те, що модель добре відтворює вподобання користувачів і формує персоналізовані передбачення з незначним відхиленням.

Для гібридної моделі 3 було отримано (рис. 3.6):

```
Hybrid Model RMSE: 0.8537
Hybrid Model MAE:  0.6556
```

Рисунок 3.6 Метрики моделі 3.

Незначне збільшення похибки зумовлене додаванням контентної моделі, яка не відображає індивідуальні вподобання користувачів. Однак цей підхід дозволяє моделі стійкіше працювати в умовах "холодного старту", що був детальніше розглянутий у розділі 1.

Отже, навіть враховуючи незначне зниження точності, РС, заснована на гібридній моделі, є доцільною з практичної точки зору завдяки розширенню можливостей системи у випадках, коли колаборативний підхід обмежений або недостатньо інформативний (холодний старт).

Підсумовуючи, отримані результати є цілком задовільними, враховуючи обмеження використаного набору даних. Датасет MovieLens “ml-latest-small” має невисоку щільність та невелику кількість користувачів та фільмів, що не повністю розкриває потенціал моделі. Водночас отримані значення RMSE та MAE свідчать про гарну та стабільну роботу системи навіть у таких умовах.

Гібридна модель продемонструвала високу гнучкість та розширюваність, а отже, її ефективність може суттєво зрости за умови застосування більших датасетів (наприклад, MovieLens 1M або 25M). Це відкриває перспективи подальшого вдосконалення системи та можливого впровадження її у реальні кіносервіси.

3.4 Висновки до розділу 3

Було розроблено повноцінну гібридну РС фільмів, яка поєднує методи колаборативної та контентної фільтрації. За основу було взято відкритий датасет MovieLens “ml-latest-small”.

Було реалізовано та порівняно наступні моделі.

1. Контентна модель на основі TF-IDF векторизації жанрів.
2. Колаборативна модель з використанням алгоритму SVD.
3. Гібридна модель, яка поєднує прогнозовані рейтинги з обох підходів із заданими ваговими коефіцієнтами.

Для оцінки якості рекомендацій використано метрики RMSE та MAE, згідно з якими модель SVD показала найкращу точність (RMSE = 0.8519, MAE = 0.6500). Гібридна модель мала трохи нижчі показники (RMSE = 0.8537, MAE = 0.6556), що пояснюється меншою інформативністю жанрових даних у порівнянні з латентними ознаками SVD. Проте, завдяки додаванню контентної складової, гібридна модель краще справляється зі сценаріями холодного старту та забезпечує кращу різноманітність рекомендацій.

Особливістю розробленої системи є її масштабованість, інтерпретованість та можливість подальшого вдосконалення. У майбутньому модель можна адаптувати до більших датасетів (наприклад, MovieLens 1M або 25M),

інтегрувати зображення, описи фільмів або використовувати глибоке навчання для обробки даних.

Отримані результати доводять доцільність використання гібридного підходу в сучасних РС та демонструють розуміння як теоретичних основ, так і практичної реалізації сучасних інтелектуальних систем.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Даний розділ присвячений дослідженню функціонально-вартісного аналізу програмного продукту, створеного для формування персоналізованих рекомендацій фільмів на основі моделей колаборативної та контентної фільтрації та їх гібриду. Даний підхід дозволяє комплексно оцінити ефективність реалізованих технічних рішень, зокрема з точки зору функціональності, якості, витрат на розробку та доцільності використання в практичних задачах.

Метою функціонально-вартісного аналізу є виявлення найбільш раціонального способу реалізації програмного продукту з урахуванням співвідношенням між функціональною цінністю кожної складової та втратами на її реалізацію.

Процедура функціонально-вартісного аналізу передбачає визначення ключових етапів створення програмного продукту, розрахунок витратних статей, оцінку технічних параметрів і подальший аналіз економічної ефективності впровадження. На основі отриманих формується обґрунтований висновок щодо найбільш доцільного варіанта реалізації.

4.1 Постановка задачі проектування

У роботі буде використано метод функціонально-вартісного аналізу для проведення техніко-економічного аналізу розробки програмного продукту, призначеного для формування персоналізованих рекомендацій фільмів на основі

моделей колаборативної та контентної фільтрації та їх гібриду. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Маємо наступні технічні вимоги до програмного продукту:

1) програмний продукт має функціонувати на персональних комп'ютерах із стандартним набором компонентів;
2) зрозумілість та зручність для користувача;
3) програмний продукт має забезпечити швидку обробку даних та доступ до інформації в реальному часі;
4) можливість зручного масштабування та обслуговування;
5) мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – формування персоналізованих рекомендацій фільмів на основі моделей колаборативної та контентної фільтрації та їх гібриду. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування;

F_2 – вибір бібліотеки для фільтрації;

F_3 – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

А) Python;

Б) C++.

Функція F_2:

А) Surprise (SVD);

Б) LightFM.

Функція F_3:

А) Jupyter Notebook;

Б) Google Colab.

Наведемо морфологічну карту системи, де наведені варіанти реалізацій основних функцій на рис. 4.1.

