Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження рекомендаційних систем для довгострокового задоволення користувачів на прикладі сучасних платформ персоналізації контенту
(тема)

Виконала:
здобувачка 2 року навчання
групи ІПЗм-23-1

Стефні ОГУ
(Власне ім’я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник доц. Наталя ВАЛЕНДА
(посада, Власне ім’я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Огу Стефні Іфомі
(прізвище, ім'я, по батькові)

1. Тема роботи *«Дослідження рекомендаційних систем для довгострокового задоволення користувачів на прикладі сучасних платформ персоналізації контенту»*

Затверджена наказом по університету від 15.04. 2025р. № 290 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16.06.2025

3. Вихідні дані до роботи опис досліджуваних алгоритмів рекомендацій, вимоги до розробки гібридної системи рекомендацій для проведення досліджень за обраною предметною областю, мови програмування Python, технології Flask/Django, машинне навчання, набір даних MovieLens 20M.

4. Перелік питань, що потрібно опрацювати в роботі аналіз та порівняння існуючих підходів до рекомендаційних систем, дослідження проблеми довгострокової задоволеності користувачів, реалізація алгоритмів контент-орієнтованої та колаборативної фільтрації, впровадження системи проксі-нагород та контекстуальних бандитів, розробка веб-інтерфейсу для тестування системи.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	16.04.2025	*виконано*
2	Аналіз предметної галузі і постановка задачі	17.04.2025	*виконано*
3	Огляд й аналіз літературних, наукових джерел	25.04.2025	*виконано*
4	Постановка задачі	25.04.2025	*виконано*
5	Теоретичне дослідження	30.04.2025	*виконано*
6	Проектування архітектури, розробка прототипу додатку	05.05.2025	*виконано*
7	Розробка модулю жанрових рекомендацій	10.05.2025	*виконано*
8	Розробка модулю колаборативних рекомендацій	20.05.2025	*виконано*
9	Розробка модулю довгострокового покращення	30.06.2025	*виконано*
10	Підготовка пояснювальної записки	10.06.2025	*виконано*
11	Підготовка презентації та доповіді	10.06.2025	*виконано*
12	Перевірка на плагіат	16.06.2025	*виконано*
13	Нормоконтроль	16.06.2025	*виконано*
14	Рецензування	17.06.2025	*виконано*
15	Попередній захист	18.06.2025	*виконано*
16	Занесення диплома в електронний архів	20.06.2025	*виконано*
17	Допуск до захисту у зав. кафедри	20.06.2025	*виконано*

Дата видачі завдання 16.04.2025

Студент (ка)  Стефні ОГУ

(підпис)

Керівник роботи _____ Доц. Наталя Валенда

(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 113 с., 44 рис., 2 табл., 40 джерел.

КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КОНТЕКСТУАЛЬНИЙ БАНДИТ, МАТРИЧНА ФАКТОРИЗАЦІЯ, ПРОКСІ-ВИНАГОРОДИ, СИСТЕМИ РЕКОМЕНДАЦІЙ, ФІЛЬТРАЦІЯ НА ОСНОВІ КОНТЕНТУ, NETFLIX

Об’єктом дослідження є рекомендаційні системи, які враховують довгострокове задоволення користувачів у сучасних платформах персоналізації контенту.

Метою роботи є дослідження можливостей досягнення довгострокового задоволення користувачів у рекомендаційних системах та підготовка до практичної частини роботи, а саме розробці рекомендаційної системи фільмів, що оптимізує довгострокову задоволеність користувачів

Методами розробки та проектування є застосування колаборативної фільтрації, матричної факторизації, використання контекстних бандитів та машинного навчання.

У результаті дослідження розроблено систему рекомендацій із модулем навчання та користувацьким інтерфейсом. Також розроблено підхід для використання проксі-нагород, що дозволяє враховувати як короткострокове, так і довгострокове задоволення користувачів, забезпечуючи ефективність рекомендаційної системи.

COLLABORATIVE FILTERING, CONTEXTUAL BANDIT, MATRIX FACTORIZATION, PROXY REWARDS, RECOMMENDATION SYSTEMS, CONTENT-BASED FILTERING, NETFLIX

The object of the research is recommender systems that take into account long-term user satisfaction in modern content personalization platforms.

The aim of the work is to investigate the possibilities of achieving long-term user satisfaction in recommendation systems and prepare for the practical part of the work, namely the development of a film recommendation system that optimizes long-term user satisfaction.

The research and design methods are the use of collaborative filtering, matrix factorization, the use of contextual bandits and machine learning.

As a result of the research, a recommendation system with a learning module and a user interface was developed. An approach for using proxy rewards was also developed, which allows taking into account both short-term and long-term user satisfaction, ensuring the effectiveness of the recommendation system.

Завідувачу кафедри
ПІ
(скорочена назва кафедри)
проф. Кирилу СМЕЛЯКОВУ
(вчене звання, сласне ім'я, прізвище)

ЗАЯВА
щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритого доступу ElAr KhNURE

Я,
Огу Стефні Іфома
(прізвище, ім'я, по батькові)

здобувач вищої освіти на другому (магістерському) рівні вищої освіти академічної групи ІПм-23-1

кафедра програмної інженерії ,
(повна назва кафедри)

заявляю: моя кваліфікаційна роботу на тему
«Дослідження рекомендаційних систем для довгострокового задоволення користувачів на прикладі сучасних платформ персоналізації контенту»,
(назва роботи)

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу "ElArKhNURE". Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з вимогами академічної доброчесності, згідно з якими виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата 16.06.2025 Підпис 

ЗМІСТ

ВСТУП

З ростом кількості додатків Web 2.0[1], таких як блоги, соціальні мережі та інші платформи, що дозволяють користувачам створювати та поширювати контент, кількість онлайн інформації зросла у рази. За даними опитування 2024 року[2], щодня створюється приблизно 402,74 мільйона терабайт даних, цього року буде згенеровано близько 147 зетабайт даних, а у 2025 році буде згенеровано 181 зетабайт даних. Робота з такою великою кількістю даних ставить потребу перед існуючими та новими онлайн-платформами розробляти або впроваджувати механізми рекомендаційних систем для вирішення проблеми інформаційного навантаження.

Рекомендаційні системи вже відіграють ключову роль у сучасному цифровому середовищі, зменшуючи витрати часу та зусиль користувачів, автоматизуючи вибір контенту, та активно застосовуються у численних галузях: від потокового відео та музики до електронної комерції, навчання й соціальних медіа. У такому сценарії, здійснюючи покупки в інтернеті чи досліджуючи потокові платформи, користувачі звикають очікувати індивідуальний досвід, який відповідає їхнім уподобанням та інтересам, глибоко резонує з їхніми унікальними смаками та бажаннями [1]. Традиційно більшість систем рекомендацій навчені оптимізувати короткостроковий зворотній зв’язок користувача, такий як вподобайки, кліки, поширення або тривалість сеансу. Але для не меншої кількості платформ більш важливим є не короткострокове задоволення користувача, довгострокове.

Дана робота має на меті дослідженні можливостей досягнення довгострокового задоволення користувачів у рекомендаційних системах та

---

[1] O’Reilly, T. (2024, December 22). What is Web 2.0. O’Reilly Media. https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

[2] Duarte, F. (2024, June 13). Amount of data created daily (2024). Exploding Topics. https://explodingtopics.com/blog/data-generated-per-day

практичне впровадження результатів дослідження у розробку рекомендаційної системи фільмів, що оптимізує довгострокову задоволеність користувачів.

Для досягнення поставленої мети необхідно виконати такі завдання:

– дослідити теоретичні основи рекомендаційних систем та їх розробку;
– провести аналіз існуючих підходів до моделювання рекомендаційних систем орієнтованих на довгострокове задоволення користувача;
– розробити пропозицію системи, що базується на наборі даних та реалізовує рекомендаційну систему для довгострокового задоволення користувача;
– підсумувати здобуті уроки та запропонувати подальші напрямки досліджень.

Об'єктом дослідження є сучасні рекомендаційні системи на прикладі платформ, які використовують алгоритми персоналізації контенту.

Предметом дослідження є методи штучного інтелекту та машинного навчання для створення рекомендаційних систем, що забезпечують довгострокову задоволеність користувачів, включаючи контент-орієнтовану фільтрацію, колаборативну фільтрацію з матричною факторизацією та контекстуальні бандити з системою проксі-нагород.

Методи дослідження включають аналіз існуючих алгоритмів рекомендацій, розробку гібридних підходів та їх тестування на основі реальних даних MovieLens. Зокрема, буде розглянуто SVD-факторизацію для вирішення проблеми розрідженості даних, алгоритм LinUCB для контекстуальних бандитів та методи оптимізації довгострокової задоволеності користувачів.

Практичне значення одержаних результатів полягає у дослідженні актуальної проблеми довгострокового задоволення користувача у рекомендаційних системах, підвищуючи утримання користувачів та їх лояльність до платформи.

Отримані результати можуть бути застосовані для розробки рекомендаційних систем у різних галузях, включаючи стрімінгові сервіси, електронну комерцію та освітні платформи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Рекомендаційні системи відіграють вирішальну роль у сучасних цифрових платформах, забезпечуючи персоналізований досвід користувачів. Їх ключова особливість полягає у здатності обробляти великі обсяги даних і формувати рекомендації, які відповідають уподобанням користувачів.

1.1 Тенденції та перспективи

Розвиток рекомендаційних систем супроводжується інтеграцією новітніх технологій:

- використання глибокого навчання для обробки мультимедійних даних (зображень, тексту, аудіо);
- мультимодальні моделі, які об’єднують різні джерела даних для підвищення точності рекомендацій;
- застосування етичних принципів у проектуванні систем, таких як прозорість і пояснюваність алгоритмів;
- використання великих мовних моделей (LLM) для аналізу тексту та передбачення вподобань користувачів.

1.2 Рекомендації для довгострокового задоволення користувачів

Тим не менш, з розвитком інформаційних систем, попит користувачів та вимоги індустрії ставлять перед рекомендаційними системами нові виклики.

Загально, метою впровадження механізмів рекомендацій у багатьох систем є отримання кращого рівню утримання клієнтів, ніж якщо рекомендації не надаються [2].

Багато популярних сервісів, на кшталт соціальних мереж Tiktok, Instagram тощо працюють з короткостроковими взаємодіями користувача та ставлять пріоритетом його короткострокове утримання.

У той час же проблемою деяких великих платформ, є довгострокове утримання користувача, якого складніше досягнути. Це відображаю одну специфіку рекомендаційних платформ, що полягає в необхідності враховувати

індивідуальні вподобання користувачів не тільки у реальному часі, але і в довгостроковій перспективі.

У цьому дослідженні розглянуто дві всесвітньо популярні системи: стрімінговий сервіс Netflix та музикальний сервіс Spotify, які є прикладами успішних і відомих рекомендаційних систем.

У Netflix 80% стрімінгового часу досягається за допомогою рекомендаційної системи. Вони також постулюють поліпшення досвіду користувача, спрямованого на утримання клієнтів [3].

А через Spotify та його рекомендації розділу «Зроблено для вас», згідно з нещодавно випущеного звіту Made to be Found, була відкрита понад третина всіх нових виконавців.

Обидві платформи мають на меті розважати світ, постійно забезпечувати позитивний досвід користувача та, головне, що цікавить нас в даному дослідженні, створювати задоволення в довгостроковій перспективі [4] [5].

1.3 Виклики довгострокового задоволення користувачів

Як визначають самі науковці, розробники та дослідники Spotify, традиційно більшість систем рекомендацій навчені оптимізувати короткостроковий зворотній зв’язок (кліки, вподобайки, тривалість сеансу тощо) [5].

А у дослідженнях Netflix визначено, що оптимізація систем рекомендацій для довгострокових результатів є складнішим завданням, ніж оптимізація для короткострокових [6].

Короткострокова оптимізація працює зі зворотнім зв’язком, який можна отримати майже негайно, коли користувач безпосередньо взаємодіє з рекомендованим елементом.

Дані для довгострокової оптимізації складніше відслідкувати, бо вони:

– зазвичай є менш щільними: даних про довгострокові дії користувачів (наприклад, чи повернувся користувач через місяць після взаємодії з рекомендованим елементом) набагато менше, ніж даних про короткострокові дії;

– мають більший рівень шуму: можуть містити більше випадкових факторів, які не пов’язані безпосередньо із самою рекомендацією. Наприклад, задоволення користувача може залежати не лише від якості елементу, але й від його настрою, обставин чи зовнішніх факторів, які важко, або навіть неможливо врахувати;
– мають слабші зв’язки з конкретними рекомендаціями: довгострокові результати (наприклад, задоволення від переглянутого фільму через тижні чи місяці) важче напряму пов’язати з конкретними рекомендаціями, які були надані раніше.

Утім, клієнт має отримати задоволення, бо задоволеність учасників тісно пов’язана з імовірністю того, що користувач залишиться на платформі, продовжить місячну передплату, що безпосередньо впливає на дохід підприємства. Тому оптимізація системи рекомендацій для максимального задоволення клієнтів і збільшення їх утримання являється критичним питанням [4].

Так було визначено, що необхідними являються способи навчання короткостроковій поведінці, щоб оптимізувати довгострокову поведінку. Для цього вже було виявлено деяку кількість підходів [6], а саме: навчання з підкріпленням (Reinforcement Learning) [7], оцінка поза політики (Off-Policy evaluation) [8], метод контекстного бандиту [9]. Тим не менш, практично визначені для використання були не всі з них. Далі розглянемо підходи двох світових лідерів для досягнення довгострокового задоволення користувача.

1.4 Вирішення Netflix

Netflix визначає, що вимірювання довгострокової задоволеності користувачів є складною задачею через віддаленість зворотного зв’язку. Детальніше рішення розглянуто у наступному розділі (див. розділ 2.8.1).

Платформа застосовує методи контекстуальних бандитів, де кожна взаємодія з користувачем розглядається як окремий контекст, а рекомендації адаптуються відповідно до отриманого зворотного зв'язку. Цей зв’язок може бути миттєвими (пропускання, відтворення, оцінка «не подобається» або додавання елементів до

списку відтворення) або відкладеними (завершення шоу чи поновлення підписки). Контекст визначається поточною сесією користувача.

Замість прямої оптимізації показника утримання, яка може бути шумною та важко вимірюваною, Netflix використовує проксі-нагороди – проміжні метрики, що відображають довгострокове задоволення. Наприклад, швидке завершення сезону серіалу або позитивна оцінка після перегляду вказують на високу задоволеність користувача. Такий підхід дозволяє системі швидше адаптуватися до вподобань користувачів, не чекаючи на довгострокові результати, як-от продовження підписки.

1.5 Вирішення Spotify

Spotify стикається з подібними викликами у випадках, коли довгострокові метрики, такі як повторні прослуховування або утримання, стають доступними лише через тривалий час після рекомендації [5] [10].

Щоб подолати цю проблему, компанія використовує проміжні результати, отримані між моментом рекомендації та остаточним спостереженням довгострокової метрики. Це дозволяє швидше оцінювати ефективність нових рекомендацій та адаптувати систему без значних затримок. Детальніше розглянуто у наступному розділі (див. розділ 2.8.2).

1.6 Масштаб проблеми, обмеження та виклики

Дослідження показують, що оптимізація довгострокового задоволення користувачів є актуальним викликом для компаній.

Основні виклики галузі включають:

– проблему "холодного старту" для нових користувачів або контенту;
– розрідженість даних, що ускладнює знаходження зв’язків між користувачами та елементами;
– затриманий зворотний зв’язок, що впливає на точність оцінки довгострокового задоволення;

– конфіденційність даних, що стає критичною у зв’язку зі зростанням регуляторних вимог.

Також, як вже було зазначено, існує кілька підходів вирішення проблеми. Але пріоритетно зробити це ефективним способом, який зможе працювати з величезними обсягами даних.

1.7 Рівень інноваційності дослідження

Це дослідження є практично значущим і інноваційним, оскільки пропонує розробку простої рекомендаційної системи з урахуванням довгострокових потреб користувачів, базуючись на публічно доступних датасетах, таких як MovieLens. Результати роботи можуть бути застосовані в різних галузях, включаючи освіту, медицину та електронну комерцію.

2 ОГЛЯД Й АНАЛІЗ ЛІТЕРАТУРНИХ, НАУКОВИХ ДЖЕРЕЛ

2.1 Огляд й аналіз літературних, наукових джерел

2.1.1 Огляд основних джерел

Дане дослідження рекомендаційних систем базується на аналізі широкого спектра літератури. До огляду були відібрані публікації у журналах і конференціях тези, статті з технічних блогів, книги та звіти (див. рис. 3.1).

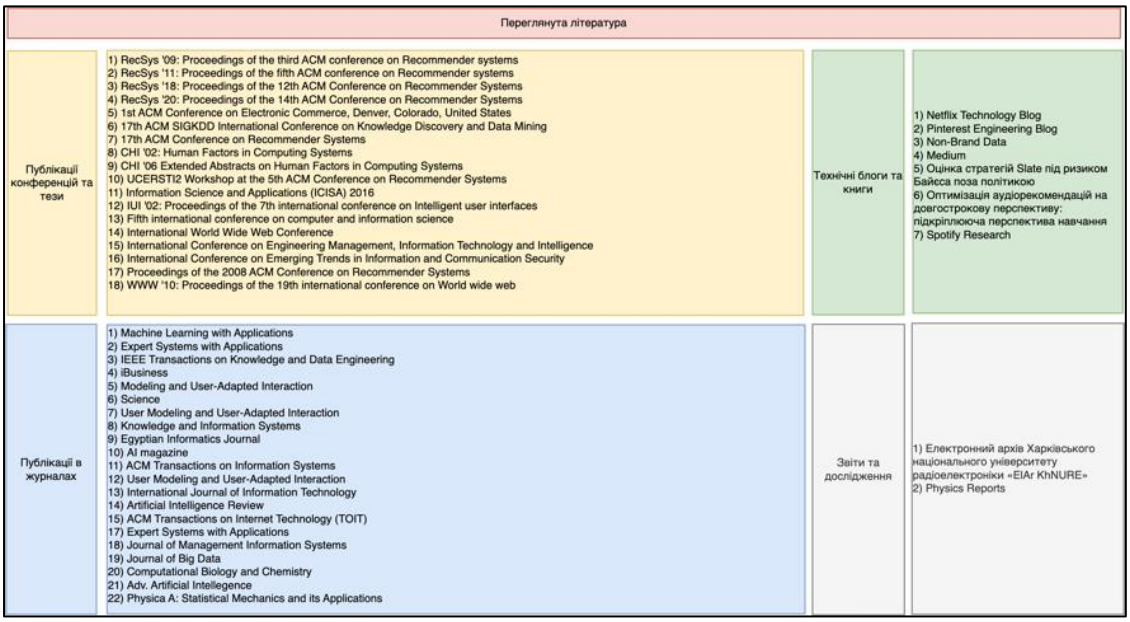


Рисунок 3.1 – Класифікація літературних джерел за типом публікації (рисунок виконано самостійно)

Огляд механізмів створення рекомендаційних систем, їх використання, поточного стану галузі та наявних викликів займає суттєву частину дослідження. Публікації у журналах, такі як Machine Learning with Applications, Expert Systems with Applications, AI Magazine, TheWebConf: The ACM Web Conference та Egyptian Informatics Journal, представляють найважливіші наукові досягнення в галузі рекомендаційних систем.

Конференційні матеріали, зокрема ACM RecSys, CHI та IUI, відображають актуальні тенденції у дослідженні взаємодії користувачів із рекомендаційними системами. Вони демонструють експерименти з оцінкою користувацького досвіду та нові методи персоналізації.

Звіти, зокрема роботи у електронному архіву Харківського національного університету [11] [12] [13], доповнюють аналіз систематизованими дослідженнями, проведеними в академічному середовищі. Ці джерела слугують основою для порівняння академічного та практичного підходів у побудові рекомендаційних систем.

Основна увага приділена технічним блогам Netflix Technology Blog, Spotify Research, Pinterest Engineering Blog, Medium та публікаціям науковців Netflix і Spotify, що були використані для пошуку цінної інформацію про актуальні проблеми галузі, практичну реалізацію алгоритмів від спеціалістів компаній. Вони ілюструють, як сучасні індустріальні лідери впроваджують інновації у свої платформи та пропонують внутрішні рішення.