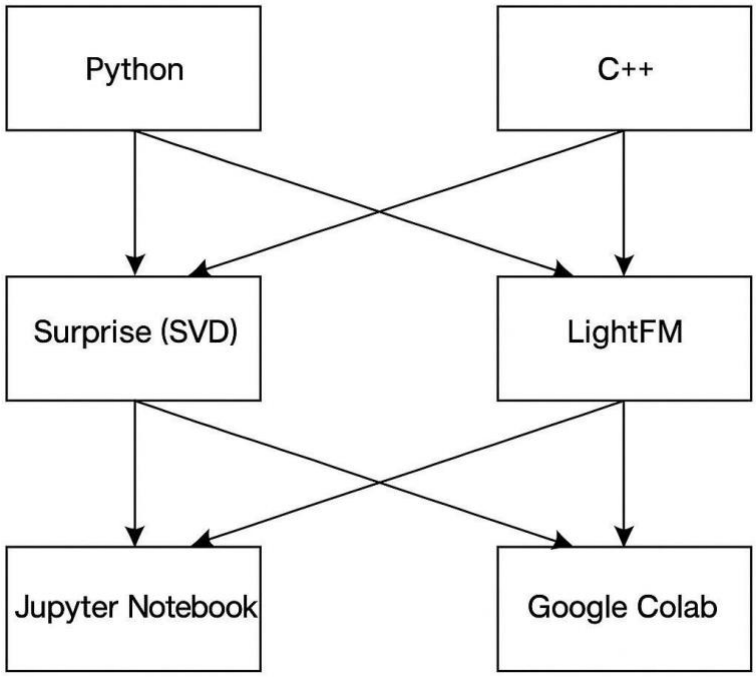

Рисунок 4.1. – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1. – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Зрозумілий синтаксис, наявність моделей для зручної та швидкої побудови програми.	Менший контроль за пам’яттю та низька швидкість виконання коду.
	Б	Висока продуктивність та контроль над ресурсами.	Складний синтаксис, довший цикл розробки.
F_2	А	Зручна інтеграція засобів розробки та наявні інструменти.	Складність реалізації моделей та менш інтуїтивний синтаксис.
	Б	Зрозумілий синтаксис та гнучкість у створенні власних моделей.	Більший обсяг пам’яті під час навчання.
F_3	А	Зручність у виведенні графіків.	Необхідність у налаштуванні середовища.
	Б	Простота у використанні.	Обмеженний час сесії.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 : надаємо більш вагому перевагу використанню мови програмування Python, через велику кількість моделей, які допомагають у побудові рекомендаційних систем. Враховуючи це, варіант Б) можна не розглядати

Функція F_2 : проаналізувавши переваги та недоліки кожного варіанту, можна сказати, що обидва варіанти є доцільними у використанні. Тому, варіанти реалізації А) та Б) слід порівняти

Функція F_3 : враховуючи зручність у виведенні графіків варіанту А та обмеження часу сесії варіанту Б, можна сказати, що перший варіант є доцільнишим за другий. Таким чином, варіант Б не розглядається

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$F_1\text{А} - F_2\text{А} - F_3\text{А}$

$F_1\text{А} - F_2\text{Б} - F_3\text{А}$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

Визначимо основні параметри вибору, які будуть використанні для розрахунку коефіцієнта технічного рівня на основі даних, які було розглянуто вище.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

1. X1 — швидкодія мови програмування.

2. X2 — час навчання моделі.

3. X3 — обсяг виористаної оперативної пам’яті.

4. X4 — загальний обсяг програмного коду.

Гірші, середні і кращі значення параметрів обираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметру	Умовні позначення	Одиниці виміру	Значення параметру		
			Гірші	Середні	Кращі
швидкодія мови програмування	X1	Оп/мс	25	15	5
час навчання моделі	X2	хв	40	20	10
обсяг використаної оперативної пам’яті	X3	Мб	3000	1500	500
загальний обсяг програмного коду	X4	Кількість строк	5000	2500	1000

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конктретної поставленої цілі — розробка

програмного продукту для формування персоналізованих рекомендацій фільмів на основі моделей колаборативної та контентної фільтрації та їх гібриду.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначені коефіцієнти в значимості передбачає:

1) визначення рівня значимості параметра шляхом присвоєння різних рангів;
2) перевірку придатності експертних оцінок для подальшого використання;
3) визначення оцінки попарного пріоритету параметрів;
4) обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметру	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	2	2	1	1	2	1	10	8,5	72,25

Продовження таблиці 4.3

X2	Час навчання моделі	хв	2	1	1	3	2	1	2	12	4,5	20,25
X3	Обсяг виористаної оперативної пам’яті	Мб	4	4	3	2	3	4	3	23	-5,5	30,25
X4	Загальний обсяг програмного коду	Кіль кість стро к	3	3	4	4	4	3	4	25	-7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	179

Для перевірки ступені достовірності експертних оцінок, визначимо наступні параметри:

1) сума рангів кожного з параметрів і загальна сума за формулою (4.1):

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів, n – кількість параметрів;

2) середня сума рангів за формулою (4.2):

$$T = \frac{1}{n} R_{ij} = 17{,}5; \quad (4.2)$$

3) відхилення суми рангів кожного параметра від середньої суми рангів за формулою (4.3):

$$\Delta_i = R_i - T, \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

4) загальна сума квадратів відхилення за формулою (4.4):

$$S = \sum_{i=1}^{N} \Delta_i^2 = 179. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості за формулою (4.5):

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 179}{7^2(4^3 - 4)} = 0{,}73 > W_k = 0{,}67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	<	<	>	>	<	>	1,5
X1 і X4	>	<	<	<	<	>	<	<	0,5
X2 і X3	<	<	>	<	<	>	<	<	0,5
X2 і X4	<	<	<	>	<	<	<	<	0,5
X3 і X4	<	<	>	<	<	<	>	<	0,5

Числове значення, що визначає ступінь переваги і–го параметра над j–тим, a_{ij} визначається за формулою (4.6).

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1{,}0 \text{ при } X_i = X_j, \\ 0{,}5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості K_{BI} за наступними формулами:

$$K_{\text{Bi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \tag{4.7}$$

$$b_i = \sum_{i=1}^{N} a_{ij}\,, \tag{4.8}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх. На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{Bi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j\,, \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	K_{Bi}	b_i^1	K_{Bi}^1	b_i^2	K_{Bi}^2
X1	1	1,5	1,5	0,5	4,5	0,28	16,25	0,27	59,125	0,27
X2	0,5	1	0,5	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X3	0,5	1,5	1	0,5	3,5	0,21	12,25	0,21	44,875	0,21
X4	1,5	1,5	1,5	1	5,5	0,35	21,25	0,36	77,875	0.36
Всього:					16	1	59	1	216	1

1.5 Аналіз рівня якості варіантів реалізації функцій

Рівень якості кожного варіанту основних функцій будемо визначати окремо.