Основні теорії та концепції, розглянуті в літературі представлені на рисунку нижче (див. рис. 2.2).

Загальні огляди та класифікації рекомендаційних систем	- Принципи, методи та оцінка РС. - Систематичний огляд та перспективи дослідження РС - Загальний огляд РС - Огляд сучасного стану та можливих розширень РС - Таксономія агентів рекомендацій в Інтернеті - Проблеми, виклики та можливості дослідження РС.
Контекстні аспекти та окремі застосування рекомендаційних систем	- Машинне навчання в персоналізованих рекомендаціях - Вивчення вподобань нових користувачів - РС в електронній комерції - Вплив РС на продажі. - Надійність та безпека РС - Безпека та приватність в РС - Соціальні мережі та мобільність. - Прогнозування мобільності - Рекомендації для подорожей - Рекомендації локацій та активностей - Алгоритми TikTok - Онтології та міри подібності - Відстеження погляду в РС - Музичні рекомендації на основі емоцій - Поведінка онлайн користувачів та її вплив на фільтрацію інформації
Алгоритми та методи рекомендаційних систем	- Багаторукі бандити в РС - Контентна фільтрація - Асоціативний пошук для колаборативної фільтрації - Адаптивна фільтрація - Огляд методів колаборативної фільтрації - Колаборативні інформаційні фільтри - Гібридні РС - Заповнення матриць - Колаборативна фільтрація з використанням соціальних мереж - Система рекомендацій, заснована на даних взаємодії користувача, застосована до інтелектуальних електронних книг - Персоналізована система рекомендацій маршруту в режимі реального часу для туристів, які рухаються самостійно, на основі зв'язку між транспортними засобами
Оцінка та метрики рекомендаційних систем	- Оцінка, орієнтована на користувача - Роль прозорості в РС - Метрики точності та їх вплив - Рекомендації на основі критики - Оцінка колаборативної фільтрації - Цілі та метрики оцінювання РС
Системи рекомендацій для довгострокового залучення	- Розуміння системи довгострокового задоволення Netflix - Глибоке навчання для РС: кейс Netflix - Інновації винагород для довгострокового задоволення в Netflix - Персоналізована головна сторінка Netflix - Довгострокове задоволення користувачів в Netflix - Як Pinterest навчає свої комп'ютери бачити - Як Pinterest використовує машинне навчання - MLEnv: стандартизація ML на Pinterest під одним механізмом ML для прискорення інновацій - Архітектура системи рекомендацій TikTok - Дослідження алгоритму самовдосконалення TikTok на основі Розширена взаємодія з користувачем - Оцінка стратегій Slate під ризиком Байєса поза політикою - Навчання співпраці в багатомодульних рекомендаціях за допомогою мультиагентного підсилення. Навчання без спілкування - Контекстно-бандитський підхід до персоналізованих рекомендацій статей новин - Оптимізація на довгострокову перспективу без затримок Spotify - Оптимізація аудіорекомендацій на довгострокову перспективу: підкріплююча перспектива навчання
Існуючі дослідження за темою рекомендаційних систем архіву Харківського національного університету радіоелектроніки	- Дослідження типів колаборативної фільтрації для побудови прогнозів у рекомендаційних системах - Дослідження використання рекомендаційних систем в вебсистемі у сфері освіти - Дослідження методів аналізу рекомендаційних систем для вирішення задач оптимізації використання часу - Дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор - Дослідження методів штучного інтелекту для розробки рекомендаційної системи в сфері аніме

Рисунок 2.2 –Тематичний аналіз літературних джерел з рекомендаційних систем (рисунок виконано самостійно)

Теоретична база дослідження охоплює джерела 1998 – 2024 років, але більшість припадає на джерела останнього десятиріччя.

2.1.2 Оцінка актуальності та новизни

Актуальність літератури підтверджується її фокусом на сучасних проблемах, таких як довгострокове задоволення користувачів. Ці дослідження та відкрити питання опубліковані Netflix та Spotify у період переважно 2024-ого року.

Інноваційність у цій галузі полягає в здатності адаптувати рекомендації до динамічних змін поведінки користувачів. Новизна представлених досліджень полягає у дослідженні сучасного підходу вирішення актуальної проблеми рекомендацій.

2.2 Рекомендаційні системи

2.2.1 Визначення

Рекомендаційні системи визначаються як стратегія прийняття рішень для користувачів у складних інформаційних середовищах [14]. Служба новин, наприклад, може запам’ятати статті, які читав користувач. Наступного разу, коли той відвідує сайт, система рекомендуватиме статті на основі тих, які користувач читав раніше.

З точки зору електронної комерції вони визначені як інструмент, який допомагає користувачам шукати в записах знань, які пов’язані з інтересами та перевагами користувачів [15].

Вони також визначаються як системи фільтрації інформації, що здатні пристосовуватися до уподобань користувача [16]. Фільтрація інформації стосується вибору потенційно цікавих чи корисних для користувача елементів з великої колекції та може розглядатися як задача класифікації.

2.2.2 Мета розробки та впровадження рекомендаційних систем

Метою впровадження систем рекомендацій у користувацькі системи є вирішення проблеми перевантаження інформацією, надаючи їм персоналізований ексклюзивний контент і рекомендація певних елементів з набору даних усіх елементів, наявних у системі.

Рекомендаційні системи проектуються для допомоги як користувачам системі, так і постачальникам послуг [17]. Для користувачів, рекомендаційні системи спрощують процес пошуку потрібної інформації, пропонуючи елементи (наприклад, інформацію та продукти), які відповідають особистим інтересам користувачів. Для постачальників послуг, вони зменшують трансакційні витрати на пошук і вибір товарів у середовищі онлайн-покупок [18].

Доведено, що системи рекомендацій покращують процес прийняття рішень і якість [19]. В умовах електронної комерції системи рекомендацій є ефективним засобом продажу більшої кількості продуктів, що збільшує доходи. У наукових бібліотеках системи рекомендацій підтримують користувачів, дозволяючи їм вийти за межі пошуку в каталозі.

Системи рекомендацій використовують різні типи вхідних даних, а саме зворотній зв’язок можна поділити на [20]:

– явний зворотний зв'язок: безпосередньо збирає оцінки користувачів через інтерфейс системи. Оцінки можуть бути представлені бінарними (подобається/не подобається) або реальними (4 зірки з п’яти) значеннями. Цей тип залежить від того, чи бажає користувач залишати оцінки і вважається більш надійнішим, бо забезпечує ясне розуміння вподобань користувача та прозорість механізму генерації рекомендацій;
– неявний зворотний зв'язок: визначає вподобання користувачів, спостерігаючи за їхньою взаємодією, як шаблони навігації, історія покупок або час, витрачений на перегляд відео чи читання веб-сторінок. У цьому випадку користувач не має робити додаткових дій для провадження зворотного зв’язку. Цей тип вважається менш точним, але більш об’єктивним;

– гібридний зворотний зв'язок: поєднує явні та неявні вхідні дані для підвищення точності системи, мінімізуючи слабкі сторони обох методів. Неявні дані використовуються, наприклад, для підтвердження явних оцінок.

Також зворотній зв’язок, як згадувалося раніше, може бути короткостроковим (вподобайки, поширення, коментар при перегляді відео), а може бути довгостроковим (оцінка серіалу після перегляду, оцінка товару після замовлення та користування).

2.2.3 Типи рекомендаційних систем

Системи рекомендацій загалом поділяються на три різні типи, а саме системи рекомендацій на основі вмісту, колаборативні та гібридні рекомендаційні системи [21]. Схематичне зображення розподілення різних типів рекомендаційних систем наведено на рисунку (див. рис. 3.4).

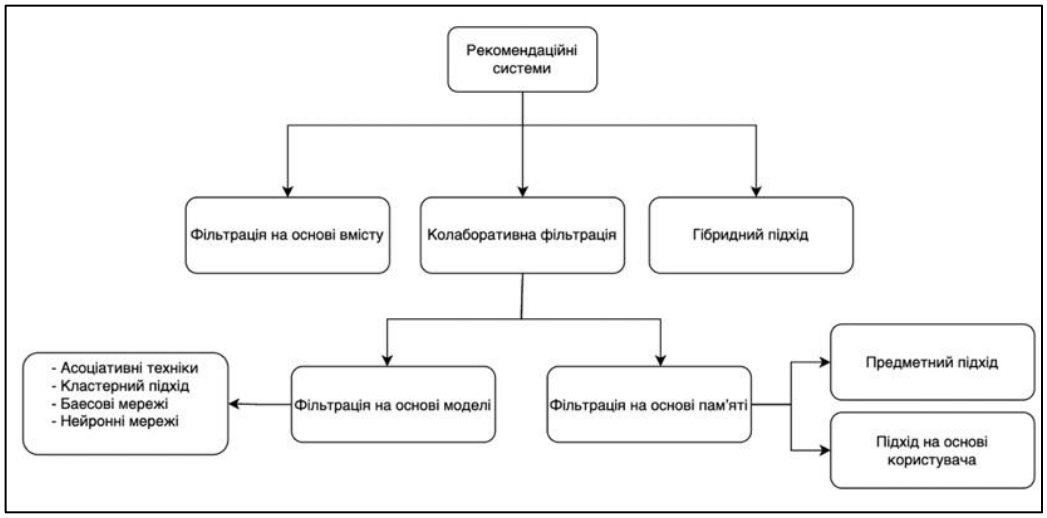


Рисунок 2.4 – Типи рекомендаційних систем (рисунок виконано самостійно)

Розглянемо детальніше типи рекомендаційних систем у наступному розділі.

2.3 Системи рекомендацій на основі вмісту

2.3.1 Принцип роботи

Системи рекомендацій на основі вмісту збирають всі елементи даних в профілі, на основі їх характеристик чи опису [21]. Наприклад, розглядаючи дані

про книги, характеристиками будуть автор, видавець тощо, а характеристиками даних про фільми будуть режисер фільму, актори, жанр тощо.

Коли користувач дає позитивну оцінку елементу (книзі, фільму тощо), тоді інші елементи, присутні у профілі елемента, об'єднуються разом для створення профілю користувача.

Цей профіль користувача об'єднує всі профілі предметів, чиї характеристики оцінені користувачем позитивно. Потім користувачеві рекомендуються елементи, присутні у профілі користувача. Тобто рекомендація на основі вмісту рекомендує користувачеві елементи, вміст яких схожий на вміст тих, що користувачу сподобались раніше, як показано на рисунку (див. рис. 2.5).

Рисунок 2.5 – Принцип роботи рекомендаційних систем на основі вмісту (рисунок виконано самостійно)

Цей підхід вимагає глибоких знань про особливості товару для точної рекомендації, тому не є доцільним до використання, якщо система не передбачає оперування цими знаннями для всіх елементів [21].

2.3.2 Етапи рекомендаційного процесу

Рекомендаційний процес складається з трьох етапів [20]: фаза збору інформації, фаза навчання та фаза передбачення (див. рис. 2.6).

Фаза збору інформації зосереджена на зборі всіх відповідних даних про користувача для створення його профілю чи моделі, на основі яких система надаватиме рекомендації. Профіль користувача може містити такі атрибути, як поведінка користувачів, їх уподобання або типи елементів, з якими вони взаємодіють. Чим більше система знає про користувача, тим кращі рекомендації вона може надати [20].

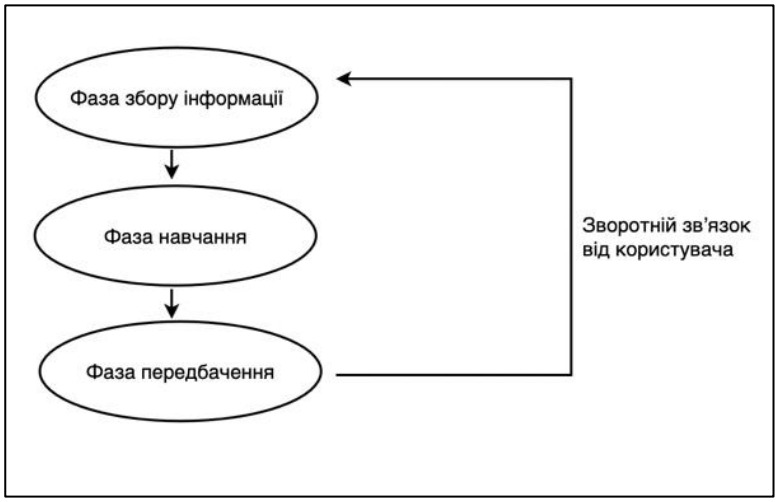


Рисунок 2.6 – Етапи рекомендаційного процесу (рисунок виконано самостійно)

Фаза навчання застосовує алгоритми для аналізу зворотного зв’язку, зібраного на попередній фазі.

Фаза прогнозування передбачає елементи, яким користувач з вищою вірогідністю віддають перевагу. Рекомендації можна генерувати безпосередньо з набору даних, зібраного на етапі збору інформації, використовуючи підходи на основі пам’яті або моделі, або вони можуть бути отримані з постійного спостереження системи за діяльністю користувача.

2.3.3 Характеристика

Цей підхід має здатність динамічно адаптуватися до мінливих уподобань користувача, що можуть змінюватись з часом.

Також цей алгоритм не потребує деталей профілю інших користувачів, оскільки один профіль користувача є специфічним лише для цього користувача, а

інші профілі не впливають на процес рекомендації, що забезпечує безпеку та конфіденційність даних користувачів.

Тип профілю користувачу залежить від використовуваного методу навчання. Використовуються дерева рішень, нейронні мережі та векторні представлення. Профілі користувачів є довгостроковими моделями та оновлюються, коли система спостерігає додаткові дані.

Якщо нові елементи мають достатній опис, методи, засновані на вмісті, можуть частково подолати проблему холодного запуску, тобто ця техніка може рекомендувати елемент, навіть якщо цей елемент раніше не оцінював жоден користувач.

Підходи до фільтрації на основі вмісту поширені в таких системах, як персоналізовані системи рекомендацій новин, публікацій, веб-сторінок тощо.

2.4 Колаборативна фільтрація

2.4.1 Принцип роботи

Колаборативна, або ще можна сказати спільна фільтрація рекомендує товари, ідентифікуючи інших користувачів із подібним смаком та використовуючи їх думку, щоб рекомендувати товари активним користувачам (див. рис. 2.7).

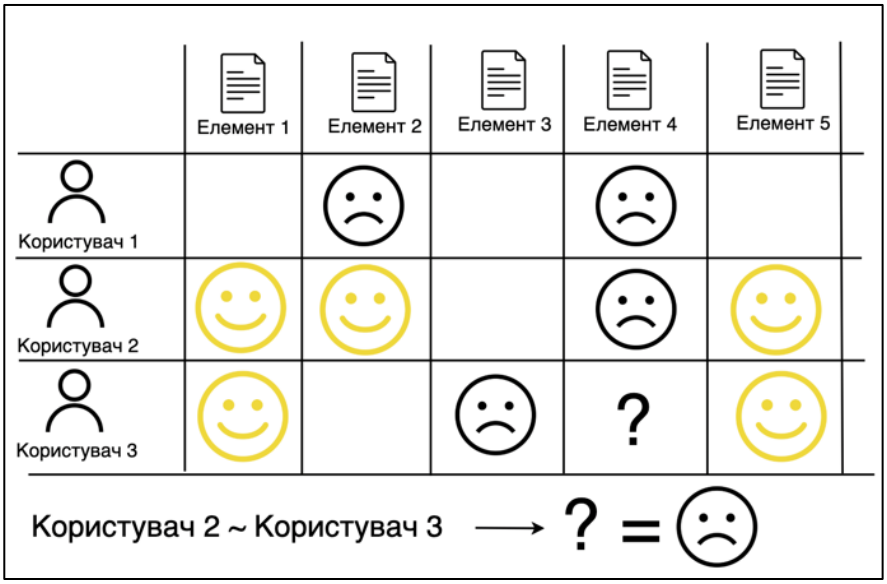


Рисунок 2.7 – Приклад матриці взаємодії на основі користувача (рисунок виконано самостійно)

Типовий профіль користувача в системі для спільної роботи складається з вектору елементів та їх оцінок (вектор взаємодій або уподобань користувача щодо певних предметів), які постійно змінюються з часом при взаємодії користувачу з системою [22].

Цей підхід для прогнозування використовує так звану матрицю взаємодії для прогнозування – матриця в якої рядки відповідають користувачам, а стовпці – предметам (наприклад, фільмам, книгам тощо). Матриця зберігає оцінки, які користувачі надають предметам. Завдання системи полягає у заповненні пропущених значень у матриці, що відповідають невідомим уподобанням.

Колаборативна фільтрація поділяється на два типи: фільтрація на основі моделі та фільтрація на основі пам’яті (який в свою чергу поділяється на предметний підхід, та на підхід на основі користувача) та працює на основі схожості користувачів чи предметів.

2.4.2 Фільтрація на основі пам’яті

Підхід на основі користувача використовує міру подібності між користувачами. Ця техніка починається з пошуку групи користувачів А, чиї вподобання подібні до уподобань користувача Б [21]. Таке угрупування називається сусідством користувачів. Нові елементи, які подобаються більшості користувачам сусідства А, потім рекомендовано користувачеві Б (див. рис. 2.8).

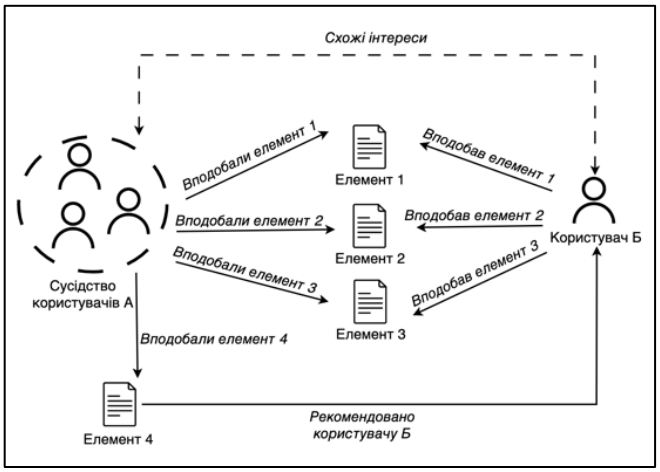


Рисунок 2.8 – Принцип роботи клаборативної фільтрації на основі користувача (рисунок виконано самостійно)

Предметний підхід аналізує схожість між елементами. Наприклад, якщо користувачі оцінюють елементи X і Y схожим чином, система віднесе їх до сусідства елементів – буде вважати їх схожими та порекомендує елемент Y іншим користувачам, які цікавляться елементом X. На рисунку 2.9 представлено, як аналізуються схожості елементів на основі отриманих від користувачів оцінок.

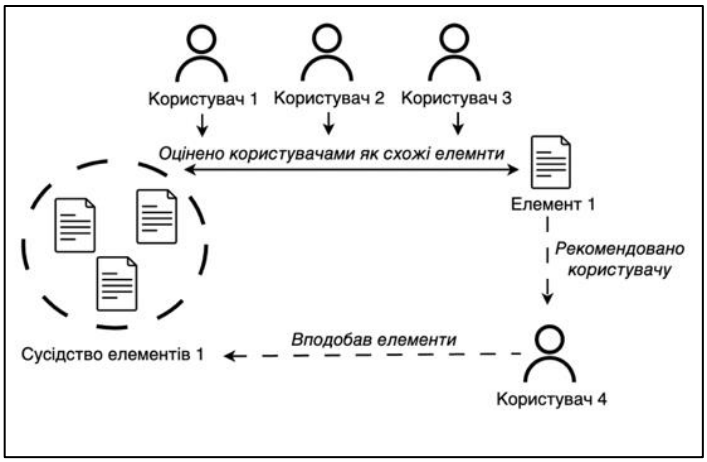


Рисунок 2.9 – Принцип роботи клаборативної фільтрації на основі предметного підходу (рисунок виконано самостійно)

Обидва підходи, засновані на пам’яті, визначають схожість за допомогою вимірювань подібності для чого застосовує такі алгоритми як кореляція Пірсона, косинус подібності, коефіцієнт Жаккара тощо [23].