Абсолютні значення параметрів X2 (час навчання моделі), X3 (обсяг використаної оперативної пам’яті), X4 (загальний обсяг програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Відмитимо, що абсолютне значення параметра X1 (швидкодія мови програмування) обрано не найгіршим.

Для кожного варіанта реалізації програмного забезпечення нашої програми обчислюється показник якості, використовуючи коефіцієнт технічного рівня, який враховує вагомість кожного параметра, який розраховується за формулою (4.11):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j}\, B_{i,j}, \qquad (4.11)$$

де n – кількість параметрів, $K_{\text{в}i}$ – коефіцієнт вагомості і–го параметра,

B_i – оцінка і–го параметра в балах.

Аналіз виконано згідно з таблицею 4.6.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	5	10	0.27	2,7
F2	А	X2	20	10	0.36	3,6
	Б	X3	1500	8	0,13	1,04
F3	А	X4	1000	8	0,18	1,44

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \cdots + K_{\text{ТУ}}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 2{,}7 + 3{,}6 + 1{,}44 = 7{,}74$$

$$K_{K2} = 2{,}7 + 1{,}04 + 1{,}44 = 5{,}18$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

1.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 40$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П = 1{,}3$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0{,}7$. Тоді загальна трудомісткість програмування першого завдання дорівнює: $T_1 = 40 \cdot 1{,}3 \cdot 0{,}7 = 36{,}4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р = 30$ людино-днів, $K_П = 1{,}5, K_{СК} = 1, K_{СТ} = 0{,}9$: $T_2 = 30 \cdot 1{,}5 \cdot 0{,}9 = 40{,}5$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (36{,}4 + 40{,}5 + 7{,}74) \cdot 8 = 677{,}12$ людино-годин.

$T_{II} = (36{,}4 + 40{,}5 + 5{,}18) \cdot 8 = 656{,}64$ людино-годин.

У розробці бере участь один програміст з окладом 28000 грн та один експерт з аналітики з окладом 26000. Визначимо середню зарплату за годину за формулою (4.14):

$$С_ч = \frac{М}{T_m \cdot t} \text{ грн.}, \quad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

Маємо для нашого варіанту:

$$С_ч = \frac{28000 + 26000}{2 \cdot 22 \cdot 8} = 153{,}4 \text{ грн}$$

Тоді, розрахуємо заробітну плату за формулою (4.15):

$$C_{ЗП} = C_ч \cdot T_i \cdot K_Д, \quad (4.15)$$

де $C_ч$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$K_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$C_{ЗП\ I} = 153{,}4 \cdot 677{,}12 \cdot 1.15 = 119570{,}83 \text{ грн}$$

$$C_{ЗП\ II} = 153{,}4 \cdot 656{,}64 \cdot 1.15 = 115753{,}38 \text{ грн}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$C_{від\ I} = C_{ЗП\ I} \cdot 0{,}22 = 119570{,}83 \cdot 0{,}22 = 26305{,}58 \text{ грн}$$

$$C_{\text{від II}} = C_{\text{ЗП II}} \cdot 0{,}22 = 115753{,}38 \cdot 0{,}22 = 25465{,}74 \text{ грн}$$

Тепер визначимо витрати на оплату однієї машино-години. ($\text{С}_\text{М}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 28000 грн., з коефіцієнтом зайнятості 0,25 то для однієї машини отримаємо:

$$\text{С}_\text{М} = 12 \cdot 28000 \cdot 0{,}25 = 84000 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$\text{С}_\text{ЗП} = \text{С}_\text{М}\,(1 + K_\text{З}) = 84000 \cdot (1 + 0{,}25) = 105000 \text{ грн}$$

Відрахування на соціальний внесок:

$$C_{\text{від}} = 105000 \cdot 0{,}22 = 23100 \text{ грн}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 17000 грн.

$$\text{С}_\text{А} = \text{К}_\text{ТМ} \cdot \text{К}_\text{А} \cdot \text{Ц}_\text{ПР} = 1{,}15 \cdot 0{,}25 \cdot 17000 = 4887{,}5 \text{ грн}$$

де $\text{К}_\text{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$\text{К}_\text{А}$ – річна норма амортизації;

$\text{Ц}_\text{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$\text{С}_\text{Р} = \text{К}_\text{ТМ} \cdot \text{Ц}_\text{ПР} \cdot \text{К}_\text{Р} = 1{,}15 \cdot 17000 \cdot 0{,}06 = 1173 \text{ грн}$$

де $\text{К}_\text{Р}$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{\text{ЕФ}} = (\text{Д}_\text{К} - \text{Д}_\text{В} - \text{Д}_\text{С} - \text{Д}_\text{Р}) \cdot t \cdot K_\text{В} =$$

$$= (365 - 104 - 11 - 13) \cdot 8 \cdot 0{,}45 = 1068{,}6 \text{ год}$$

де $\text{Д}_\text{К}$ – календарна кількість днів у році;

$\text{Д}_\text{В}, \text{Д}_\text{С}$ – відповідно кількість вихідних та святкових днів;

$\text{Д}_\text{Р}$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$\text{К}_\text{В}$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_{\text{З}} \cdot \text{Ц}_{\text{ЕН}} = 1068{,}6 \cdot 0{,}25 \cdot 0{,}2 \cdot 9{,}43 = 503{,}84 \text{ грн}$$

де $N_{\text{С}}$ – середньо-споживча потужність приладу;

$K_{\text{З}}$ – коефіцієнтом зайнятості приладу;

$\text{Ц}_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_{\text{Н}} = \text{Ц}_{\text{ПР}} \cdot 0{,}67 = 17000 \cdot 0{,}67 = 11390 \text{ грн}$$

Тоді, річні експлуатаційні витрати будуть рахуватись за формулою:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}}, \quad (4.16)$$

$$C_{\text{ЕКС}} = 105000 + 23100 + 4887{,}5 + 1173 + 503{,}84 + 11390 = 141554.34 \text{ грн}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = \frac{C_{\text{ЕКС}}}{T_{\text{ЕФ}}} = \frac{141554{,}34}{1068{,}6} = 132{,}46 \text{ грн/год}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу обчислюються за формулою (4.17):

$$C_{\text{М}} = C_{\text{М-Г}} \cdot T, \quad (4.17)$$

$$C_{\text{М I}} = 132{,}46 \cdot 677{,}12 = 89696{,}12 \text{ грн}$$

$$C_{\text{М II}} = 132{,}46 \cdot 656{,}64 = 86978{,}53 \text{ грн}$$

Накладні витрати складають 67% від заробітної плати і обчислюються за формулою (4.18):

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0{,}67, \quad (4.18)$$

$$C_{\text{Н I}} = 119570{,}83 \cdot 0{,}67 = 80126{,}46 \text{ грн}$$

$$C_{\text{Н II}} = 115753{,}38 \cdot 0{,}67 = 77554{,}76 \text{ грн}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}}, \quad (4.19)$$

$$C_{\text{ПП I}} = 119570{,}83 + 26305{,}58 + 89696{,}12 + 80126{,}46 = 315698{,}99 \text{ грн}$$

$$C_{\text{ПП II}} = 115753{,}38 + 25465{,}74 + 86978{,}53 + 77554{,}76 = 305752{,}41 \text{ грн}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою (4.20):

$$K_{\text{ТЕР}j} = \frac{K_{\text{К}j}}{C_{\Phi j}}, \tag{4.20}$$

$$K_{\text{ТЕР 1}} = \frac{7{,}74}{315698{,}99} = 2{,}45 \cdot 10^{-5}$$

$$K_{\text{ТЕР 2}} = \frac{5{,}18}{305752{,}41} = 1{,}69 \cdot 10^{-5}$$

Можемо побачити, що найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР 1}} = 2{,}45 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, які залишились після першого відбору двох варіантів виконання програмного комплексу, оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{\text{ТЕР 1}} = 2{,}45 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

1) варіант мови програмування Python;

2) вибір бібліотеки для фільтрації Surprise (SVD);

3) вибір середовища розробки Jupyter Notebook.

Даний варіант виконання програмного комплексу дає користувачу зручну реалізацію програми, зрозумілий інтерфейс, швидку роботу сервера і ефективний та інформативний аналітичний модуль.

4.8 Висновки до розділу 4

У даному розділі було проведено функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

Головною метою даного розділу було визначення найважливіших функцій користувача, оцінка їх впливу на загальну ефективність та розуміння того, які ресурси мають бути задіяні у реалізації програмного продукту.

На основі отриманих результатів визначено найоптимальніший підхід, який забезпечує мінімальні витрати та максимальну продуктивність.

ВИСНОВКИ

У даній дипломній роботі було проведено дослідження та реалізовано гібридну рекомендаційну систему фільмів, яка поєднує методи контентної та колаборативної фільтрації. У рамках дослідження було розглянуто ефективність підходів рекомендацій на основі метаінформації про фільми (через обробку жанрів із застосуванням TF-IDF) та латентних факторів, отриманих за допомогою алгоритму SVD із бібліотеки Surprise.

Було реалізовано обидві моделі незалежно, а також запропоновано підхід до їх поєднання у вигляді гібридної моделі з можливістю налаштування вагового коефіцієнта. Система протестована на наборі даних MovieLens ml-latest-small, а якість рекомендацій оцінена за метриками RMSE та MAE.

Результати експериментального дослідження продемонстрували, що гібридна модель забезпечує загальне покращення результатів у порівнянні з кожним із окремих підходів через вирішення проблеми холодного старту.

Крім того, реалізована система демонструє гнучкість, масштабованість та придатність до подальшої інтеграції в реальні платформи з великими обсягами даних. Запропонований підхід може бути використаний у стримінгових сервісах, інформаційних порталах або інших платформах для підвищення якості рекомендацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Konstan J. A. Recommender systems: from algorithms to user experience. User Modeling and User-Adapted Interaction. 2012. Vol. 22, No. 1– 2. P. 102–128.

2. Schafer J. B., Konstan J. A., Riedl J. E-commerce recommendation applications. Data Mining and Knowledge Discovery. 2001. Vol. 5, No. 1–2. P. 108–121.

3. Sarwar B., Karypis G., Konstan J., Riedl J. Analysis of recommendation algorithms for e-commerce. In: Proceedings of the 2nd ACM Conference on Electronic Commerce. Minnesota, USA, Oct. 17–20, 2000. P. 144–158.

4. Pu P., Chen L., Hu R. A user-centric evaluation framework for recommender systems. In: Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys'11). New York, NY, USA : ACM, 2011. P. 162– 179.

5. Candillier L., Meyer F., Boullé M. Comparing state-of-the-art collaborative filtering systems. In: Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition. Lecture Notes in Computer Science, Vol. 4571. 2007. P. 555–567.

6. Isinkaye F. O., Folajimi Y. O., Ojokoh B. A. Recommendation systems: principles, methods and evaluation. Egyptian Informatics Journal. 2015. Vol. 16. P. 258–277.

7. Das D., Sahoo L., Datta S. A survey on recommendation system. International Journal of Computer Applications. 2017. Vol. 160, No. 7. P. 1–15.

8. Resnick P., Varian H. R. Recommender systems. Communications of the ACM. 1997. Vol. 40. P. 56–58.

9. Wikipedia. Spotify [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Spotify (дата звернення: 11.05.2025).

10. ProIT. Spotify, Apple Music чи YouTube Music: який стрімінговий сервіс обрати у 2024 році? [Електронний ресурс]. URL: https://proit.ua/spotify-applemusic-chi-youtube-music-iakii-striminghovii-siervis-obrati-u-2024-rotsi/ (дата звернення: 11.05.2025).