Для цього обидва підходи використовують матрицю вазємодії, але використовують її по-різному. Нехай, у нас є деяка матриця взаємодії R. Вихідна матриця взаємодії R розбивається на дві менші матриці шляхом матричної факторизації [24]: Q (показуватиме схожість між користувачами) та P (визначатиме схожість між предметами) (див. рис. 2.10).

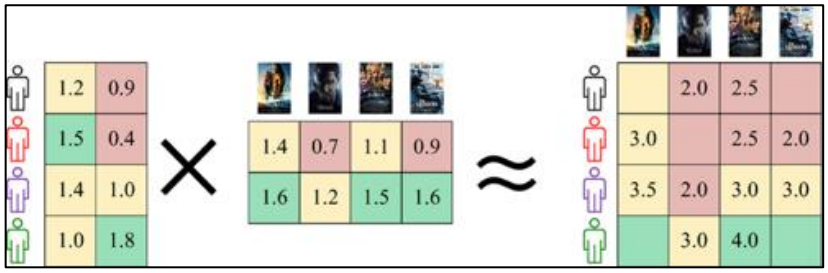


Рисунок 2.10 – Розкладання матриці взаємодії для користувацького та предметного підходів (за даними [24])

Для підходу на основі користувача використовуватиметься матриця Q. Наприклад, якщо двоє користувачів мають схожі вподобання, їм можуть бути запропоновані схожі елементи.

Для предметного підходу – матриця P. Наприклад, якщо два фільми мають схожі характеристики, вони можуть бути рекомендовані користувачам, які вподобали один із них.

Математично це відображено за формулою 2.1 [25]:

$$R \approx Q * P \tag{2.1}$$

де Q має розмірність $n \times k$,

P має розмірність $k \times m$,

n – кількість користувачів,

m – кількість елементів,

k – кількість прихованих факторів.

2.4.3 Фільтрація на основі моделі

Фільтрація на основі моделі працює відмінно від фільтрації на основі пам’яті. Замість використання даних "на льоту", цей підхід створює прогностичну модель, яка використовую ту ж саму матрицю взаємодії, але ґрунтується на прихованих закономірностях у даних. Приховані закономірності – це узагальнені характеристики, які не задаються явно, але виявляються у процесі факторизації. Для випадку елементів, які представлені, наприклад, фільмами – це можуть бути жанри, стилі режисера тощо. А для користувачів у тому ж контексті системи з фільмами – уподобання до жанрів, актора або певного типу контенту. Підхід може працювати зі змішаними ознаками, включаючи як дані про користувачів, так і характеристики елементів.

Для прогнозування використовуються алгоритми машинного навчання, як матрична факторизація (наприклад, SVD (Singular Value Decomposition) або ALS (Alternating Least Squares), нейронні мережі або спектральне кластерування, для побудови моделі, що прогнозує рейтинг.

2.4.4 Етапи рекомендаційного процесу

Рекомендаційний процес колаборативної фільтрації на основі пам’яті можна описати у 5 етапах [26], а саме сбір інформації, представлення даних, попередня обробка, прогнозування рейтингу та рекомендації (див. рис. 2.11).

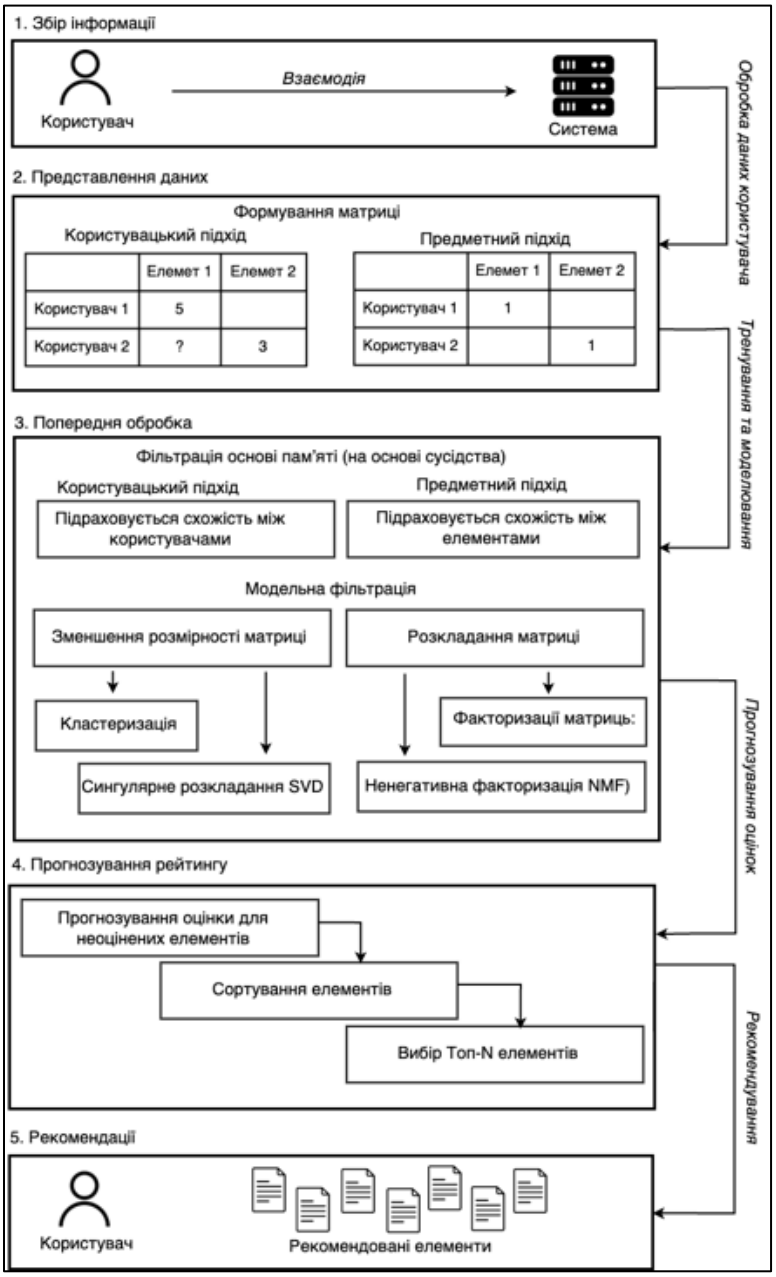


Рисунок 2.11 – Етапи рекомендаційного процесу колаборативної фільтрації (рисунок виконано самостійно)

На першому етапі збору інформації відбувається збір даних о користувачі під час його взаємодії з системою (шаблони купівлі товарів, кількість часу перегляду

відео, запит після читання веб-сторінки тощо).

Другий етап – це представлення даних. Під час нього отримані на першому етапі дані упорядковуються у матрицю взаємодії. Наприклад, оцінки для користувацького підходу (оцінки користувачів до переглянутих фільмів), або поведінка для предметного підходу (бінарні дані, чи переглядав користувач товар).

Третім етапом відбувається попередня обробка, під час якої алгоритм відрізняється для фільтрування на основі моделі та фільтрування на основі пам'яті:

а) під час фільтрування на основі пам'яті підраховується схожість для користувачів чи елементів, та ті об'єднуються у сусідства. Схожість обчислюється за допомогою алгоритмів, таких як косинусна подібність, кореляція Пірсона тощо;

б) для фільтрування на основі моделі використовуються математичні методи для побудови моделі, яка виявляє приховані закономірності в даних, а саме:

1) зменшення розмірності матриці взаємодії без втрати важливої інформації (для роботи з великими і розрідженими даними);

2) розкладання матриці шляхом факторизації для поділу на дві менші – матриці користувачів і предметів. Це дозволяє виявити приховані фактори, які пояснюють взаємодію.

Наступний етап – це прогнозування рейтингу. Відбувається прогнозування пропущених даних у матриці взаємодії для неоцінених користувачем елементів. Прогнозовані елементи сортуються (наприклад за рейтингом релевантної) та формуються у список, Топ-N елементів з яких прогнозуються користувачеві.

Отримані результати показуються користувачу.

2.4.5 Характеристика

Традиційно колаборативні системи на основі фільтрації страждають від проблеми холодного запуску (через відсутність жодних оцінок на елементи від користувачів) та конфіденційності (оскільки існує потреба обмінюватися даними користувача) [21]. Також, у реальних обставинах складно представити, щоб

користувачі взаємодіяли з усіма елементами і тим паче лишали оцінки на них, тому матриця взаємодії зазвичай є розрідженою, через що використовуються методи зменшення розмірності, такі як SVD.

Однак підходи спільного фільтрування не вимагають жодних знань про особливості товару для створення рекомендації. Крім того, цей підхід може допомогти розширити існуючі інтереси користувача шляхом відкриття нових елементів. Так наприклад, користувач платформи фільмів, який цікавиться романтикою, може отримати рекомендацію драматичного фільму з романтичним сюжетом.

2.5 Гібридні рекомендаційні системи

Щоб подолати обмеження окремих видів рекомендаційних алгоритмів, сукупність двох або більше підходів використовується разом, що називається гібридними рекомендаційними системами [21].

Існують різні підходи гібридизації [22], а саме: зважений, перемикаючий, змішаний, комбінація ознак, каскадний, розширення ознак та мета-рівень (див. табл. 2.1).

Таблиця 2.1 – Методи гібридизації (за даними [22])

Метод гібридизації	Опис методу
Зважений (Weighted)	Оцінки (або голоси) кількох технік рекомендацій комбінуються разом для створення однієї рекомендації.
Перемикаючий (Switched)	Система перемикається між техніками рекомендацій залежно від поточної ситуації.
Змішаний (Mixed)	Рекомендації від кількох різних систем пропонуються одночасно.
Комбінація ознак (Feature combination)	Ознаки з різних джерел даних рекомендацій об'єднуються в один алгоритм рекомендацій.

Кінець таблиці 2.1

Метод гібридизації	Опис методу
Каскадний (Cascade)	Одна система рекомендацій уточнює результати, запропоновані іншою системою.
Розширення ознак (Feature augmentation)	Вихід однієї техніки використовується як вхідні дані для іншої.
Мета-рівень (Meta-level)	Модель, навчена однією системою рекомендацій, використовується як вхідні дані для іншої.

Гібридне використання різних методів, як правило, призводить до підвищення продуктивності та підвищення точності в багатьох додатках із рекомендаціями [22].

2.6 Рекомендаційна система Netflix

2.6.1 Опис рекомендаційної системи

Платформа Netflix використовує комбінацію різних моделей і методів машинного навчання для кожної частини системи. Netflix для проблеми рекомендацій виділяє різні підзавдання та поділяє їх виконання між різними системами рекомендацій [2]. Так, наприклад, кожну частину своєї домашньої сторінки поділено на різні проблеми, де для створення рекомендацій використовуються різні системи рекомендацій [27].

На рисунку нижче показано домашню сторінку Netflix з позначеннями, що характеризують різні завдання рекомендацій, кожне з яких виконується за іншим алгоритмом (див. рис. 2.12).

Алгоритм №1 призначений для рекомендації вибору першого для показу на головній сторінці відео, №3 робить рекомендації для повідомлень, алгоритм №5 покликаний допомогти учасникам відкривати нові відео, а алгоритм №7 призначений для ранжування вже переглянутих відео, які користувач може захотіти продовжити перегляд [4]. Вихідні дані кожного з цих алгоритмів можуть відображатися як різні рядки рекомендованих відео на головній сторінці.

Рисунок 2.12 – Використання рекомендаційних алгоритмів на домашній сторінці Netflix (за даними [4])

Для відповіді на питання, який алгоритм працюватиме, окрім завдання рекомендації, вирішальними стають доступні дані та їхні властивості, а саме чи містять дані лише взаємодію користувача з елементом, чи додаткову інформацію, як атрибути користувача, атрибути елемента або контекстну інформацію щодо користувача.

Цікавим емпіричний висновком у літературі є те, що у добре налаштованих системах, де використовуються лише дані про взаємодію між елементами користувача, досить неглибокі алгоритми навчання (зазвичай з одним-трьома прихованими шарами) досягають найкращої точності прогнозування рекомендацій [4]. Тобто, в цьому випадку може використовуватись як і фільтрація на основі пам’яті, так і фільтрація на основі моделі, але застосовуючи неглибокі алгоритми навчання.

Тим не менш, коли дані збагачені додатковою різнорідною інформацією, більш ефективно працюють моделі глибокого навчання, що відсилає нас на використання фільтрації на основі моделі.

2.6.2 Використання моделей глибокого навчання

Netflix впроваджує моделі глибокого навчання (детальніше див. розділ 3.6.2), які можуть обробляти не лише дані взаємодій, але й додаткову інформацію, як, наприклад, мультимедійні характеристики контенту: текст, відео, аудіо та

зображення, що дозволяє створювати приховані представлення контенту, які використовуються для кращого персоналізованого ранжування.

Види моделей глибокого навчання розглянуті нижче (див. рис. 3.13).

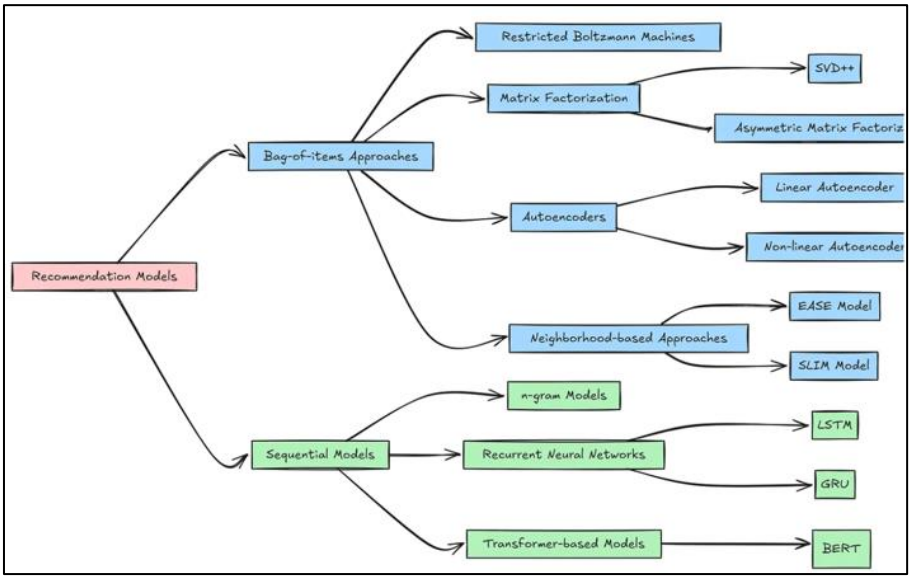


Рисунок 2.13 – Використання моделей глибокого навчання Netflix (за даними [2])

Моделі глибокого навчання поділяються на «Bag of Item» підхід, та послідовні моделі [4]:

а) «Bag of Item» методи – ігнорують часовий порядок взаємодій і розглядають елементи, з якими взаємодіяв користувач, як невпорядковані (мішок слів). Зазвичай вони використовуються для моделювання довгострокових тенденцій:

1) обмежені машини Больцмана (RBM) – підхід на основі нейронної мережі, який передбачає взаємодію між користувачем і елементом, вивчаючи приховані представлення з матриці взаємодії;
2) матрична факторізація (див. розділ 3.3.2):
   – SVD++: розширює традиційну матричну факторизацію, включаючи неявний зворотний зв’язок, наприклад, кліки або тривалість перегляду. Це дозволяє краще враховувати приховані вподобання користувачів;
   – асиметрична матриця факторизації: зменшує складність вбудовування для користувачів, обчислюючи їх вбудовування як

середнє значення елементів, з якими вони взаємодіяли. Це особливо корисно для роботи з новими користувачами;

3) автокодери: нейронні мережі, що створюють вихідний вектора, який є близьким до вхідного вектора шляхом мінімізації помилки реконструкції. У завданні рекомендації ці вектори зазвичай є багатофункціональним кодуванням історії відтворення користувача, тобто кожен елемент вектора відноситься до відео, а його значення дорівнює 1, якщо користувач відтворив відео та 0 в іншому випадку. Окрім двійкових векторів, можна також використовувати безперервні вектори, наприклад, щоб зафіксувати тривалість часу, протягом якого користувач переглядав відео:
   – лінійні автокодер: базова версія автокодера, схожа за принципом роботи на матричну факторизацію з одним прихованим шаром;
   – нелінійний автокодер: складніша модель, яка враховує нелінійні зв’язки між користувачами та елементами;
4) підходи на основі сусідства: аналізують подібність між елементами для рекомендацій, базуючись на попередніх взаємодіях користувача;
   – модель EASE: неглибока модель, яка використовує автокодери для вивчення подібності між елементами, що безпосередньо допомагає у формуванні рекомендацій;
   – модель SLIM: застосовує розріджені лінійні методи для визначення зв’язків між елементами, що покращує точність рекомендацій;

б) послідовні моделі: підходи враховують порядок, у якому користувачі взаємодіють із елементами, що дозволяє враховувати вплив останніх дій користувачів:
1) N-грамові моделі: передбачають наступний елемент у послідовності на основі декількох останніх елементів, з якими взаємодіяв користувач, подібно до підходів у обробці природної мови (NLP);
2) рекурентні нейронні мережі (RNN): нейронні мережі, які обробляють послідовні дані, фіксуючи зв’язки між попередніми та поточними

взаємодіями:

– LSTM (довгокороткочасна пам’ять): тип RNN, який зберігає довгострокові залежності, що ідеально підходить для аналізу часових послідовностей;
– GRU (згорнуті рекурентні блоки): більш проста та швидка альтернатива LSTM, яка також ефективно фіксує довгострокові залежності;

3) моделі на основі трансформаторів: використовують механізми самоконтролю для аналізу залежностей у послідовності взаємодій, навіть у випадках із довгими послідовностями:
   – BERT: трансформаторна модель, що враховує шаблони минулих і майбутніх взаємодій для передбачення наступного елемента.

Як було зазначено, моделі глибокого навчання можуть використовувати додаткову інформацію. Були проведені дослідження використання часу як додаткової інформації [4]. Сезонність, добові цикли та інші часові закономірності впливають на поведінку користувачів і є важливими для створення точних рекомендацій. Наприклад, користувачі можуть частіше дивитися дитячий контент у другій половині дня або ж фільми жахів під час Хелловіну. Такі закономірності складно врахувати за допомогою лінійних моделей або традиційних методів, що працюють із дискретизованими часовими інтервалами, тим не менш, використання моделей глибокого навчання та часу показало значний приріст у точності рекомендацій (див. рис. 2.14).

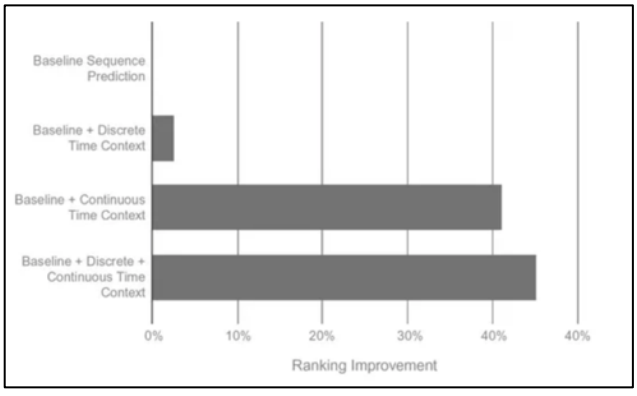


Рисунок 2.14 – Покращення ранжирування під час додавання часу як контексту (за даними [4])

2.7 Рекомендаційна система Spotify

Подібно до Netflix, Spotify використовує гібридну модель рекомендацій [28]. Spotify також використовує різні методи для різних частин рекомендацій системи, в залежності від мети рекомендації та наявних даних (наприклад, алгоритм Your Time Capsule знаходить музикальну композицію, які користувач любив, але давно не слухав. Discover Weekly використовує дані про спорідненість і схожість, щоб запропонувати схожі композиції, які користувач ще не чув. Your Daily Mix кластеризує вподобання користувача в групи й доповнює їх подібними композиціями). Система ставить за важливе подолати проблему холодного запуску під час роботи з щойно завантаженими композиціями, тому використовує дані, згенерованих обома методами, що дозволяє отримати цілісне уявлення про вміст платформи.