11. Wikipedia. YouTube [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/YouTube (дата звернення: 11.05.2025).
12. Netflix. Netflix [Електронний ресурс]. URL: https://www.netflix.com/ua/ (дата звернення: 11.05.2025).
13. Galante G. Practical guide to content-based recommender systems [Електронний ресурс]. Medium. 2022. 12 May. URL: https://medium.com/@guillaumegalante/a-practical-guide-to-content-basedrecommender-systems-865708a8595d (дата звернення: 11.05.2025).
14. Melissa M. Recommendation systems [Електронний ресурс]. Medium. 2024. 04 Jan. URL: https://medium.com/simple-ml/recommendationsystems-df15f1eaee57 (дата звернення: 11.05.2025).
15. Gan L. Content-based filtering for efficient online materialized view maintenance [Електронний ресурс]. Association for Computing Machinery. 2008. 172 p. URL: https://dl.acm.org/doi/abs/10.1145/1458082.1458107 (дата звернення: 11.05.2025).
16.Factorization Machines (FM) [Електронний ресурс]. Cnblogs. URL: https://www.cnblogs.com/lnas01/p/13179336.html (дата звернення: 11.05.2025).
17. Dziadkiewicz Z. Recommender systems: machine learning metrics and business metrics [Електронний ресурс]. Neptune.ai Blog. 2023. 07 Aug. URL: https://neptune.ai/blog/recommender-systems-metrics (дата звернення: 11.05.2025).

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО МОДУЛЮ

Через право на інтелектуальну власність було вирішено показувати лише частини програмного продукту.

Hybrid_recsys.ipynb – модуль гібридної моделі рекомендаційної системи фільмів.

```
from sklearn.metrics.pairwise import cosine_similarity

userId = 13

print(f'Для користувача з ID {userId} рекомендовано (Hybrid):')

print('-------------------------------------------')

user_movies = movies_with_ratings[movies_with_ratings.userId == userId].title.unique()

scores = []

titles = []

last_user_movie = user_movies[-1]

movie_genres = title_genres[last_user_movie]

movie_genres = change_string(movie_genres)

predict = count_vect.transform([movie_genres])

X_tfidf2 = tfidf_transformer.transform(predict)

res = neigh.kneighbors(X_tfidf2, n_neighbors=50, return_distance=True)

similar_movies = movies.iloc[res[1][0]].title.values

for movie in similar_movies:

    if movie in user_movies:
```

```
            continue
        svd_score = svd.predict(uid=userId, iid=movie).est
        content_score = 1.0
        try:
            genres_query = change_string(title_genres[movie])
            vect = count_vect.transform([genres_query])
            tfidf_vect = tfidf_transformer.transform(vect)
            content_score = cosine_similarity(X_tfidf2, tfidf_vect)[0][0]
        except:
            pass
        content_score_scaled = content_score * 2 + 2.5
        if abs(content_score_scaled - svd_score) > 1.5:
            content_score_scaled = svd_score
        hybrid_score = 0.93 * svd_score + 0.07 * content_score_scaled
        scores.append(hybrid_score)
        titles.append(movie)
    titles = np.array(titles)
    scores = np.array(scores)
    top_n = min(10, len(scores))
    best_indexes = np.argsort(scores)[-top_n:]
    best_indexes = best_indexes[np.argsort(scores[best_indexes])]
    for i in reversed(best_indexes):
```

```
        print(f'{titles[i]}  {scores[i]:.2f}')
```

hybrid_metrics.ipynb – модуль обрахунку та виведення метрик для гібридної моделі рекомендаційної системи фільмів.

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics.pairwise import cosine_similarity
alpha = 0.93  # 93% SVD, 7% content
y_true = []
y_pred = []
for uid, iid, true_rating in testset:
    try:
        svd_score = svd.predict(uid=uid, iid=iid).est
        movie_genres = title_genres.get(iid)
        if movie_genres is None:
            content_score_scaled = svd_score
        else:
            movie_genres = change_string(movie_genres)
            vect = count_vect.transform([movie_genres])
            tfidf_vect = tfidf_transformer.transform(vect)
            user_movies = movies_with_ratings[movies_with_ratings.userId == uid].title.unique()
            if len(user_movies) == 0:
```

```
            X_tfidf2 = tfidf_vect
        else:
            last_movie = user_movies[-1]
            last_genres = change_string(title_genres.get(last_movie, ''))
            pred = count_vect.transform([last_genres])
            X_tfidf2 = tfidf_transformer.transform(pred)
        content_score = cosine_similarity(X_tfidf2, tfidf_vect)[0][0]
        content_score_scaled = content_score * 2 + 2.5
        if abs(content_score_scaled - svd_score) > 1.5:
            content_score_scaled = svd_score
    hybrid_score = alpha * svd_score + (1 - alpha) * content_score_scaled
    y_true.append(true_rating)
    y_pred.append(hybrid_score)
  except Exception as e:
    fallback_score = svd.predict(uid=uid, iid=iid).est
    y_true.append(true_rating)
    y_pred.append(fallback_score)
rmse_hybrid = mean_squared_error(y_true, y_pred) ** 0.5
mae_hybrid = mean_absolute_error(y_true, y_pred)
print(f"Hybrid Model RMSE: {rmse_hybrid:.4f}")
print(f"Hybrid Model MAE:  {mae_hybrid:.4f}")
```

ДОДАТОК Б ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

Побудова рекомендаційної системи фільмів з урахуванням вподобань користувачів та мета-інформації про фільм

Виконав:
студент IV курсу, групи
КА-12 Дроздов Н.А.

Керівник:
Доцент кафедри ММСА, к.ф.-м.н.,
доц. Подколзін Г. Б.

Київ – 2025

ВСТУП

У роботі розглянуто сучасні підходи до побудови рекомендаційних систем, реалізовано два незалежні компоненти: модель колаборативної фільтрації за допомогою алгоритму SVD з бібліотеки Surprise та контентну модель на основі векторного представлення фільмів із використанням TF-IDF. Запропоновано метод поєднання результатів моделей у вигляді гібридної системи. Система тестувалася на наборі даних MovieLens ml-latest-small.

Об'єктом дослідження є платформи та сервіси для перегляду фільмів і серіалів з можливістю персоналізованих рекомендацій.

Предметом дослідження є алгоритми рекомендацій, їхні методи та можливість оптимізації.

Метою роботи є дослідити як будуються рекомендаційні системи, які підходи є найефективнішими при роботі з різними обсягами даних, та реалізувати власну рекомендаційну систему.

Постановка задачі дипломної роботи

Основні завдання роботи:
1. Проаналізувати існуючі методи побудови рекомендаційних систем.
2. Визначити, які з методів найкраще підходять для роботи з різними за обсягом даними.
3. Реалізувати та протестувати власну РС

3

Актуальність дослідження

Головним призначенням рекомендаційних систем є вирішення та допомога у проблемі вибору продукту та скороченні часу на пошук необхідних товарів чи послуг. РС пропонують взаємовигідну співпрацю споживачам, адже клієнти отримують саме те, що їм потрібно, а платформи – прибуток завдяки кращому таргету та збільшенню продажів як результат.

4

Історія та класифікація рекомендаційних систем

Однією з перших РС була система GroupLens від однойменної дослідницької лабораторії, створена у 1994. Саме їх датасет використаний у роботі. За класифікацією РС поділяються на:

1) Колаборативні системи: базуються на аналізі вподобань користувачів.
2) Контентно – орієнтовані системи. Аналізують характеристики фільмів: жанри, актори, режисер тощо, та надають рекомендації на основі тих фільмів, що вже оцінені користувачем.
3) Системи на основі моделей з глибоким навчанням: створені на основі нейронних мереж, що здатні відслідковувати складні патерни взаємодій користувачів із фільмами.
4) РС на основі великих мовних моделей (LLM): забезпечують високий рівень персоналізації завдяки натренованим трансформерам для розуміння описів, відгуків та навіть запитів.
5) Гібридні системи: поєднують у собі декілька методів, часто колаборативний та контентно – орієнтований.

5

Існуючі рекомендаційні системи

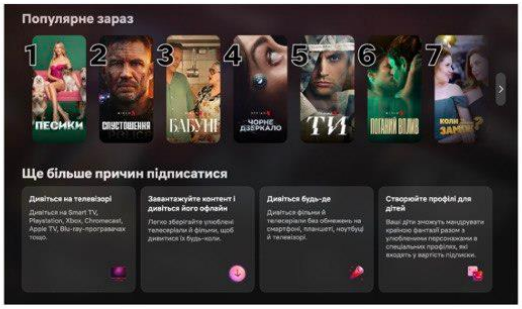

Рис. 1 Рекомендаційна система сервісу Netflix [1].

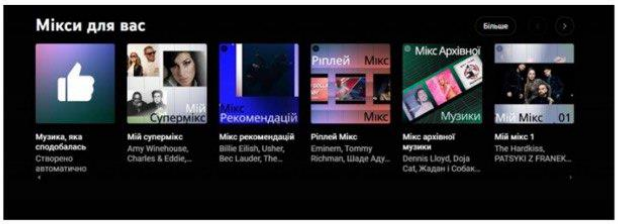

Рис. 2 Рекомендаційна система сервісу Spotify [2].

6

Існуючі рекомендаційні системи

Рис. 3 Рекомендаційна система сервісу YouTube.

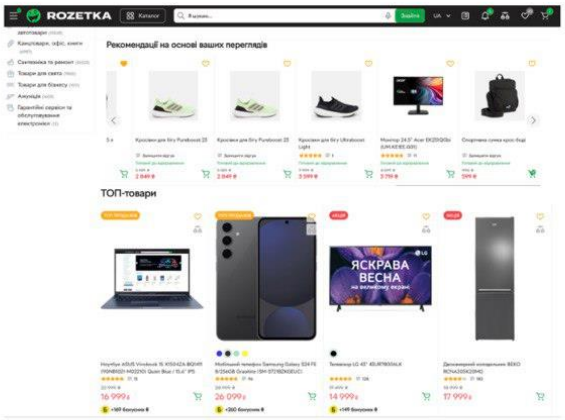

Рис. 4 Рекомендаційна система маркетплейсу Rozetka

Рис. 5 Рекомендаційна система платформи Instagram. Робота з чітким запитом.

7

Детальний огляд основних методів фільтрації

Контентно–орієнтована фільтрація (content–based filtering)

Даний метод фільтрації використовує дані про активності та існуючі вподобання користувача та на основі них створює рекомендації. Можемо виділити наступні головні дані:

- Жанр
- Ключові слова (теги)

Метод ґрунтується на аналізі даних про фільм та індивідуальних вподобань користувача, а рекомендації створюються через порівняння нових об'єктів (за умови достатньої кількості даних) з тими, що вже були позитивно оцінені користувачем.

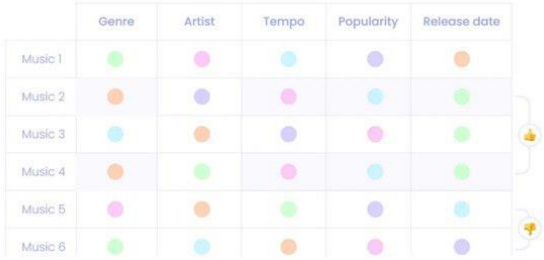

Рис. 6 Приклад даних для контентно-орієнтованої фільтрації. [3]

8

Детальний огляд основних методів фільтрації

Контентно–орієнтована фільтрація (content–based filtering)

На рис. 7 наведено матрицю взаємодій, де кожен рядок — це вектор ознак (embedding vector), який представляє окремий відеокліп, кожен з яких описується за допомогою характеристик.

Горизонтальний вектор знизу відображає користувача, який також описується вищезгаданими характеристиками. Задля того, щоб визначити схожість між користувачем та кожним фільмом, використовується скалярний добуток цих векторів — кількість ознак, які наявні одночасно в обох векторах.

Відповідно, чим більше спільних ознак, тим більшим буде скалярний добуток і, як результат, вищим рівень схожості.

Косинусна подібність – найпопулярніша з метрик міри подібності

$$cosine\ similarity\ (A,B) = \cos(A,B) = \frac{A \cdot B}{||A|| \cdot ||B||},$$

де A і B — вектори інтересів користувача та ознак відповідно.

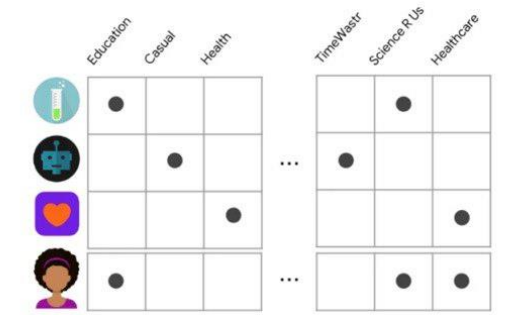

Рис. 7 Матриця взаємодій користувача. [4]

Детальний огляд основних методів фільтрації