2.7.1 Фільтрація на основі вмісту

Щойно Spotify отримає новий елемент (пісню), алгоритм аналізує всі загальні метадані автора (в ідеальному сценарії всі метадані заповнено, цей список має містити ім'я виконавця і його рідне місце, дату випуску, теги жанру та піджанру, теги музичної культури, теги настрою, теги стилю, інструменти, що використовуються під час запису, типологія композиції тощо).

Потім відбувається аналіз необробленого аудіо пісні, алгоритм якого не розголошується Spotify. Композиція описується як мінімум 12-ю звуковими показниками, що пов’язані з об’єктивними звуковими описами: наприклад метрика «інструментальності» відображає впевненість алгоритму в тому, що композиція не містить вокалу, оцінюється за шкалою від 0 до 1 та принаймні трьома сприйнятливими, високоякісними функції рівня, призначені для більш цілісного відображення звучання композиції: танцвальність, енергія, музична позитивність.

Система також розділяє композицію на сегменти (куплети, приспіви, соло) для більш детального аналізу (див. рис. 2.15).

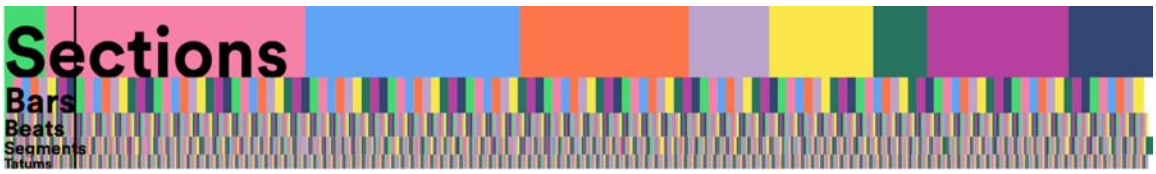


Рисунок 2.15 – Аналіз аудіо для Lil Nas X – Industry Baby feat. Jack Harlow (за даними [28])

На цьому етапі алгоритм створює опис до пісні типу: «Ця пісня дотримується структури VCVCBVC, накопичує енергію до приспіву та містить агресивне, дисонансне гітарне соло, яке перетворюється на більш меланхолійне та спокійне аутро».

Третім кроком, За допомогою NLP алгоритми аналізують тексти пісень, відгуки в блогах, опис плейлистів та інші текстові дані, щоб додати соціальний вимір до аудіо аналізу.

2.7.1 Колаборативна фільтрація

Ці рекомендації базується на поведінці користувачів із подібними музичними вподобаннями. Наприклад, якщо користувач А і користувач В слухають схожі пісні, Spotify рекомендує користувачу А композиції, які слухав користувач В, але ще не слухав А. Основний алгоритм – це факоризація матриці (наприклад, SVD), який дозволяє виявляти приховані шаблони у вподобаннях користувачів.

Spotify також, подібно Netflix, використовує моделі глибокого навчання для визначення залежності рекомендацій від контексту, наприклад, часу доби, географічного розташування чи активності користувача (прогулянка, тренування тощо).

2.8. Проблема довгострокового задоволення користувача

Дослідження Netflix визначають, що рекомендаційні системах часто використовуються короткострокові метрики, такі як кліки чи перегляди, які легко виміряти. Однак, головна мета, довгострокове задоволення користувача, є суб’єктивною і складною для оцінки. Якщо модель глибокого навчання надто

оптимізується під ці короткострокові метрики, вона може втратити зв’язок із довгостроковими цілями [4]. Також з’являється поняття як «шум короткострокових дій»: короткостроковий зворотній зв’язок, що може бути дуже випадковими, і навіть незначним в короткостроковій перспективі, в довгостроковій може суттєво змінити рекомендації.

Дослідження Spotify визначають, що основним викликом є те, що довгострокові метрики, такі як утримання користувачів чи частота повторного використання платформи, мають відкладений зворотний зв’язок [5] [10].

Це все ускладнює створення зв’язку між короткостроковими проксі-метриками та довгостроковими результатами.

2.8.1 Вирішення Netflix: контекстні бандити та проксі-нагороди Netflix

Netflix активно використовує методи контекстних бандитів і проксі-нагород для оптимізації рекомендаційних систем, орієнтованих на довгострокове задоволення користувачів.

Існує класична структура навчання з підкріпленням – багаторукий бандит, у якій агент повинен обирати між кількома варіантами дій (або «руками»), щоб максимізувати загальну винагороду з часом. Кожна рука представляє різну дію або вибір, і кожне «потягування руки» дає певну винагороду (див. рис. 2.16). Винагороди невідомі та можуть змінюватися, оскільки агент не знає розподілу даних для винагород.

Рисунок 2.16 – Покращення ранжирування під час додавання часу як контексту (рисунок виконано самостійно)

Модель багаторукого бандиту представлена тройним кортеджем *(A,R,Q)*, де агент (у нашому випадку це буде рекомендаційна система) має постійно обирати дію $\alpha \in A$ протягом $T \in N$ спроб із набору дій $A(|A|=K)$ з метою максимізувати сумарну винагороду за формулою 2.2 [29]:

$$\sum_{t=1}^{T} r_t \tag{2.2}$$

де $r_t = R_t \alpha_t$ – винагорода, отримана в результаті виконання дії α.

Алгоритм виконує дію α на кожній спробі t відповідно до політики вибору дій π. Ця політика визначається ймовірнісним розподілом, який зазвичай називають функцією цінності Q, над кожною можливою дією α.

Функція Q є основною для визначення, чи варто обирати дію α, оскільки вона вимірює очікувану винагороду за формулою 2.3:

$$Q_t(\alpha) = E[r_t|\alpha] \tag{2.3}$$

На кожному часовому кроці t агент обирає руку α_t і отримує винагороду r_t. Історія дій та отриманих винагород допомагає агенту коригувати свої вибори на наступному кроці $t+1$ і під час подальших спроб.

Контекстуальні бандити є розширенням багаторукого бандита, де кожне рішення або «потягування руки» (наприклад, рекомендація певного фільму) базується на певному контексті. Контекстом є поточна сесія або стан користувача (вподобання, історія переглядів, час доби, платформа використання, історія взаємоді тощо), а дією – набір рекомендованих елементів (наприклад, фільми або серіали) [30]. Завдяки цьому Netflix забезпечує адаптивність рекомендацій, водночас уникаючи перенавчання на короткострокових метриках.

За дослідженнями Netflix, методи контекстного бандиту є відносно простим рішенням, що також здатні розірвати цикл зворотного зв’язку та усунути різноманітні упередження у даних шляхом внесення певної кількості випадковості в рекомендації [4]. За допомогою бандитських алгоритмів можна постійно збирати точніші навчальні дані, відстежуючи схильності до показаних рекомендацій. Навіть незважаючи на те, що користувацький досвід може іноді дещо погіршуватися в короткостроковій перспективі через цю випадковість, але в у довгостроковій перспективі це допомагає покращити якість рекомендацій.

Проксі-нагороди *r(user, item)* виступають як ключовий інструмент у вимірюванні довгострокового задоволення користувачів.

Здавалося б, утримання користувачів і є мірою утримання користувачів, оскільки учасники повинні залишатися, якщо вони задоволені. Однак довгострокове задоволення важко передбачити, адже учасники можуть скасувати лише після серії поганих рекомендацій, та на процес утримання можуть впливати численні зовнішні чинники, такі як сезонні тенденції, маркетингові кампанії чи особисті обставини, не пов’язані з послугою. Такий тип реакції дуже відкладений та в більшості утримання орієнтовано на учасників, які вже збираються скасувати свою підписку, не охоплюючи повного спектру задоволеності учасників.

З цієї причини Netflix застосовує метрики, які можна отримати швидше та які корелюють із довгостроковими результатами, наприклад швидке завершення сезону серіалу, додавання фільму до списку улюблених, оцінка контенту через позитивні відгуки чи рейтинги (див. рис. 2.17).

Ці метрики дозволяють системі швидко адаптувати свої рекомендації, не чекаючи на відкладений зворотний зв'язок. Наприклад, якщо користувач завершив перегляд серіалу за короткий час, це вказує на його задоволеність, навіть якщо підписка буде продовжена через місяці.

Використання проксі-нагород у комбінації з контекстними бандитами забезпечує більш стабільне підвищення якості рекомендацій. Контрольована випадковість, властива методу бандитів, дозволяє збирати дані для навчання моделей, тоді як проксі-нагороди допомагають враховувати складність

довгострокових вподобань користувачів. Завдяки цій інтеграції Netflix створює рекомендації, які забезпечують користувачам не лише миттєву зацікавленість, але й тривалу лояльність до платформи.

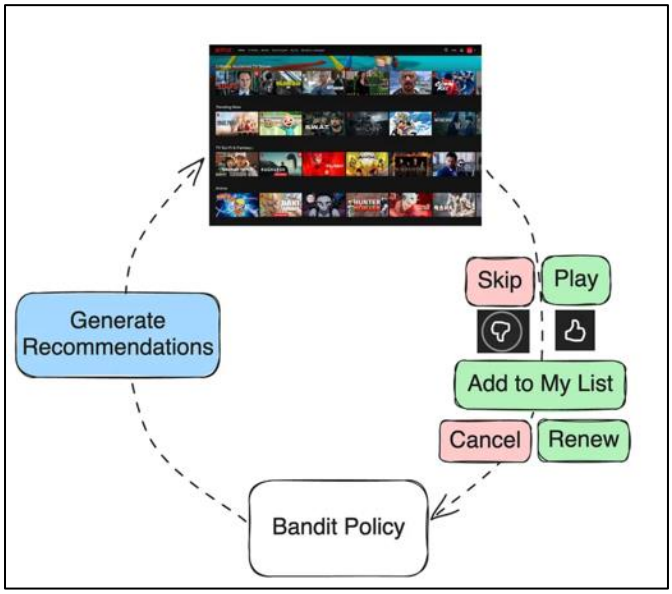


Рисунок 2.17 – Використання контекстуального бандита у Netflix (за даними [30])

Так же, як і Spotify, Netflix вирішує проблему відкладеного зворотного зв’язку (наприклад, завершення серіалу через кілька тижнів після початку перегляду) або його відсутності. Netflix прогнозує затриманий фідбек за допомогою окремих моделей машинного навчання. Так моделі прогнозування затриманого фідбеку оцінюють імовірність отримання "thumbs-up" на основі короткострокових патернів, таких як початковий час відтворення чи зупинки (див. рис. 2.18).

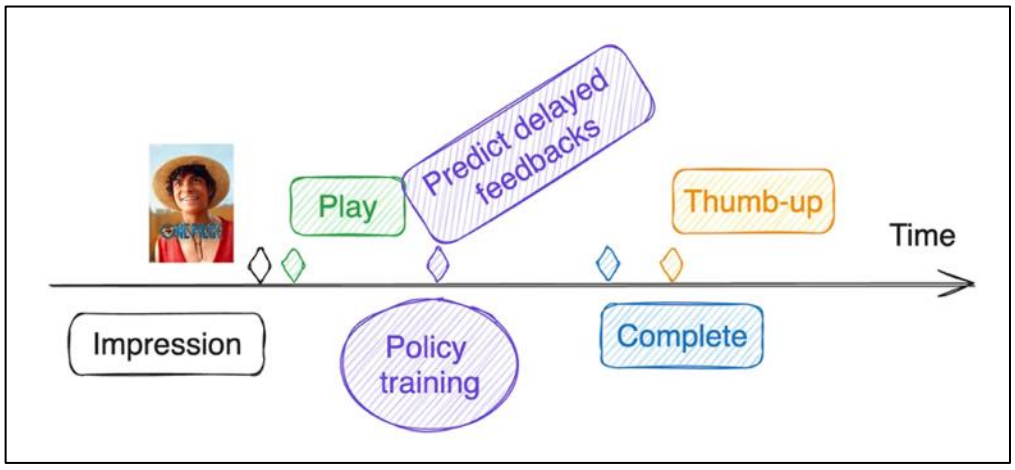


Рисунок 2.18 – Прогнозування відкладеного зворотнього зв’язку (за даними [30])

Прогнозований фідбек інтегрується у функцію проксі-нагороди, яка враховує і спостережуваний, і передбачений фідбек. Використання проксі-нагород дозволяє швидко оновлювати політику рекомендацій, навіть якщо остаточний фідбек ще не отриманий

2.8.2 Вирішення Spotify: нетерплячі бандити та навчання з підкріпленням

Spotify вирішує проблему оптимізації рекомендацій для довгострокового задоволення користувачів за допомогою інтеграції кількох підходів: моделі нетерплячих бандитів, проксі-метрик і методів навчання з підкріпленням [5] [10]. Ці компоненти працюють разом для створення системи, яка враховує як короткострокову, так і довгострокову взаємодію користувачів із платформою.

Модель нетерплячого бандиту є розширенням багаторукого бандита. У цій моделі алгоритм приймає рішення на основі: короткострокових сигналів (наприклад, пропуски композицій, збереження або завершення прослуховування), прогнозування довгострокового впливу (алгоритм оцінює, як дії вплинуть на поведінку користувача в майбутньому, зокрема, на ймовірність повернення до функції або регулярність використання). Модель забезпечує динамічне ухвалення рішень у реальному часі, враховуючи короткострокові сигнали.

Проксі-метрики виступають інструментом для вимірювання задоволення користувачів у короткостроковій перспективі, яке корелює із довгостроковими цілями. Наприклад, у Discover Weekly задоволення користувача оцінюється за такими діями, як регулярне використання функції, додавання композицій до бібліотеки або відсутність пропусків, а для Release Radar проксі-метрики включають частоту повернення до функції та кількість збережених композицій (див. рис. 2.19). Використовує моделі, як дерева рішень для прогнозування проксі-нагород на основі короткострокової взаємодії з композиціями, історичних даних і кластеризації цілей користувачів.

Навчання з підкріпленням інтегрує проксі-метрики та модель бандитів у єдиний підхід. Алгоритм отримує винагороди за дії, які збільшують імовірність досягнення довгострокових цілей задоволення користувачів.

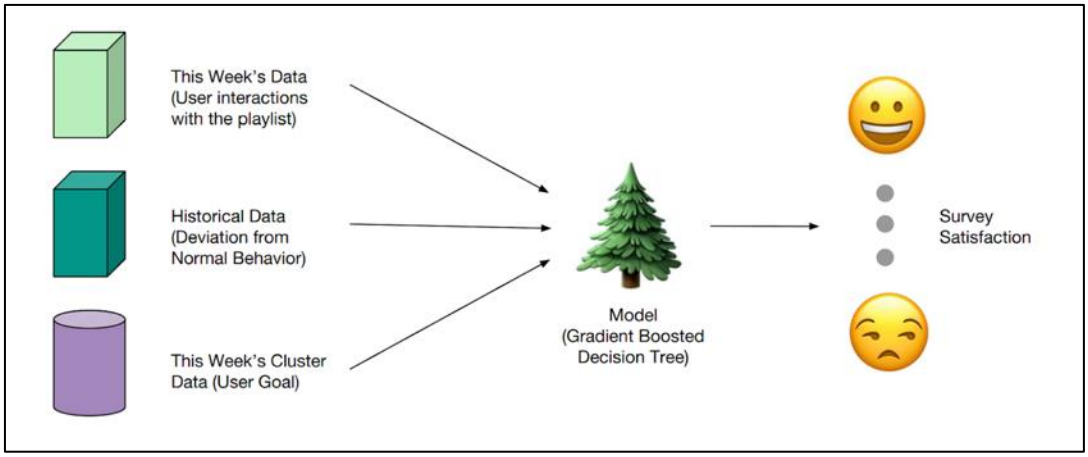


Рисунок 2.19 – Прогнозування короткострокових метрик чи проксі-нагород (за даними [28])

Алгоритм навчається на історичних даних взаємодії користувачів із платформою, прогнозуючи, які дії принесуть найбільшу цінність у майбутньому та дозволяє моделювати складну поведінку користувачів, об'єднуючи короткострокові дії та довгострокові цілі.

2.9 Висновки з огляду

Аналіз літератури показав, що область рекомендаційний систем являється актуальною та тенденційною областю дослідження. Рекомендаційні системи незамінно інтегруються у великому спектрі сучасних інформаційних систем, вирішуючи проблему інформаційної завантаженості користувачів цих систем.

Не вдивлячись на ступінь їх дослідженності та розвитку, все ще наявні виклики, обумовлені тенденціями росту області. Один з них – це розуміння та врахування довгострокового задоволення користувачів.

Проведений огляд підходів Netflix та Spotify до створення рекомендаційних систем для довгострокового задоволення користувачів показав, що обидві компанії використовують складні, багатокомпонентні стратегії, які орієнтовані на вирішення ключових викликів у роботі систем персоналізації контенту.

Попри відмінності у підходах, обидві компанії активно використовують проксі-нагороди та різні види бандитів для швидкої оцінки впливу своїх рекомендацій, долаючи обмеження традиційних метрик, що базуються лише на

короткострокових діях. Крім того, адаптивність моделей машинного навчання та врахування контексту дозволяють обом платформам забезпечувати персоналізований і релевантний досвід для своїх користувачів.

Таким чином, аналіз доводить, що для успішного вирішення проблеми довгострокового задоволення користувачів важливо інтегрувати проксі-нагороди та моделі бандитів і використовувати сучасні методи машинного навчання, що дозволить забезпечувати адаптацію системи до змін у поведінці користувачів.

3 ПОСТАНОВКА ЗАДАЧІ

3.1. Мета та очікувані результати

Мета практичного дослідження полягає в створенні рекомендаційної системи, яка використовує проксі-нагороди для оцінки короткострокової взаємодії та симуляції довгострокового задоволення, перевірці, як вибраний підхід впливає на якість рекомендацій у довгостроковій перспективі.

Результатом роботи стане рекомендаційна система, яка:

- пропонує користувачу первинні рекомендації шляхом фільтрації на основі вмісту;
- навчається на деякому наборі даних, який містить матрицю взаємодії користувачів з деякими елементами. На основі колаборативної фільтрації пропонує рекомендації та враховує довгострокові вподобання користувачів за допомогою використання проксі-нагород;
- демонструє можливість застосування методів моделей бандитів;
- підтверджує ефективність розробленого підходу шляхом експериментального тестування на публічно доступних наборах даних.

3.2 Обґрунтування вибору методів

Вибір методів дослідження обґрунтований сучасними викликами, що стоять перед рекомендаційними системами. Традиційні підходи, орієнтовані на короткострокові метрики (наприклад, кліки або перегляди), не дозволяють врахувати більш складні та відкладені аспекти задоволення користувачів. Методи проксі-нагород та контекстуальних бандитів обрані через їх здатність враховувати відкладений зворотний зв’язок, адаптуватися до динаміки вподобань користувачів та оптимізувати рекомендації у довгостроковій перспективі.

3.3. Специфіка обмежень і викликів

В процесі реалізації проєкту очікується подолання наступних викликів:

- недостатність даних чи холодний старт: початок взаємодії нових користувачів з будь-якою платформою, як правило, супроводжується

недостатньою кількістю інформації для формування рекомендацій для них. Саме через це деякі платформи при першому їх використанні новими користувачами, щоб сформувати початковий профіль користувача, запитують деяку базову інформацію у користувача, наприклад вік, інтереси, локація (див. рис. 3.1). У рамках розроблювальної системи будуть запитуватися 3 улюблені жанри користувача.

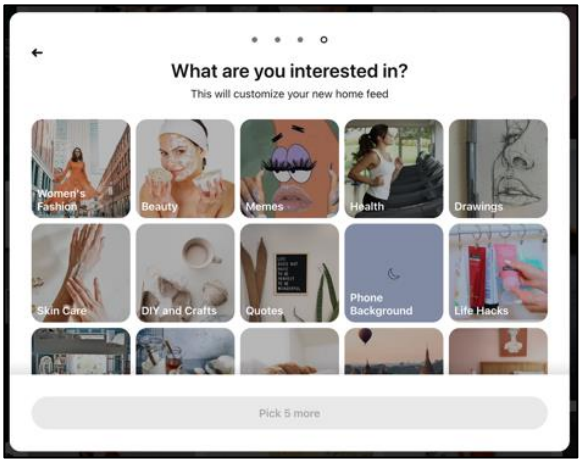


Рисунок 3.1 – Збір інтересів нового користувача у Pinterest (скріншот з сайту Pinterest[3])

– проблема невизначеності довгострокової задоволеності користувачів: визначення довгострокової задоволеності не є конкретним. Натомість необхідне використання короткострокових метрик та проксі-нагород для прогнозування довгострокового задоволення користувача;
– проблема затриманого зворотного зв’язку: метрики, які свідчать про довгострокове задоволення (наприклад, повернення до контенту або повторна передплата), зазвичай відомі з великим запізненням.