Колаборативна фільтрація (collaborative filtering)

Метод колаборативної фільтрації. Його головна перевага — побудова рекомендацій з урахуванням подібностей як між самими користувачами, так і між об'єктами одночасно. Це дозволяє створювати неочевидні, але якісні рекомендації — наприклад, система може запропонувати користувачеві А фільм, яким цікавився користувач В, якщо їхні вподобання мають збіг.

На рис. 8 наведемо наявне порівняння методів контентно-орієнтованої та колаборативної фільтрації

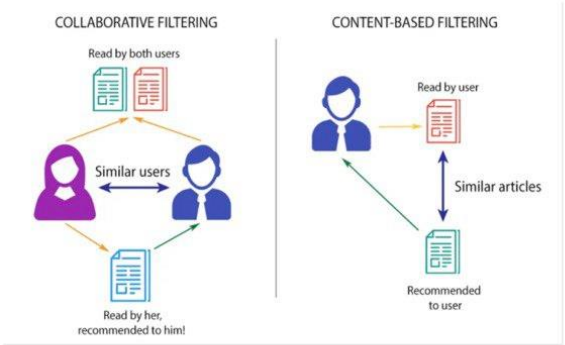

Рис. 8 Наявне порівняння методів контентно-орієнтованої та колаборативної фільтрації. [4]

Детальний огляд основних методів фільтрації

Колаборативна фільтрація (collaborative filtering)

Нехай U – множина всіх користувачів системи (U = {u_1, u_2, u_2 …, u_n}), а I – множина доступних елементів (фільмів) I = {i_1, i_2, …, i_m}. Тоді позначає множину оцінок, де – це оцінка, яку користувач u залишив для фільму i.

Матрична факторизація (MF) – один з найефективніших підходів до побудови великих РС. На практиці матрична факторизація показує високу ефективність у створенні моделей прихованих факторів (latent factor models). Основна ідея методу полягає в тому, щоб на основі шаблонів оцінювання фільмів побудувати вектори ознак, що характеризують як користувачів, так і самі елементи. Надалі рекомендації формуються через міру схожості між вищезгаданими векторами.

Розглянемо матрицю взаємодій $R_{u \times i}$ розміром uxi відповідно, де u — кількість користувачів, а i — кількість фільмів. Отже, матриця відображає наявні взаємодії між користувачами та фільмами. Її можна апроксимувати добутком двох матриць меншого розміру:

$$R_{u \times i} = P_{u \times k} \cdot Q^T_{i \times k}$$

У моделі факторизації є гіперпараметр k, який визначає розмірність векторів ознак — ембедингів (embedding vector). Отримуємо, що кожен користувач і кожен елемент представлені у вигляді дійсного вектора довжини k. Взаємодія між користувачем та елементом моделюється як скалярний добуток відповідних векторів. На рис. 9 та рис. 10 наведено наявний алгоритм матричної факторизації у графічному вигляді за формулою вищче

11

Детальний огляд основних методів фільтрації

Колаборативна фільтрація (collaborative filtering)

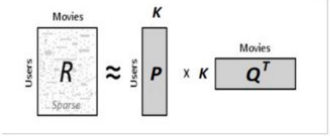

Рис. 9 Алгоритм матричної факторизації у графічному вигляді. [5]

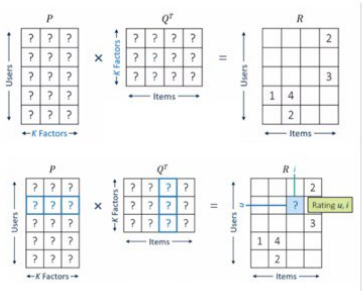

Рис. 10 Приклад роботи алгоритму матричної факторизації.[5]

Далі використовуємо формулу прогнозування оцінки користувача u для фільму i:

$$\widehat{r_{ui}} = \boldsymbol{p_u} \cdot \boldsymbol{q_i^T} = \sum_{k=1}^{K} p_{uk} \cdot q_{ki}$$

Використовуємо формулу для мінімізації похибок (різниці між реальною оцінкою r та прогнозованої) під час навчання моделі.

$$e_{ui}^2 = (r_{ui} - \widehat{r_{ui}})^2 = \left(r_{ui} - \sum_{k=1}^{K} p_{uk} \cdot q_{ki} \right)^2$$

У рівнянні нище попередню модель розширено завдяки урахуванню упередженості, а також додано регуляризаційні члени, які забезпечують стабільність і коректність значень прихованих факторів

$$\widehat{r_{ui}} = \mu + b_u + b_i + p_u \cdot q_i^T$$

де μ, b_u, b_i — це середнє упередження, упередження користувача та упередження елемента відповідно.

12

Детальний огляд основних методів фільтрації

Гібридна модель

Контентно–орієнтована та колаборативна фільтрації мають як переваги, так і недоліки. Для мінімізації недоліків та максимізації переваг варто використовувати саме гібридні моделі, в яких враховуються слабкі та сильні сторони обох методів.

Було використано комбінацію результатів кількох окремих моделей шляхом зваженого сумування через ваговий коефіцієнт. Наприклад, якщо r_{ui}^{CF} — прогноз на основі колаборативної фільтрації, а r_{ui}^{CB} — прогноз на основі content-based фільтрації, то остаточний результат буде обчислюватись за формулою:

$$\widehat{r_{ui}} = \alpha \cdot \widehat{r_{ui}^{CF}} + (1 - \alpha) \cdot \widehat{r_{ui}^{CB}}$$

де $\alpha \in [0,1]$— ваговий коефіцієнт, що визначає важливість та відповідно вагу кожної складової. Можливість вибору α надає змогу регулювати баланс між двома підходами. Наприклад, при α= 0.7 70% довіри (ваги) надається колаборативній частині, а 30% — контентній.

Контентні ознаки фільмів можна подати у вигляді багатовимірного вектору ознак. Для цього можна застосувати TF-IDF — статистичну міру, що оцінює важливість терміну в фільмі відносно всієї колекції.

$\text{TF-IDF}(t,d) = \text{TF}(t,d) \cdot \text{IDF}(t)$, де

– $\text{IDF}(t) = \log\left(\frac{N}{1+n_t}\right)$, де N— загальна кількість документів, n_t— кількість документів, що містять термін t.

– $\text{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$ — частота терміна t у документі d.

13

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

Для реалізації програми було обрану мову програмування Python та середовище VSCode. У практичній частині вирішено порівняти результати та оцінки роботи алгоритмів, що будуть автоматично виведені програмою. Після підключення необхідних бібліотек було завантажено дата-сет з офіційного сайту MovieLens у форматі csv.

У файлі movies.csv зберігається інформація про фільми: поле movieId відповідає за унікальний ідентифікатор фільму, title — назву, genres — за перелік жанрів, розділених символом «|». Загально у файлі представлено дані щодо 9742 фільмів.

Файл ratings.csv містить інформацію про оцінювання фільмів користувачами: userId — унікальний ідентифікатор користувача, movieId — ідентифікатор фільму, rating — виставлена оцінка, timestamp — час оцінювання. Загальна кількість оцінок – 100836.

Було прийнято рішення розробити та порівняти результати моделей, заснованих на:

1) Контентно-орієнтованій фільтрації (content-based)
2) Колаборативній фільтрації (модель SVD з бібліотеки surprise)
3) Гібридному методі з двох попередніх