Можливі виклики включають:

– розрідженість даних: Не всі користувачі залишають відгуки або демонструють явні вподобання, що ускладнює навчання моделей;

[3] Pinterest. (n.d.). Pinterest. https://pinterest.com/

– масштабованість: використання сучасних алгоритмів може вимагати значних обчислювальних ресурсів, особливо при роботі з великими наборами даних;
– складність реалізації: дослідження являється експериментальною розробкою, що спирається на наявні дослідження. Проблема довгострокового задоволення користувача є новою, актуальною та не має великої кількості наявних для розгляду рішень, що потребує уважного вивчення наявних джерел та особистої розробки.

3.4. Інструменти та ресурси

Для реалізації поставлених завдань будуть використані такі інструменти:
– набір даних: MovieLens [31], що надає збагачену інформацію для моделювання рекомендаційної системи (детальніше розглянуто в розділі 5.1);
– технології: Python як основна мова програмування (написання бекенду з використанням Flask), бібліотеки для аналізу даних (такі як Pandas, NumPy), бібліотеки для машинного навчання (такі як scikit-learn), сховище даних SQLLite;
– метрики оцінки: Метрики точності для оцінки якості рекомендацій, а також проксі-метрики для довгострокового задоволення, такі як прогнозування повторних дій користувача.

3.5. Етапи реалізації

Проєкт буде реалізовано в кілька етапів:
а) підготовчий етап:
   1) визначення вимог до системи та формулювання основних цілей дослідження;
   2) аналіз доступних наборів даних (MovieLens, Netflix Prize Dataset) і вибір найбільш підходящого для моделювання;
б) розробка первинних рекомендацій: реалізація алгоритму рекомендацій на

основі вмісту, коли користувач вказує лише первинні вподобання в жанрах фільмів;

в) розробка рекомендаційної системи: реалізація алгоритму рекомендацій на основі колаборативної фільтрації для створення початкової моделі;

г) інтеграція сучасних методів для довгострокових рекомендацій: інтеграція проксі-нагород для оптимізації довгострокового задоволення користувачів з використанням контекстуальних бандитів;

д) тестування та оцінка: проведення експериментів для оцінки ефективності системи за на тестовій виборці;

е) аналіз результатів: інтерпретація отриманих даних і формулювання висновків щодо доцільності застосування обраного підходу;

ж) підготовка звіту: формування повного звіту про виконану роботу, включаючи рекомендації щодо практичного впровадження системи в різних галузях.

4 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

4.1 Вибір набору даних

4.1.1 Опис варіантів наборів даних

Можливості такого дослідження тісно пов’язані та навіть обмежені наявними наборами даних. Набір даних має надавати достатньо матеріалів для навчання та тестування впроваджених моделей.

Так як у даному звіті розглядалися дві системи: Netflix та Spotify, були обрані два набори даних, які пропонувалися ними для відкритих конкурсів вирішення проблем рекомендацій:

– Netflix Prize Dataset: був розроблений для конкурсу Netflix Prize і містить набір рейтингів у понад 100 мільйонів фільмів від сотень тисяч користувачів. Основна мета цього набору – стимулювати розвиток моделей для прогнозування рейтингу, що користувач може поставити фільму;

– Spotify Million Playlist Dataset: розроблений в рамках конкурсу RecSys Challenge 2018, складається з набору даних мільйону плейлістів, створених користувачами у період з січня 2010 року по жовтень 2017 року. Був націлений на вирішення проблеми продовження плейліста, створеного користувачем, автоматично запропонованими композиціями.

Також були розглянуті два інших популярних в даній предметній області набори даних, що стосуються фільмів і музики:

– MovieLens: один із найбільш популярних наборів даних для навчання та тестування рекомендаційних систем. Він має кілька версій: від малого (100,000 рейтингів) до великого (20 мільйонів рейтингів). Він застосовувався у широкому спектрі досліджень;

– Last.fm Dataset: створює детальний профіль музичних смаків користувача, записуючи деталі композицій, які користувач слухає, надає інформацію про взаємодії користувачів із більше ніж 100 тисячами композицій.

4.1.2 Порівняльний аналіз

Вибір набору даних був заснований на порівнянні, наведеному у таблиці (див. табл. 4.1).

Таблиця 4.1 – Порівнянна наборів даних (таблиця виконана самостійно)

Набір даних	Можливості	Обмеження
Netflix Prize Dataset	– Містить інформацію про фільми, а саме ідентифікатор, назву, рік випуску (movie_titles.csv); – містить матрицю взаємодії користувача з фільмами з рейтингом та часом оцінки (чотири combined_data.txt файли); – містить файл для передбачення фільмів для користувачів (qualifying.txt); – дозволяє тестувати масштабовані моделі рекомендацій та підходить для прогнозування рейтингів.	– Немає метаданих про користувачів або фільми; – відсутність контекстуальних факторів, таких як тип пристрою, точний час.
Movie Lens	– містить матрицю взаємодії користувача з фільмами з рейтингом та часом оцінки (rating.csv); – містить теги, застосовані до фільмів користувачами та час оцінки (tag.csv); – містить назви та жанри про фільми (movie.csv); – містить ідентифікатори для The Movie Database (TMDB) [31] та Internet Movie Database (IMDB) [32] баз даних, звідки можна взяти додаткові дані про фільм.	Відсутність контекстуальних факторів, таких як тип пристрою, точний час.

Кінець таблиці 4.1

Spotify Million Playlist Dataset	– містить дані про плейлісти: назву, список композицій (включно з ідентифікаторами композицій і метаданими) та інші поля метаданих (час останнього редагування, кількість редагувань списку відтворення тощо); – підходить для складних моделей з різними типами даних; – забезпечує можливість аналізувати контекстні дані.	Не має матриці взаємодії – більше націлений на завдання продовження плейлісту, аніж створення рекомендацій.
Last.fm	– Включає дані про прослуховування музики з метаданими як ім’я користувача, виконавець, назва пісня, альбом, дата прослуховування, час прослуховування. Ці дані включають не тільки першу, а і повторну взаємодію користувача з композицією (див. рис. 4.1).	– Відсутність мультимодальних характеристик, таких як тексти чи зображення; – матриця взаємодії має бути побудована вручну; – даних про повторну взаємодію користувача з композицією не так багато (див. рис. 4.1).

Бачимо, Netflix дозволяє виконувати аналіз поведінки користувачів у динаміці (наприклад, зміну вподобань з часом), надаючи об’ємний набір даних, але

відсутність інформації про жанри фільмів або інші характеристики контенту ускладнює розробку рекомендацій на основі змісту.

```
SELECT
    SUM(CASE WHEN times = 1 THEN 1 ELSE 0 END) AS one,
    SUM(CASE WHEN times > 10 THEN 1 ELSE 0 END) AS moreThan10,
    SUM(CASE WHEN times > 20 THEN 1 ELSE 0 END) AS moreThan20,
    SUM(CASE WHEN times > 30 THEN 1 ELSE 0 END) AS moreThan30,
    SUM(CASE WHEN times > 40 THEN 1 ELSE 0 END) AS moreThan40,
    SUM(CASE WHEN times > 50 THEN 1 ELSE 0 END) AS moreThan50,
    SUM(CASE WHEN times > 60 THEN 1 ELSE 0 END) AS moreThan60
FROM (
    SELECT column1, column2, column3, COUNT(*) AS times
    FROM data
    GROUP BY column1, column2, column3
) grouped_data;
```

	one	moreThan10	moreThan20	moreThan30	moreThan40	moreThan50
1	"113318"	"212"	"53"	"31"	"10"	"0"

Рисунок 4.1 – Повторне прослуховування композицій у наборі даних Last.fm (рисунок виконано самостійно)

MovieLens дозволяє будувати гібридні моделі, роблячи і фільтрацію на основі вмісту й аналізуючи елементи фільмів, і колаборативну фільтрацію. Часові позначки у взаємодіях можуть бути використані для аналізу довгострокових вподобань.

Spotify Million Playlist Dataset має багато контекстної інформації для рекомендацій на основі вмісту композицій, але матриця взаємодії не може бути створена з даного набору даних, що не реалізує потенціал гібридної фільтрації та аналізів дати взаємодії користувача з композицією, кількістю прослуховувань тощо. У ході аналізу визначено, що вирішує дуже вузьку проблему обраної області.

У Last.fm відсутність текстів пісень, аудіо-фічерів або зображень (наприклад, обкладинок альбомів) зменшує можливість використовувати складні моделі на основі вмісту. Інформація часу прослуховування є дуже цінною та цікавою для прогнозування часу наступного прослуховування цієї або схожої пісні користувачем. Таких записів не багато 212 записів на прослуховування більше одного разу на більше ніж 113 тисяч одного прослуховування).

Жоден набір даних не містить інформації про самих користувачів (наприклад, вік, регіон), що обмежує персоналізацію.

4.1.3 Результат аналізу

Для даного дослідження рекомендацій для довгострокового задоволення було обрано використовувати набір даних MovieLens, тому що він містить метадані для побудови гібридних моделей, підтримує аналіз довгострокових уподобань через часові позначки та інтегрується з іншими джерелами даних, що розширює можливості аналізу.

4.2 Моделювання рекомендацій

4.2.1 Первинні рекомендації

Перша частина практичної реалізації дослідження складається з реалізації початкових рекомендацій на основі набору даних. Проблема, яка присутня на даному етапі – це наявність мінімальної інформації про користувача, тобто система стикається з проблемою холодного старту. Для реалізації рекомендацій у цьому контексті було обрано використовувати фільтрацію на основі вмісту.

Як було вказано раніше, фільтрація на основі вмісту аналізує поведінку користувача та рекомендує елементи, подібні до нього, на основі врахованих параметрів. Параметри, які ми можемо запросити у користувача на первинному етапі – це його улюблені жанри.

Елементи (фільми) перетворюються на вектори за допомогою описів метаданих або внутрішніх характеристик як ознак [33]. У випадку цього дослідження кожен жанр розглядається як окрема бінарна ознака (feature).

Для кожного твору формується вектор жанрів, де значення 1 означає належність до жанру, а 0 – відсутність. Усі вектори творів розташовуються в n-вимірному просторі, де n – це кількість всіх жанрів.

Ближчість двох векторів відповідає ступеню схожості творів за жанровими характеристиками. Для вимірювання відстані (ступеня схожості) між двома жанровими векторами застосовуються [33]:

– косинусна подібність: означає вимірювання кута між двома векторами. Це може бути будь-яке значення від -1 до 1. Чим вищий косинусний бал, тим більше схожими вважаються два елементи;

– евклідова відстань: вимірює довжину гіпотетичного відрізка прямої, що з'єднує дві точки вектора. Значення евклідової відстані можуть бути такими низькими, як нуль, без верхньої межі. Чим менша евклідова відстань двох векторів елементів, тим більш схожими вони вважаються;
– скалярний добуток: добуток косинуса кута між двома векторами та відповідної евклідової величини кожного вектора відносно визначеного початку координат.

Вважається, що скалярний добуток найкраще використовувати для порівняння елементів із помітно різною величиною, наприклад, популярності книг чи фільмів [33], тому для реалізації було обрано саме його. Він розраховується за формулою 4.1

$$DotSim(\overline{D}, \overline{Q},) = \sum_{j=1}^{n} d_j \, q_j \tag{4.1}$$

де d та q – два вектори.

Далі профіль користувача формується, як це сукупність векторів тих елементів, якими він цікавився. На основі цих векторів будується агрегований вектор уподобань (наприклад, середнє значення або зважена сума).

Алгоритм порівнює вектор профілю користувача із векторами нових (непереглянутих) елементів. Ті фільми, вектори яких лежать найближче до вектора користувачевого профілю за обраною метрикою, рекомендуються користувачу як найбільш релевантні. Алгоритмом також було обрано використовувати скалярний добуток.

Користувач буде використовувати систему для введення реєстрації, та вказування трьох улюблених жанрів, на основі який відбуватиметься первинна рекомендація. Даний процес ілюстрований на діаграмі послідовності рисунку 4.2.

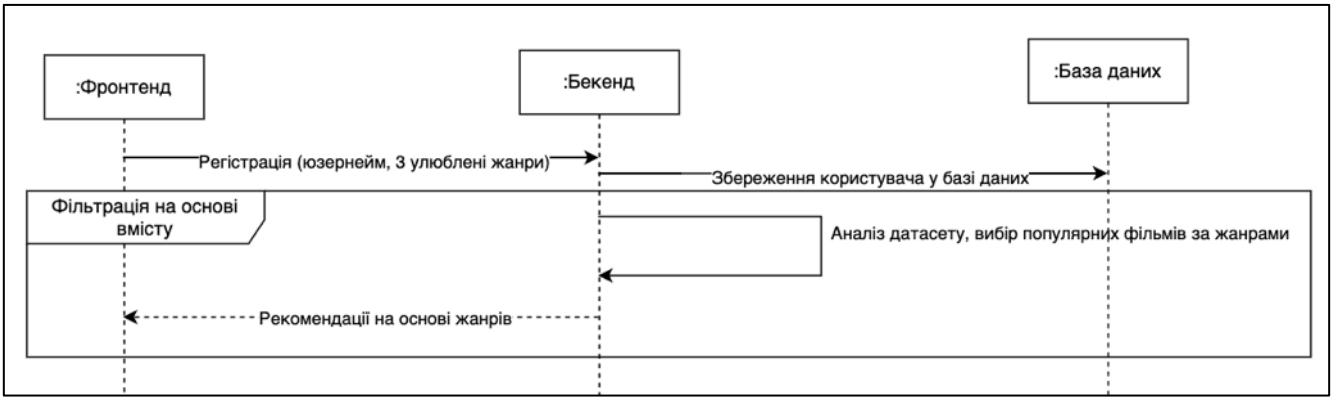


Рисунок 4.2 – Діаграма послідовності отримання первинних рекомендацій (рисунок виконано самостійно)

4.2.2 Основна рекомендаційна модель

Надалі необхідно розробити основну рекомендаційну модель, яка буде пропонувати користувачу фільми. Так як первинні рекомендації базуються на жанрах, реалізовуючи фільтрацію на основі вмісту, нехай друга модель буде базуватись на колаборативній фільтрації.

Колаборативна фільтрація, як було оглянуто в Розділі 2, базується на гіпотезі: "Користувачі, які мали схожі вподобання в минулому, матимуть схожі вподобання в майбутньому". Для цього необхідно сформувати матрицю взаємодії R розмірністю m×n, де m та n – кількість користувачів та фільмів відповідно, джерелом якої слугуватиме rating.csv, який включає оцінки (рейтенги). $R_{u,i}$ в такій матриці представлятиме собою оцінку. Використовуючи відомі частини матриці R, відбувається прогнозування невідомих значень $\hat{r}_{u,j}$ для кожного u і j, які ще не пов’язані в базі.

Основна характеристика наявних даних – це їх розрідженість. Матриця рейтингів R у реальних системах зазвичай є надзвичайно розрідженою (sparse), оскільки більшість користувачів оцінює лише невелику частку доступних товарів. У контексті MovieLens 20M це означає, що з понад 27,000 фільмів та 138,000 користувачів, заповненими є менше ніж 1% усіх можливих комбінацій користувач-фільм. Математично, щільність матриці рейтингів ρ можна описати за формулою 4.2.

$$\rho = \frac{|\Omega|}{(m \times n)} \quad (4.2)$$

де $|\Omega|$ – кількість відомих рейтингів,

m – кількість користувачів,

n – кількість фільмів.

Для типових рекомендаційних систем $\rho < 0.01$ [34]. Щільність дійсно великої рекомендованої системи, такої як Netflix, набагато менша та становить приблизно 0,02 відсотка, тоді як у Movie Lens вона становить лише 0,005 відсотка, що створює такі проблеми як:

– недостатність даних для обчислення схожості: нетрадиційні методи (cosine similarity, Pearson correlation) працюють неефективно при малій кількості спільно оцінених елементів
– проблема холодного старту: нові користувачі або товари не мають достатньо взаємодій для генерації рекомендацій
– обчислювальна неефективність: прямі обчислення на розріджених матрицях є ресурсоємними.

Для вирішення проблеми розрідженості застосовується підхід на основі моделі, зокрема матричної факторизації через SVD [35]. Основна ідея полягає у тому, що розріджену матрицю R можна апроксимувати як добуток трьох матриць меншої розмірності, як представлено формулою 4.3 [36].

$$R = U\Sigma V^T \quad (4.3)$$

де A – вхідна матриця (матриця рейтингів користувач-фільм),

U – ліві сингулярні вектори (матриця користувачів у латентному просторі),

Σ – діагональна матриця сингулярних значень/власних значень,

V – праві сингулярні вектори (матриця фільмів у латентному просторі).

Кожен латентний фактор представляє прихований аспект, який впливає на вподобання користувачів. У контексті доступної про фільми інформації це можуть бути жанрові характеристики. Кожного користувача та кожен фільм можна подати у вигляді вектора в прихованому (латентному) просторі. Замість того, щоб працювати з великою та розрідженою матрицею оцінок, система навчається будувати компактні «портрети» користувачів та фільмів, які не задаються вручну, а виявляються автоматично на основі вже наявних оцінок.

Для кожного користувача алгоритм виводить набір чисел, що умовно можна інтерпретувати як його схильність до певних типів жанрів фільмів, а для кожного фільму відповідні характеристики. Після цього прогноз оцінки здійснюється як поєднання цих двох векторів. Чим більше вони співпадають, тим вища буде ймовірна оцінка. Класичний SVD не може бути безпосередньо застосований до розріджених матриць, оскільки потребує повністю заповнених даних, тому у рекомендованих системах натомість використовується модифікований стохастичний варіант SVD, який навчається лише на відомих оцінках, поступово оптимізуючи вектори користувачів та фільмів.

Процес аналізу:

– створюємо матрицю користувачів × фільмів із рейтингами;
– використовуємо SVD для представлення кожного користувача та фільм у вигляді векторів у спільному латентному просторі;
– визначаємо схожість користувачів (косинусна схожість між векторами латентних факторів). Для кожного користувача обчислюються ймовірні оцінки тих фільмів, які він ще не переглядав. Прогноз ґрунтується на схожості між векторами користувача та фільму;
– виконуємо сортування, фільтрацію, видаляємо з рекомендацій фільми, які користувач вже дивився, пріоритизуємо нові релізи, щоб користувачі мали актуальні пропозиції та обираємо N кількість для показу.

4.2.3 Довгострокова задоволеність

Аналіз досвіду Netflix та Spotify продемонстрував потужність проксі-нагород та контекстних бандитів для вирішення проблеми довгострокової задоволеності користувача. Впровадження цього підходу в контексті даного дослідження потребує інтеграції з існуючими жанровими та колаборативними алгоритмами. Вже розроблена система адаптується тільки на рейтинги. Але якщо користувач не закінчив перегляд фільму, або не порахував важливим лишити фідбек, можливість надати користувачу рекомендації повністю відпадає.

Ідея з використанням проксі нагород полягає в логуванні дій користувача. Для спрощення системи явні та неявні проксі-нагороди буде зображено як кнопки інтерфейсу:

а) явні:
   1) додати у подивитись пізніше: чи користувач додав фільм у збережений список;
   2) рейтинг, як безпосередній показник задоволення від перегляду фільму;

б) неявні:
   1) чи почав користувач дивитися фільм;
   2) чи завершив дивитися.

Цей список, звісно, можна продовжувати різноманітними діями (повторний перегляд, відкрити фільм в стрічці рекомендацій тощо), але в рамкаї цього дослідження, обмежимось вище наведеними діями. Значення нагород буде встановлено емпірично.

Після кожної взаємодії з фільмом (оцінка, перегляд) відбуватиметься обчислення проксі-нагороди. Ці нагороди слугуватимуть основою для моделі контекстуального бандиту, який узгоджуються з принципами навчання з підкріпленням [37].

Як вже було зазначено впродовж літературного огляду, контекстуальний бандит обирає дію, з якої передбачувано може отримати найкращий результат. Руки багаторукого контекстуального бандиту представляють можливі варіанти дій, які може обрати алгоритм. У класичній постановці задачі кожен фільм міг би бути

окремою "рукою", але така архітектура призводить до серйозних проблем масштабованості. При наявності тисяч фільмів система мала б підтримувати тисячі окремих моделей, що тим паче неможливо у випадку цього дослідження через обмеження у ресурсах.

Тому вирішенням буде розроблення єдиного контекстуального підходу, де замість окремих "рук" для кожного фільму використовується одна універсальна модель, що оперує контекстними векторами. Це означає, що кожен фільм представляється через свій контекстний вектор. Базуючись на наявних даних у цей вектор можуть входити такі дані як жанрова приналежність фільму, рейтингові характеристики, часові ознаки, персоналізовані метрики відповідності уподобанням користувача та показники популярності й якості фільму. Алгоритм навчатиметься передбачати нагороду на основі цього контексту, а не ідентифікатора фільму.

4.3 Функціональні вимоги системи

Взаємодія користувача наведена на Use-case діаграмі (див. рис. 4.3).

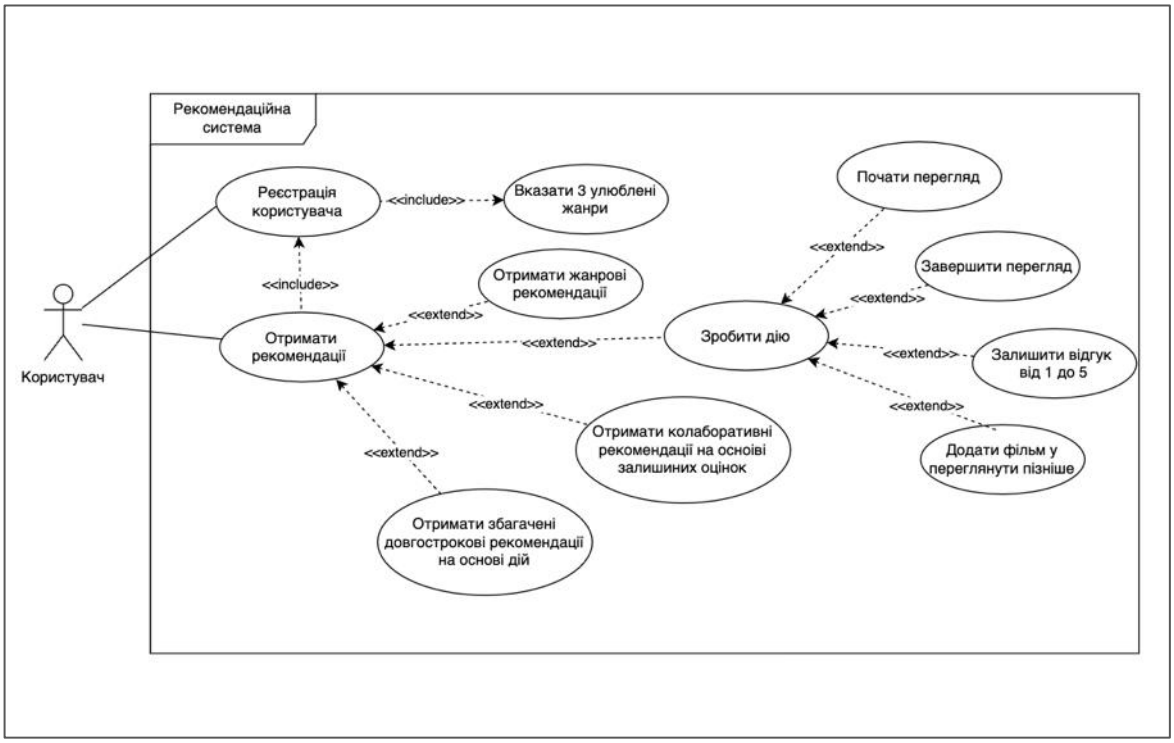


Рисунок 4.3 – Use-case діагарама системи (рисунок виконано самостійно)

Основною дією є вхід у систему, після чого користувач отримує персоналізований список рекомендацій.

Дії, на кшталт відкрити інформацію про окремі фільми, зберегти фільм до списку обраного для перегляду в майбутньому, переглянути фільм, завершити фільм, поставити відгук – це можливості, як користувач може взаємодіяти з фільмом, що буде основою для проксі-винагород.

4.4 Архітектура системи

З урахуванням описаного, архітектура системи буде виглядати наступним чином (див. рис. 4.4)

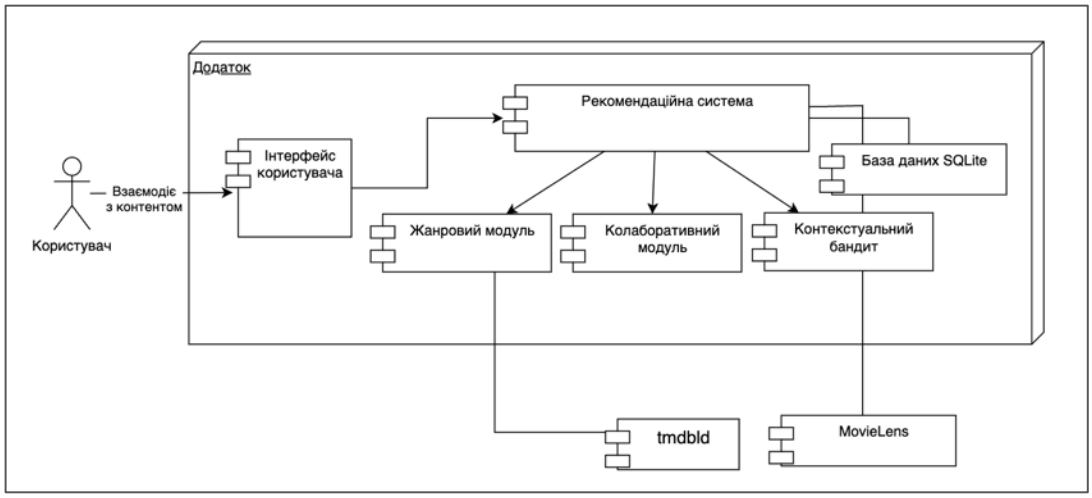


Рисунок 4.4 – Діаграма компонентів системи (рисунок виконано самостійно)

Користувач взаємодіє з додатком через інтерфейс, який дозволяє виконувати такі дії. Всі ці дії передаються у внутрішню систему, де обробляються і зберігаються для подальшого навчання у базі даних SQL Lite.

Система складається з модуля рекомендацій, який відповідає за генерацію початкових рекомендацій на основі даних із набору MovieLens, співпрацює з модулем колаборативної фільтрації та модулем контекстуального бандиту.

5 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

5.1 Первинні рекомендації

Як було вказано в минулому розділі, перша частина практичної реалізації дослідження складається з реалізації початкових рекомендацій на основі 3-ох улюблених жанрів користувача. Цей етап складається з чотирьох кроків, елементи коду до кожного з кроків наведені на рисунку 5.1.

```python
# 1. Крок 1: Створення жанрової матриці фільмів (1 якщо жанр є, 0 якщо нема)
genre_matrix = pd.DataFrame(0, index=movies['movieId'], columns=genre_columns)

for idx, row in movies.iterrows():
    if pd.isna(row['genres']): continue
    for genre in row['genres'].split('|'):
        if genre in genre_columns:
            genre_matrix.at[row['movieId'], genre] = 1

# 2. Крок 2: Створення жанрового профілю користувача (за вибраними жанрами)
data = request.get_json()
username = data.get("username")

row = c.execute("SELECT genres FROM users WHERE username = ?", (username,)).fetchone()
if not row:
    return jsonify({"error": "User not found"}), 404

user_genres = row[0].split(",")

# Профіль користувача — вектор, де 1 для обраних жанрів
user_profile = pd.Series(0, index=genre_columns)
for g in user_genres:
    if g in user_profile:
        user_profile[g] = 1

# 3. Крок 3: Обчислення релевантності фільмів до профілю користувача (через скалярний добуток — скільки жанрів збігаються)
scores = genre_matrix.dot(user_profile)
scores.name = "score"

# 4. Крок 4: Об'єднання жанрової релевантності з рейтингами та фільтрація
movies_scored = movies.set_index("movieId").join(scores)
movies_scored = movies_scored.join(movie_stats.set_index("movieId"))  # mean, count

# Прибираємо непопулярні фільми
movies_scored = movies_scored[movies_scored["count"] > 50]

# Сортування за score та по рейтингу
movies_scored = movies_scored.sort_values(by=["score", "mean"], ascending=False).head(15)

result = []
for _, row in movies_scored.iterrows():
    result.append({
        "title": row["title"],
        "mean_rating": round(row["mean"], 2),
        "ratings_count": int(row["count"]),
        "score": int(row["score"])
    })
```

Рисунок 5.1 – Реалізація початкових рекомендацій на основі жанрів (рисунок виконано самостійно)

Крок 1: створення жанрової матриці фільмів. Жанри в матриці представлені у бінарному форматі. З набору з 20 наявних жанрів, якщо фільм має певний жанр, він отримує значення 1. Якщо жанру немає у фільмі, він отримує значення 0.

Крок 2: створення жанрового профілю користувача. На основі трьох представлених жанрів користувача створюється жанровий профіль у вигляді бінарного вектору, де значення 1 відповідає вибраним жанрам. Цей вектор слугує опорною точкою для порівняння з кожним фільмом у базі.

Крок 3: обчислення релевантності фільмів до профілю користувача. Використовується скалярний добуток між жанровою матрицею фільмів і жанровим профілем користувача. У результаті кожен фільм отримує числову оцінку score, що відображає кількість збігів між жанрами фільму та вподобаннями користувача. Наприклад, якщо фільм містить два з трьох жанрів, вибраних користувачем, його score дорівнює 2.

Крок 4: об’єднання жанрової релевантності з рейтингами та фільтрація. Для покращення якості рекомендацій, результати поєднуються зі статистикою рейтингу: середнім балом (mean) та кількістю оцінок (count). Це дозволяє фільтрувати маловідомі або низько оцінені фільми. У фінальний список потрапляють лише ті, що мають понад 50 оцінок, та сортуються за релевантністю і рейтингом.

Для користувача з обраними жанрами «Анімація», «Романтика», «Фантазія» (див. рис. 5.2) система обрала наступні рекомендації (див. рис. 5.3).

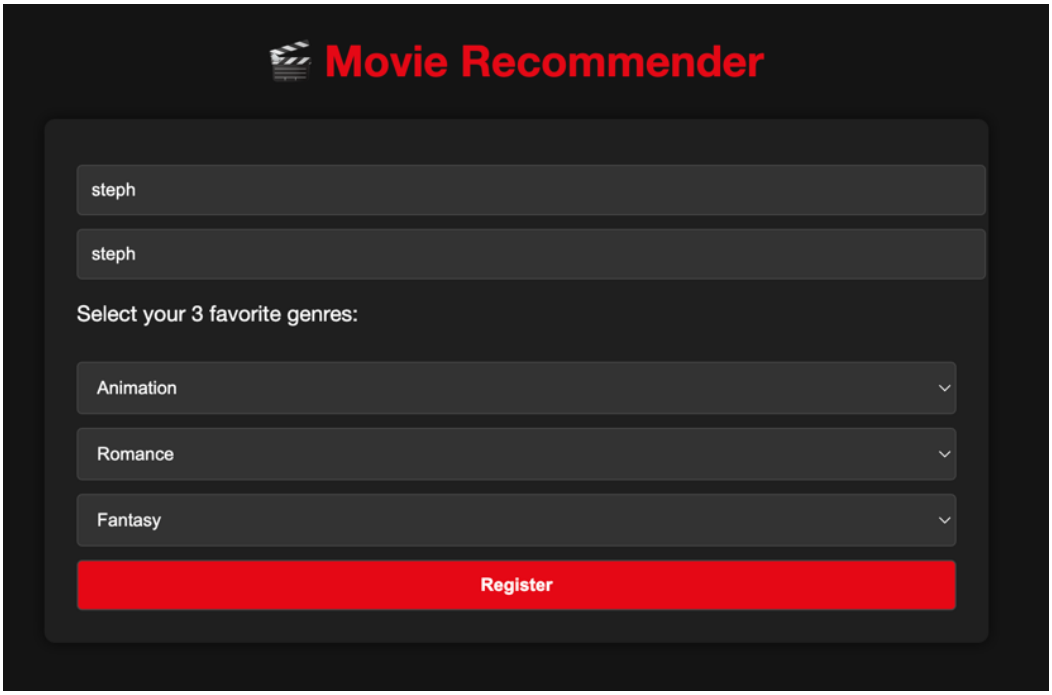


Рисунок 5.2 – Реєстрація користувача (рисунок виконано самостійно)

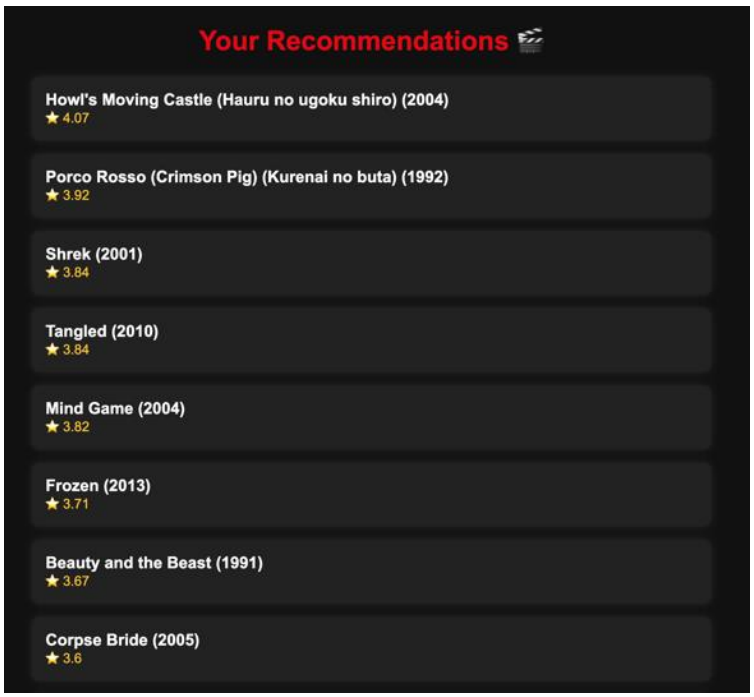


Рисунок 5.3 – Отримані рекомендації (рисунок виконано самостійно)

5.2 Основна рекомендаційна модель

5.2.1 Створення рейтингів

До інтерфейсу необхідно додамо можливість додавати рейтинг до фільму (див. рис. 5.4).

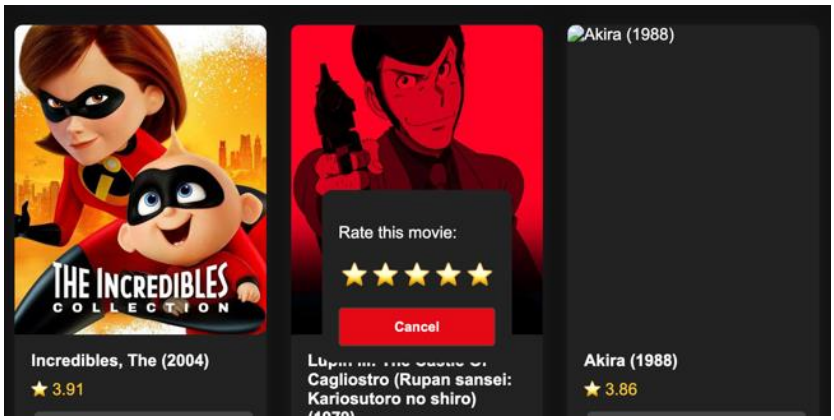


Рисунок 5.4 – Інтерфейс додавання рейтингу (рисунок виконано самостійно)

Коли користувач буде лишати рейтинг (а в подальшому і дії, важливі для довгострокової задоволеності користувача), інформація про це буде додаватись в таблицю дій. Реалізація таблиці та маршруту наведена на рисунку 5.5.

```python
def init_actions_table():
    conn = get_connection()
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS actions (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT,
        movieId INTEGER,
        action TEXT,
        rating REAL,
        timestamp TEXT
    )''')
    conn.commit()
    conn.close()

@action_bp.route('/action', methods=['POST'])
def log_action():
    data = request.json
    username = data.get("username")
    movie_id = data.get("movie_id")
    action_type = data.get("action")
    rating = data.get("rating", None)
    timestamp = datetime.datetime.now().isoformat()

    print("Received action:", data)
    try:
        conn = get_connection()
        c = conn.cursor()
        c.execute('''INSERT INTO actions (username, movieId, action, rating, timestamp)
                     VALUES (?, ?, ?, ?, ?)''',
                  (username, movie_id, action_type, rating, timestamp))
        conn.commit()
        conn.close()
        print("Action saved to DB.")
        return jsonify({"status": "ok", "message": "Action logged."})
    except Exception as e:
        print("DB Error:", e)
        return jsonify({"status": "error", "message": str(e)})
```

Рисунок 5.5 – Створення таблиці дій коритувача та логування дій користувача (рисунок виконано самостійно)

Також отримання додаткових даних про фільм зроблено шляхом інтеграції з TMDB [31]. Елемент коду представлений на рисунку 5.6.