```
movies
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9742 non-null   int64
 1   title    9742 non-null   object
 2   genres   9742 non-null   object
dtypes: int64(1), object(2)
memory usage: 228.5+ KB
Дубльованих записів: 0
----------
ratings
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100836 non-null  int64
 1   movieId    100836 non-null  int64
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
...
dtypes: int64(3), object(1)
memory usage: 115.2+ KB
Дубльованих записів: 0
----------
```

Рис. 11 Результат аналізу даних

14

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

Відповідно було отримано наступні результати:

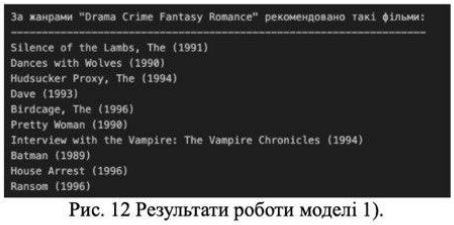

Рис. 12 Результати роботи моделі 1).

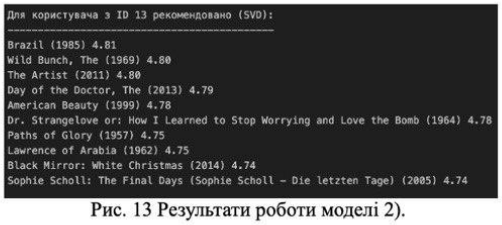

Рис. 13 Результати роботи моделі 2).

Рис. 13 Результати роботи моделі 3).

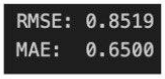

Рис. 14 Метрики моделі 2).

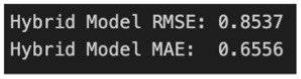

Рис. 15 Метрики моделі 3).

15

ВИСНОВКИ

У даній дипломній роботі було проведено дослідження та реалізовано гібридну рекомендаційну систему фільмів, яка поєднує методи контентної та колаборативної фільтрації. У рамках дослідження було розглянуто ефективність підходів рекомендацій на основі метаінформації про фільми (через обробку жанрів із застосуванням TF-IDF) та латентних факторів, отриманих за допомогою алгоритму SVD із бібліотеки Surprise.

Було реалізовано обидві моделі незалежно, а також запропоновано підхід до їх поєднання у вигляді гібридної моделі з можливістю налаштування вагового коефіцієнта. Система протестована на наборі даних MovieLens ml-latest-small, а якість рекомендацій оцінена за метриками RMSE та MAE.

Результати експериментального дослідження продемонстрували, що гібридна модель забезпечує загальне покращення результатів у порівнянні з кожним із окремих підходів через вирішення проблеми холодного старту.

Крім того, реалізована система демонструє гнучкість, масштабованість та придатність до подальшої інтеграції в реальні платформи з великими обсягами даних. Запропонований підхід може бути використаний у стримінгових сервісах, інформаційних порталах або інших платформах для підвищення якості рекомендацій.

16

Напрямки подальшого розвитку

- Розробка автоматизації підбору вагового коефіцієнта гібридної моделі
- Інтеграція моделі в більш зручний графічний інтерфейс
- Розширення системи на часову динаміку вподобань задля виявлення трендів

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. *Netflix*. URL: https://www.netflix.com/ua/.

2. *Proit*. URL: https://proit.ua/spotify-apple-music-chi-youtube-music-iakii-striminghovii-siervis-obrati-u-2024-rotsi/.

3. Галанте Г. Практичний посібник із систем рекомендацій на основі контенту. *Medium*. 12.05.2022. URL: https://medium.com/@guillaumegalante/a-practical-guide-to-content-based-recommender-systems-865708a8595d.

4. Melissa M. Recommendation Systems. *Medium*. 04.01.2024. URL: https://medium.com/simple-ml/recommendation-systems-df15f1eaee57.

5. Factorization Machines (FM). *Cnblogs*. URL: https://www.cnblogs.com/lnas01/p/13179336.html.

Дякую за увагу!