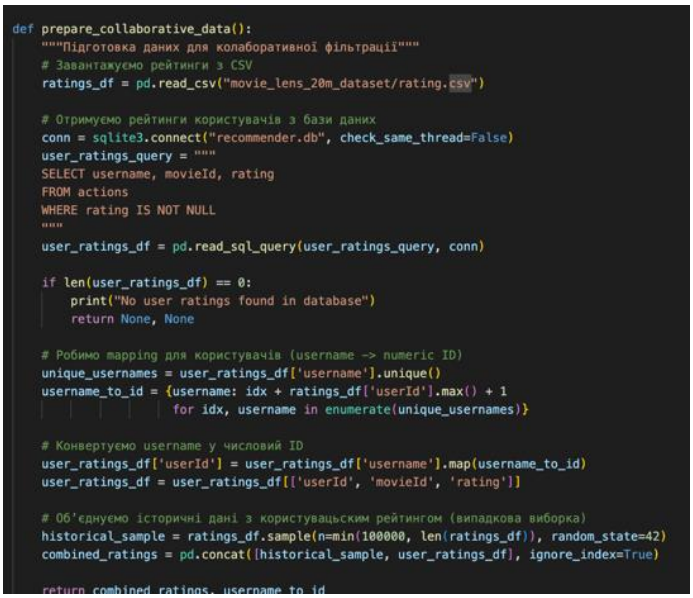
```python
try:
    response = requests.get(
        f"{TMDB_BASE_URL}{tmdb_id}?language=en-US",
        headers={
            "Authorization": TMDB_API_KEY,
            "accept": "application/json"
        }, timeout=4
    )
    if response.status_code == 200:
        data = response.json()
        poster_url = data.get("poster_path")
        if poster_path is None:
            belongs = data.get("belongs_to_collection")
            if belongs and isinstance(belongs, dict):
                poster_path = belongs.get("poster_path")
        print(poster_path)
except Exception as e:
    print(f"Error fetching TMDB for movieId {row['movieId']}:", e)
```

Рисунок 5.6 – Інтеграція з TMDB [31] (рисунок виконано самостійно)

5.2.1 Підготовка даних

Для тренування моделі колаборативної фільтрації була створена комбінована вибірка, яка включає до 100 000 випадкових оцінок з датасету MovieLens 20M, а також актуальні оцінки, що надходять від користувачів системи. Розмір вибірки у 100,000 записів був визначений емпірично після серії експериментів з різними

значеннями. Тестування показало, що вибірка у 10,000 записів недостатня для формування якісних рекомендацій, тоді як збільшення (наприклад до 500,000) записів призводить до повільного навчання системи. Для уникнення конфліктів ідентифікаторів було застосовано мапінг імен користувачів на нові числові ID, які починаються після максимального ID у базовому датасеті. Елемент коду представлений на рисунку 5.7.



```python
def prepare_collaborative_data():
    """Підготовка даних для колаборативної фільтрації"""
    # Завантажуємо рейтинги з CSV
    ratings_df = pd.read_csv("movie_lens_20m_dataset/rating.csv")

    # Отримуємо рейтинги користувачів з бази даних
    conn = sqlite3.connect("recommender.db", check_same_thread=False)
    user_ratings_query = """
    SELECT username, movieId, rating
    FROM actions
    WHERE rating IS NOT NULL
    """
    user_ratings_df = pd.read_sql_query(user_ratings_query, conn)

    if len(user_ratings_df) == 0:
        print("No user ratings found in database")
        return None, None

    # Робимо mapping для користувачів (username -> numeric ID)
    unique_usernames = user_ratings_df['username'].unique()
    username_to_id = {username: idx + ratings_df['userId'].max() + 1
                      for idx, username in enumerate(unique_usernames)}

    # Конвертуємо username у числовий ID
    user_ratings_df['userId'] = user_ratings_df['username'].map(username_to_id)
    user_ratings_df = user_ratings_df[['userId', 'movieId', 'rating']]

    # Об'єднуємо історичні дані з користувацьким рейтингом (випадкова виборка)
    historical_sample = ratings_df.sample(n=min(100000, len(ratings_df)), random_state=42)
    combined_ratings = pd.concat([historical_sample, user_ratings_df], ignore_index=True)

    return combined_ratings, username_to_id
```

Рисунок 5.7 – Підготовка даних для колаборативної фільтрації (рисунок виконано самостійно)

5.2.3 Реалізація колаборативної фільтрації з використанням SVD

Для написанні моделі було використано бібліотеку Surprise[4], яка дозволяє працювати з колаборативними моделями. Алгоритм SVD створює два латентні векторні представлення: для кожного користувача, як вектор вподобань та для кожного фільму, як вектор характеристик). Кожен рейтинг моделюється за формулою 5.1:

$$\hat{r}_{u,i} = \mu + b_u + b_i + q^T{}_i * p_u \quad (5.1)$$

де μ – глобальне середнє значення рейтингу,

[4] Бібліотека Surpriselib

b_u, b_i – зміщення для користувача та фільму відповідно,

q_i, p_u – латентні вектори ознак фільму та вподобань користувача.

Латентні фактори q_i та p_u представляють приховані характеристики (наприклад, "схильність до драм", "любов до спецефектів"), які SVD автоматично виявляє з патернів рейтингів. Конфігурація алгоритму була визначена на основі емпіричних досліджень та аналізу академічної літератури з рекомендаційних систем. Відповідний елемент коду представлений на рисунку 5.8.

```
def train_svd_model():
    """Навчання SVD моделі на об'єднаних даних"""
    global trained_svd_model, username_mapping_global

    print("Preparing data for collaborative filtering...")
    combined_ratings, username_mapping = prepare_collaborative_data()

    if combined_ratings is None:
        print("No data available for training")
        return None, None

    username_mapping_global = username_mapping

    # Створюємо Dataset для surprise
    reader = Reader(rating_scale=(0.5, 5.0))
    data = Dataset.load_from_df(combined_ratings[['userId', 'movieId', 'rating']], reader)

    # Розбиваємо на train/test
    trainset, testset = train_test_split(data, test_size=0.2, random_state=42)

    # Створюємо та навчаємо SVD модель
    print("Training SVD model...")
    svd = SVD(
        # 50-100 латентних факторів оптимально для фільмових датасетів. Менше факторів призводить до underfitting, більше - до overfitting при обмежених даних
        n_factors=50,
        # Емпірично встановлено, що 10 епох достатньо для конвергенції SGD на комбінованих даних при збереженні захисту від перенавчання
        n_epochs=10,
        # Learning rate. Консервативна швидкість навчання забезпечує стабільну конвергенцію. Більші значення можуть спричинити осциляції, менші - повільну конвергенцію.
        lr_all=0.005,
        reg_all=0.02,
        # Регуляризація
        random_state=42
    )

    svd.fit(trainset)
    trained_svd_model = svd
```

Рисунок 5.8 – Створення моделі колаборативної фільтрації (рисунок виконано самостійно)

Реалізація включає механізм lazy loading, коли модель навчається лише при першому запиті рекомендацій. Збережена модель зберігається статично та, як показано на відповідному елементі коду, представленому на рисунку 5.8, перенавчається, коли в моделі накопичується 50 нових рейтингів від користувача.

```
# Перенавчання моделі
def retrain_model_if_needed():
    """Для випадків, коли в нас з'явилось достатньо історичних даних"""
    global trained_svd_model

    conn = sqlite3.connect("recommender.db", check_same_thread=False)
    c = conn.cursor()

    # Перевіряємо кількість нових рейтингів
    c.execute("SELECT COUNT(*) FROM actions WHERE rating IS NOT NULL")
    new_ratings_count = c.fetchone()[0]

    # Перенавчаємо кожні 50 нових рейтингів
    if new_ratings_count > 0 and new_ratings_count % 50 == 0:
        print("Retraining SVD model with new data...")
        trained_svd_model = None  # Скинути модель
        train_svd_model()
```

Рисунок 5.8 – Перенавчання моделі (рисунок виконано самостійно)

5.2.4 Гібридизація

Модель працює таким чином, що коли користувач залишає хоча б 15 оцінок фільмам через інтерфейс, інформація зберігається в таблиці actions. Система автоматично перевіряє кількість оцінок користувача при кожному запиті рекомендацій. Якщо користувач має менше 10 оцінок, система використовує жанрові рекомендації. При накопиченні 10 або більше оцінок активується колаборативна фільтрація на основі SVD.

Процес генерації рекомендацій включає наступні етапи: спочатку система створює числовий маппінг для нового користувача та інтегрує його рейтинги з базовими даними MovieLens. Потім SVD модель обчислює передбачені рейтинги для всіх неоцінених фільмів, використовуючи латентні фактори користувача та фільмів.

З метою досягнення оптимального балансу між точністю та різноманітністю пропозицій, обидва підходи (колаборативний та жанровий), застосовуються одночасно, як показано на рисунку 5.9.

```python
app.py > hybrid_recommend
# Гібриждні рекомендаціх (комбинация колаборативної та за жанрами)
def hybrid_recommend(username, collaborative_weight=0.7):
    """Гібридні рекомендації: комбінація коллаборативної та контентної фільтрації"""
    try:
        from collaborative_module import collaborative_recommend

        # Отримуємо жанрові рекомендації
        print("Getting genre-based recommendations...")
        genre_resp = genre_based_recommend(username)
        if isinstance(genre_resp, Response):
            genre_recs = genre_resp.get_json()
        else:
            genre_recs = genre_resp

        # Отримуємо коллаборативні рекомендації
        print("Getting collaborative recommendations...")
        collab_resp = collaborative_recommend(username)
        if isinstance(collab_resp, Response):
            collab_recs = collab_resp.get_json()
        else:
            collab_recs = collab_resp

        print("log: we made it here")

        # Якщо не вдалося отримати коллаборативні - fallback на жанрові
        if not collab_recs or collab_recs is None:
            print("No collaborative recommendations available, falling back to genre-based")
            return genre_based_recommend(username)

        # Комбінуємо рекомендації
        print("Combining recommendations...")
        hybrid_scores = combine_recommendations(genre_recs, collab_recs, collaborative_weight)

        # Формуємо фінальний результат з постерами
        result = format_hybrid_results(hybrid_scores)

        return jsonify(result)

    except Exception as e:
        print(f"Error in hybrid recommendation: {e}")
        # Fallback до жанрових рекомендацій
        return genre_based_recommend(username)
```

Рисунок 5.9 – Гібридизація рекомендацій (рисунок виконано самостійно)

Спочатку генеруються жанрові рекомендації, потім колаборативні, після чого отримані рекомендації зваженого комбінуються, як показано на рисунку 5.10.

```python
def combine_recommendations(genre_recs, collab_recs, collaborative_weight=0.7):
    """Комбінує жанрові та коллаборативні рекомендації"""
    genre_weight = 1.0 - collaborative_weight
    combined_scores = {}

    # маплінг якщо потрібно — у dict по movieId
    if isinstance(genre_recs, list):
        genre_recs = {item['movieId']: item for item in genre_recs}
    if isinstance(collab_recs, list):
        collab_recs = {item['movieId']: item for item in collab_recs}

    # Отримуємо всі унікальні фільми
    all_movie_ids = set(genre_recs.keys()) | set(collab_recs.keys())

    for movie_id in all_movie_ids:
        genre_score = genre_recs.get(movie_id, {}).get('genre_score_norm', 0)
        collab_score = collab_recs.get(movie_id, {}).get('collab_score_norm', 0)

        # Гібридний скор
        hybrid_score = (genre_weight * genre_score) + (collaborative_weight * collab_score)

        # Додаємо бонус за популярність (середній рейтинг)
        mean_rating = genre_recs.get(movie_id, {}).get('mean', 0)
        if mean_rating > 0:
            rating_bonus = (mean_rating - 2.5) / 5.0  # Нормалізуємо до [-0.5, 0.5]
            hybrid_score += 0.1 * rating_bonus  # Невеликий бонус за якість

        combined_scores[movie_id] = {
            'hybrid_score': hybrid_score,
            'genre_score': genre_score,
            'collab_score': collab_score,
            'title': genre_recs.get(movie_id, {}).get('title') or collab_recs.get(movie_id, {}).get('title', 'Unknown'),
            'mean_rating': mean_rating,
            'count': genre_recs.get(movie_id, {}).get('count', 0)
        }

    return combined_scores
```

Рисунок 5.10 – Комбінування рекомендацій (рисунок виконано самостійно)

Коефіцієнти комбінування були встановлені на основі емпіричного тестування з групою користувачів. Для колаборативної фільтрації було обрано вагу 0.7 (70%), а для жанрової фільтрації – 0.3 (30%).

5.3 Довгострокова задоволеність

До інтерфейсу додамо кнопки дій, що будуть імітувати основну поведінку користувача: «Додати в переглянути пізніше», «Почати перегляд», «Закінчити перегляд» та оцінка рейтингу від одного до п’яти. (див. рис. 5.11).

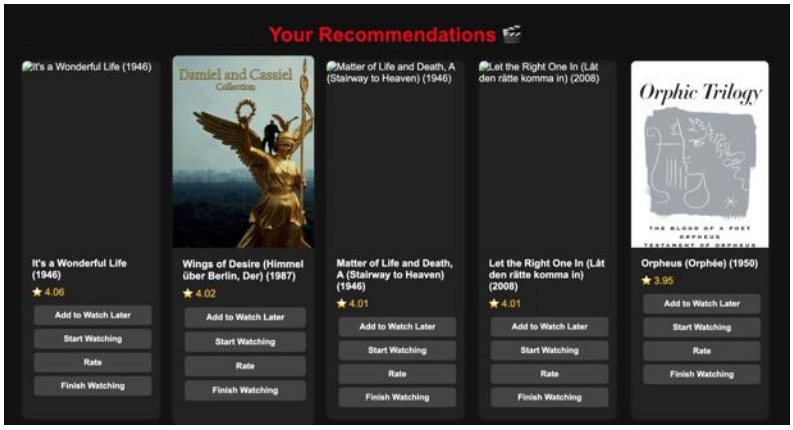


Рисунок 5.11 – Додавання взаємодії користувача з фільмами (рисунок виконано самостійно)

Кожна з дій була визначена значенням проксі-нагороди, як показано на рисунку 5.12.

```
# Прокси-нагороды за дії користувача
PROXY_REWARDS = {
    "like": 1.0,
    "watch_later": 0.3,
    "start": 0.2,
    "finish": 0.7,
    "dislike": -1.0,
    "rate_5": 1.0,
    "rate_4": 0.5,
    "rate_3": 0.0,
    "rate_2": -0.5,
    "rate_1": -1.0
}
```

Рисунок 5.12 – Визначення дій як проксі нагород (рисунок виконано самостійно)

Значення проксі винагороди були визначені на основі досліджень неявного зворотного зв'язку в рекомендованих системах. Так наприклад, завершення перегляду розглядається як сильний індикатор задоволення користувачів [38], що обґрунтовує більш високе значення для 'завершити перегляд' у порівнянні з іншими типами взаємодій. Для дій, що індукують, що користувачу не сподобалось (рейтинг «один» чи «два» тощо), поставимо негативні значення.

При створенні системи рекомендацій ініціалізуються дві ключові структури даних, як показано на рисунку 5.13.

```
class ContextualBandit:
    def __init__(self, d_context, alpha=2.0):
        self.d = d_context
        self.alpha = alpha
        # використання контексту
        self.A = np.identity(self.d)
        self.b = np.zeros((self.d, 1))
        self.model_path = "bandit_model.pkl"
        self.load_model()
```

Рисунок 5.13 – Ініціалізація моделі (рисунок виконано самостійно)

Матриця A розміром 20×20 представляє коваріаційну матрицю, яка відслідковує взаємозв'язки між характеристиками фільмів. Спочатку це одинична

матриця, що означає відсутність попередніх знань про зв'язки між жанрами, рейтингами та іншими ознаками.

Вектор b розміром 20×1 накопичує зважені нагороди для кожної з двадцяти характеристик контексту.

Початково всі значення нульові, оскільки система ще не отримала жодної користувацької реакції. Параметр alpha контролює баланс між дослідженням нових варіантів та використанням вже вивчених переваг. Більше значення призводить до більш агресивного дослідження.

Реалізація контекстуального бандита базується на алгоритмі LinUCB (Linear Upper Confidence Bound), як показано на рисунку 5.14. Згідно до нього значення фільму – це сума очікуваної нагороди та довірчого інтервалу.

```python
def get_recommendation_score(self, context_vector):
    """Отримуємо значення рекомендації"""
    try:
        A_inv = np.linalg.inv(self.A)
        theta = A_inv.dot(self.b)
        x = context_vector.reshape(-1, 1)

        # LinUCB формула: очікувана нагорода + доверительный интервал
        expected_reward = theta.T.dot(x)[0, 0]
        confidence = self.alpha * np.sqrt(x.T.dot(A_inv).dot(x))[0, 0]
        print(f"Expected: {expected_reward:.3f}, Confidence: {confidence:.3f}, Score: {score:.3f}")

        return expected_reward + confidence
    except np.linalg.LinAlgError:
        # Якщо матриця не обратима, додаємо регуляцію
        A_reg = self.A + 0.1 * np.identity(self.d)
        A_inv = np.linalg.inv(A_reg)
        theta = A_inv.dot(self.b)
        x = context_vector.reshape(-1, 1)

        expected_reward = theta.T.dot(x)[0, 0]
        confidence = self.alpha * np.sqrt(x.T.dot(A_inv).dot(x))[0, 0]

        return expected_reward + confidence
```

Рисунок 5.14 – Обчислення значення фільму (рисунок виконано самостійно)

Очікувана нагорода – це передбачення системи, наскільки користувачу сподобається конкретний фільм. Наприклад, якщо система вивчила, що користувач любить комедії та фільми з високим рейтингом, то для комедії з високим рейтингом очікувана нагорода буде високою вищою, ніж для фільму жахів з низьким рейтингом.

Довірчий інтервал – це міра невпевненості системи у своєму передбаченні. Якщо про фільм мало даних або він має рідкісну комбінацію жанрів, система менш упевнена у своєму передбаченні, тому довірчий інтервал буде великим. Це дає шанс невідомим фільмам потрапити у рекомендації.

Процес оцінки фільму починається з обчислення оберненої матриці A, яка необхідна для знаходження оптимальних параметрів моделі. Вектор theta представляє вивчені системою переваги користувача для кожної характеристики фільму. Очікувана нагорода обчислюється як скалярний добуток вектора параметрів на контекстний вектор фільму. Це дає лінійне передбачення того, наскільки фільм сподобається користувачу на основі його характеристик.

Кожна взаємодія користувача з фільмом призводить до оновлення моделі за логікою, показаною на рисунку 5.15.

```
def update(self, context_vector, reward):
    """Оновлюємо модель на основі зворотнього зв'язку"""
    x = context_vector.reshape(-1, 1)
    self.A += x.dot(x.T)
    self.b += reward * x
    self.save_model()
```

Рисунок 5.15 – Оновлення моделі (рисунок виконано самостійно)

5.4 Тестування та аналіз результатів

5.4.1 Тестування моделей жанрових та колаборативних рекомендацій

Валідація жанрового підходу відбувалася переглядом співпадінь улюблених жанрів користувача та рекомендацій. Результати показали очікуване 100 відсоткове співпадіння.

Валідація якості колаборативної моделі здійснювалася через розділення даних на навчальну та тестову вибірки у співвідношенні 80/20. Для валідації моделі була обрана метрика RMSE – середньоквадратична помилка. RMSE достатньо сильно штрафує більші помилки, що може мати сенс у контексті систем рекомендацій [12], бо чутливість до великих помилок передбачення, важлива для рекомендаційних систем, де значні відхилення можуть призвести до незадовільного користувацького досвіду.

Отримане значення RMSE: 0.9674, що свідчить про високу точність рекомендацій. Гарним значенням середньоквадратичного відхилення RMSE, що вказує на відносно точні прогнози, як правило, вважається значення від 0,2 до 1 [39].

5.4.2 Тестування контекстуального бандиту

Стосовно контекстуальних бандитів, було ідентифіковано проблему, з якою зіткнулась модель: рекомендації переважно старих фільмів, як показано на рисунку 5.16, не дивлячись на те, що було враховано рік випуску фільмів, що сподобались користувачу.

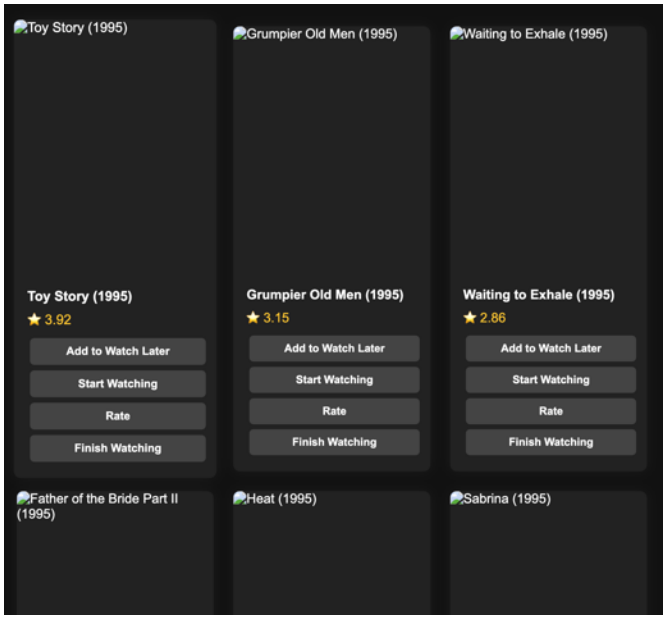


Рисунок 5.16 – Рекомендації контекстуального бандиту (рисунок виконано самостійно)

Причина цього була в тому, що старі фільми мають більш стабільні та передбачувані характеристики. Чим більше оцінок (історичних даних) накопичено, тим вище показник популярності, класичні жанри краще збігаються з базовими уподобаннями, а сталий рейтинг дає менше невизначеності. Тому бандит обирає «безпечні» варіанти, де фільми з найменшою невизначеністю.

Для подолання проблеми було прийнято рішення ввести агресивного бонусу за новизну фільму у контекстному векторі, як показано на рисунку 5.17.

```
# 3. Бонус за новітність
title = movie_row['title']
year = extract_year_from_title(title)
if year:
    normalized_year = max(0, min(1, (year - 1990) / 35))

    # Бонуси
    if year >= 2020:
        recency_bonus = 100.0   # Дуже дуже великий бонус для 2020+
    elif year >= 2015:
        recency_bonus = 50.0    # Дуже великий бонус для 2015+
    elif year >= 2010:
        recency_bonus = 25.0    # Великий бонус для 2010+
    elif year >= 2005:
        recency_bonus = 10.0    # Середній бонус для 2005+
    elif year >= 2000:
        recency_bonus = 5.0     # Маленький бонус для 2000+
    else:
        recency_bonus = -50.0   # Великий штраф для старих фільмів
else:
    normalized_year = 0
    recency_bonus = -50.0

context_features.extend([normalized_year, recency_bonus])

# 4-7. Інші характеристики
if user_genres and movie_genres:
    user_match_score = sum(1 for genre in user_genres
                           if genre in movie_genres) / len(user_genres)
else:
```

Рисунок 5.17 – Бонус за новітність у контексті фільу (рисунок виконано самостійно)

У функції формування контексту було додано спеціальний компонент recency_bonus, який надає значні переваги сучасним фільмам.

Також параметр α було збільшено з 0.1 до 20.0, що значно посилює схильність алгоритму до досліджування нових варіантів замість експлуатації вже відомих "безпечних" рішень. Це змушує бандит частіше обирати фільми з більшою невизначеністю, включаючи сучасні.

Надалі було проведено тестування порівняння довгострокового задоволення користувача контекстуальних бандитів та колаборативної фільтрації. Метою цього тесту було зрозуміти, яка система працює краще з точки зору довгострокового задоволення.

Для цього було вирішено зробити симуляцію користувача. Наприклад, гіпотетичний користувач на ім’я Ганна. Вона в січні любить дивитися романтичні комедії, у березні трилери, червні документальні фільми і так далі, а потім знову комедії.

Колаборативна фільтрація працює таким чином, що дивиться на всі оцінки Ганни та рекомендує їй те, що сподобалось користувачам, які таким же чином оцінили фільми, які оцінила вона. Проблема в тому, що може Ганна навіть не ставить оцінки тому, що їй сподобалось. Також в січні в неї почалася підготовка до

магістерської роботи, та вона лише інколи заходила на сервіс подивитися, що там є нового. Вона зазвичай або додавала то, що її цікавить в подивитись пізніше, або максимум починала дивитись та відкладала. Контекстуальний бандит має врахувати як раз ці важливі дії, які колаборативна чи жанрова фільтрація пропустять.

Для симуляції користувача початкові уподобання вкажемо як. Що сесії смаки будуть змінюватися з невеликим гауссівським шумом ($\sigma = 0.05$), щоб відтворити реалістичність (може фільм підходить Гані ідеально, але там грає актор, якого вона ненавидить, тому, фільм їй не сподобався).

Для кожного фільму також буде формуватися контекстний вектор. На основі скалярного добутку контексту фільму та уподобань користувача з додаванням шуму можемо оцінити, на скільки смаки користувача у конкретну сесію та рекомендований фільм співпадають.

Було проведено 100 експериментів. На рисунку 5.18 видно, що сині точки (контекстний бандит) переважно розташовані вище червоних точок (колаборативна система), що свідчить про систематично кращу продуктивність бандита. Особливо помітно, що у верхній частині графіка (кращі результати) домінують сині точки, тоді як червоні концентруються переважно у нижній частині. Це демонструє не випадкову перевагу, а стабільну тенденцію переваги контекстного бандита над традиційною колаборативною фільтрацією.

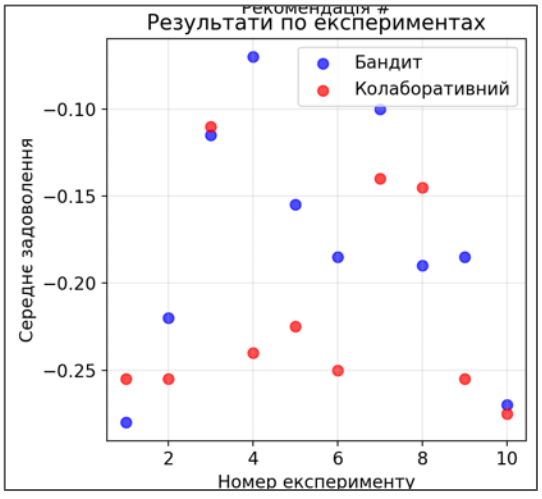


Рисунок 5.18 – Порівняння середнього задоволення користувачів (рисунок виконано самостійно)

На рисунку 5.19 відображений результат статистичного аналізу розподілів обох систем. Медіана бандита знаходиться помітно вище медіани колаборативної системи, що підтверджує кращу продуктивність у середньому. Крім того, інтерквартільний розмах (висота стовпчику) у бандита є меншим, що свідчить про вищу стабільність та передбачуваність результатів. Це означає, що бандит не тільки дає кращі результати в середньому, але й працює більш консистентно з меншою варіативністю між різними експериментами.

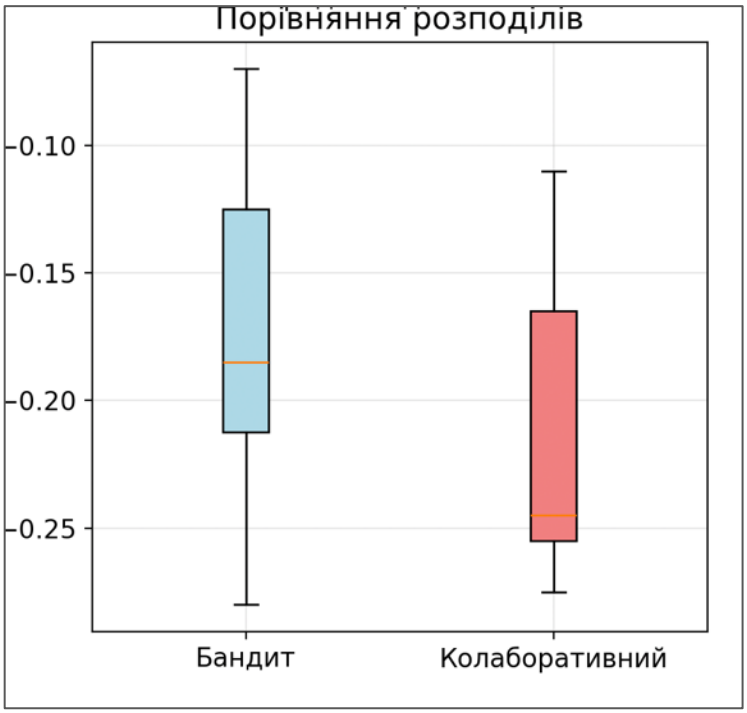


Рисунок 5.19 – Статистичний розподіл середнього задоволення користувачів (рисунок виконано самостійно)

Стовпчикова діаграма на рисунку 5.20 показує процентний розподіл перемог бандита по трьох ключових метриках. Найвища перевага спостерігається у "Середньому задоволенні" (70%), що вказує на загалом кращу якість рекомендацій. Метрики "Адаптація" та "Позитивний досвід" показують однакові 60% перемог, демонструючи здатність бандита краще пристосовуватися до змін користувацьких уподобань та частіше генерувати рекомендації, які подобаються користувачам. Загалом бандит виграє у 63.3% всіх метрик (190 з 300), що є статистично значущим результатом.

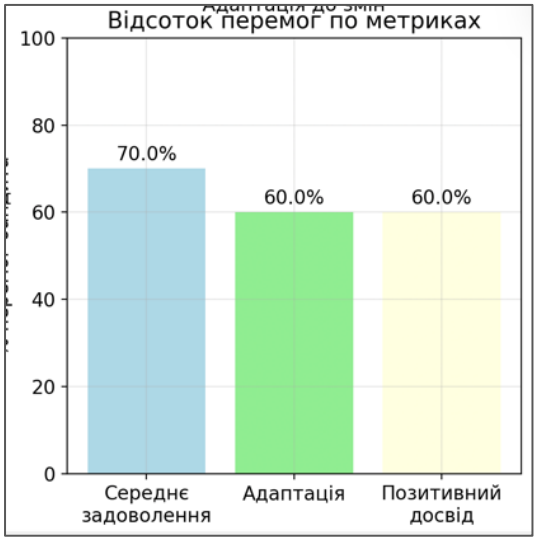


Рисунок 5.19 – Відсоток перемог контекстного бандита по ключових метриках ефективності (рисунок виконано самостійно)

5.5 Недоліки та обмеження системи

У даній роботі була проведена спроба вирішення проблеми довгострокового задоволення користувачів у рекомендаційних системах. Тим не менш, рішення не є ідеальним. По-перше, слід зазначити обмеження масштабованості через обмеженність ресурсів. Система використовує лише 100,000 записів з MovieLens 20M, що значно менше за реальні обсяги даних великих платформ. Netflix обробляє мільярди взаємодій, що може кардинально змінити поведінку алгоритмів. Крім того, система не тестувалася на одночасну роботу з великою кількістю користувачів.

По-друге, відсутність можливості тестування на справжніх користувачах. Тестування системи у реальних умовах само по собі являється окремим дослідженням, що включає взаємодію з користувачами. У роботі використовується імітація користувацького задоволення, але для того, щоб справді зрозуміти вподобання людини необхідні реальні користувачі конкретної системи. Так, наприклад, було виявлено, що алгоритм LinUCB показав схильність до консервативних рішень, надаючи перевагу старим фільмам з більшою кількістю даних. І хоч ця проблема була вирішена, тим не менш, складно сказати, чи є якісь приховані проблеми системи, які через обмеженість часу дослідження просто не

було виявлено. Також складно, чи було б це проблемою для конкретної системи, бо насправді головне – це влаштовувати задоволеність користувача. В контексті кожної системи користувачі будуть різні, тому будуть різні їх вподобання та патерни поведінки. Це каже про неможливість досягти універсальності та про комплексність подібних систем: у розробці систем для людей необхідно поєднання програмної інженерії з соціальними науками.

5.6 Подальші шляхи розвитку

Потенціал продовження роботи на основі використання отриманих результатів у науковій і практичній діяльності великий. Впродовж розробки системи були визначені такі ключові аспекти, що можна розвивати як окремі напрямки дослідження, як ефективна інтеграція різних моделей рекомендацій в одну систему, дослідження значення проксі-нагород на якість рекомендацій, тестування та розробка покращених SVD моделей для колаборативної рекомендації.

У майбутньому доцільно продовжити дослідження, працюючи у описаних напрямках розвитку, а саме оптимізації системи, дослідження впливу зміни значення проксі нагород для різних показників дій користувача, використання штучного інтелекту та машинного навчання для передбачення на основі проксі нагород, оптимізація та тестування різних видів моделей бандитів, тестування системи на одночасному використанні різної кількості користувачів. У роботі була реалізована мінімальна інтеграція з TMDB [31], але це також можна продовжити як пошук релевантних залежностей між фільмами на основі метрик з додавання збагачених даних шляхом інтеграції з TMDB та IMDB [32].

Майбутні дослідження також можуть побудовані врахуванні системою демографічних характеристик користувачів, що могло б покращити персоналізацію, або мати вектор досліджень, описаній у праці Spotify проблеми відкладеного зворотного зв’язку [5]. Глобально, робота може бути продовжена як поєднання вивчення користувачів з боку бізнесу чи соціальних наук та адаптацію здобутків у алгоритми покращення взаємодії з користувачем.

ВИСНОВКИ

В сьогоденному інтернеті, де вибір інформації величезний, існує потреба фільтрувати, пріоритезувати та ефективно пропонувати інформацію, щоб полегшити проблему інформаційного та когнітивного перевантаження користувача. У цьому відношенні було досягнуто багато успіхів. Системи рекомендацій вирішують цю проблему, аналізуючи великий обсяг динамічно створеної інформації та надаючи користувачам персоналізований вміст і послуги.

Тим не менш, з розвитком інформаційних систем, попит користувачів та вимоги індустрії ставлять перед рекомендаційними системами нові виклики. Так, коли багато популярних сервісів типу соціальних мереж працюють з короткостроковими взаємодіями користувача та ставлять пріоритетом короткострокове утримання, проблемою деяких великих платформ, як Netflix та Spotify є довгострокове утримання користувача, якого складніше досягнути.

У ході виконання роботи було досягнуто мети проектування рекомендаційної системи, здатної враховувати як короткострокове, так і довгострокове задоволення користувачів, оптимізуючи персоналізацію контенту. Для цього було вирішено задачі, зокрема проведено аналіз існуючих підходів до створення рекомендаційних систем, розроблено архітектуру системи, реалізовано та протестовано систему.

Результатом роботи стала побудова дизайну рекомендаційної системи на основі набору даних MovieLens. На першому етапі роботи було створено систему рекомендації фільмів за жанрами, що дозволило сформувати первинні рекомендації. На наступному етапі було реалізовано колаборативну фільтрацію з використанням методів матричної факторизації (SVD), що дозволило проаналізувати вподобання користувачів і забезпечити персоналізацію рекомендацій на основі схожості. Особливістю розробленої системи стало впровадження модуля обчислення проксі-нагород, який базується на логуванні дій користувачів. Завдяки цьому система враховує як короткострокові, так і довгострокові метрики, що оцінюються за допомогою контекстуального бандита. Такий підхід дозволяє динамічно адаптувати рекомендації, орієнтуючись на актуальні потреби користувача та довгострокову взаємодію з платформою.

Практичне застосування майбутній результатів дослідження буде актуальним для різних галузей, включаючи потокові сервіси (відео, музика), електронну комерцію, освітні платформи. Система може використовуватися для автоматизації вибору контенту та підвищення лояльності аудиторії, забезпечуючи конкурентні переваги для компаній, що використовують рекомендаційні системи.

Водночас робота має певні обмеження як масштабованість, обмеженість ресурсів. Відсутність можливості та розробки стратегії реального тестування обмежує можливості повної валідації ефективності системи в реальних умовах використання. Крім того, значення проксі-нагород та коефіцієнти гібридизації моделей були встановлені емпірично без проведення систематичного A/B тестування, що може впливати на оптимальність рішень.

Незважаючи на ці обмеження, розроблена система демонструє перспективний підхід до вирішення проблеми довгострокового задоволення користувачів та може слугувати основою для подальших досліджень у цій галузі. У майбутньому доцільно продовжити дослідження, працюючи у описаних напрямках розвитку, а саме оптимізації системи, впровадження фільтрації на основі вмісту, дослідження впливу зміни значення проксі нагород для різних показників дій користувача, використання штучного інтелекту та машинного навчання для передбачення на основі проксі нагород, оптимізація та тестування різних видів моделей бандитів.

Виконана робота підтвердила важливість урахування довгострокового задоволення користувачів у рекомендаційних системах, практично застосувала отримані під час дослідження здобутки та відкрила нові можливості для їхнього вдосконалення. Результати дослідження можуть бути використані як основа для подальших розробок у цій сфері.

Під час виконання даної роботи було підготовлено та опубліковано тези на XI міжнародній науково-практичній конференція «Science And Technology: Challenges, Prospects And Innovations», Осака, Японія (див. додаток Б).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Chaubey J. How Machine Learning Plays A Role In Personalized Recommendations. Medium. URL: https://medium.com/infiniticube/how-machine-learning-plays-a-role-in-personalized-recommendations-5b7f63fb4be2 (дата звернення: 18.12.2024).

2. Cornellius Yudha Wijaya. Understanding Netflix Long-Term Satisfaction Recommendation System. Non-Brand Data. URL: https://www.nb-data.com/p/understanding-netflix-long-term-satisfication (дата звернення: 24.12.2024).

3. David Chong. Deep Dive into Netflix's Recommender System. Medium. URL: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48.

4. Deep learning for recommender systems: A Netflix case study / Harald Steck та ін. {AI} magazine. 2021. Т. 42, № 3. С. 7–18. URL: https://onlinelibrary.wiley.com/doi/10.1609/aimag.v42i3.18140 (дата звернення: 24.12.2024).

5. Optimizing for the Long-Term Without Delay / Thomas M. McDonald та ін. Spotify Research. URL: https://research.atspotify.com/2023/07/optimizing-for-the-long-term-without-delay/ (дата звернення: 27.12.2024).

6. Reward innovation for long-term member satisfaction / Gary Tang та ін. 17th {ACM} Conference on Recommender Systems. 2023. С. 396 – 399. URL: https://doi.org/10.1145/3604915.3608873 (дата звернення: 25.12.2024).

7. Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication / X. {HE} та ін. {RecSys} '20: Proceedings of the 14th {ACM} Conference on Recommender Systems. Virtual Event, Brazil, 2020. С. 210–219. URL: https://doi.org/10.1145/3383313.3412233 (дата звернення: 27.12.2024).

8. Vlassis N., Gil F., Chandrashekar A. Off-Policy Evaluation of Slate Policies under Bayes Risk. 2021. URL: https://www.researchgate.net/publication/348324749_Off-

Policy_Evaluation_of_Slate_Policies_under_Bayes_Risk (дата звернення: 27.12.2024).

9. A contextual-bandit approach to personalized news article recommendation / Lihong Li та ін. {WWW} '10: Proceedings of the 19th international conference on World wide web. 2010. С. 661 – 670. URL: https://dl.acm.org/doi/abs/10.1145/1772690.1772758 (дата звернення: 27.12.2024).

10. Lucas Maystre, Russo D., Zhao Y. Optimizing Audio Recommendations for the Long-Term: A Reinforcement Learning Perspective. 2023. URL: https://www.researchgate.net/publication/368332881_Optimizing_Audio_Recommendations_for_the_Long-Term_A_Reinforcement_Learning_Perspective (дата звернення: 27.12.2024).

11. Temporal modeling of user preferences in recommender system Chalyi, S., Leshchynskyi, V. CEUR Workshop Proceedings, 2020, 2711, с. 518-528.

12. Method of forming recommendations using temp oral constraints in a situation of cyclic cold start of the recommender system Chalyi, S., Leshchynskyi, V., Leshchynska, I. EUREKA, Physics and Engineering, 2019, 2019(4), с. 34-40.

13. Method of constructing explanations for recommender systems based on the temporal dynamics of user preferences Chaly i, S., Leshchynskyi, V. EUREKA, Physics and Engineering, 2020, 2020(3), стр. 43-50 DOI 10.21303/2461-4262.2020.001228.

14. Getting to know you: learning new user preferences in recommender systems / A. M. Rashid та ін. {IUI} '02: Proceedings of the 7th international conference on Intelligent user interfaces. 2002. С. 127 – 134. URL: https://doi.org/10.1145/502716.502737 (дата звернення: 22.12.2024).

15. Schafer B., Konstan J., Riedl J. Recommender Systems in E-Commerce. 1st {ACM} Conference on Electronic Commerce, Denver, Colorado, United States. 1999. URL: https://doi.org/10.1145/336992.337035 (дата звернення: 22.12.2024).

16. Meteren V. Using content-based filtering for recommendation. In Proceedings of the Machine Learning in the New Information Age {MLnet}/{ECML}2000 Workshop : phdthesis. Barcelona, Spai, 2000. URL: https://users.ics.forth.gr/~potamias/mlnia/paper_6.pdf (дата звернення: 18.12.2024).

17. Pearl Pu, Li Chen, Rong Hu. A user-centric evaluation framework for recommender systems. {RecSys} '11: Proceedings of the fifth {ACM} conference on Recommender systems. 2011. С. 157 – 164. URL: https://doi.org/10.1145/2043932.2043962 (дата звернення: 22.12.2024).

18. Rong Hu, Pearl Pu. Acceptance issues of personality-based recommender systems. {RecSys} '09: Proceedings of the third {ACM} conference on Recommender systems. 2009. С. 221 – 224. URL: https://doi.org/10.1145/1639714.1639753 (дата звернення: 22.12.2024).

19. Empirical analysis of the impact of recommender systems on sales / B. Pathak та ін. Journal of Management Information Systems. 2010. Т. 27, № 2. С. 159 – 188. URL: https://www.scopus.com/record/display.uri?eid=2-s2.0-78649357436 (дата звернення: 22.12.2024).

20. F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh. Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal. 2015. Т. 16, № 3. С. 261–273. URL: https://doi.org/10.1016/j.eij.2015.06.005 (дата звернення: 22.12.2024).

21. Roy D., Dutta M. A systematic review and research perspective on recommender systems. Journal of Big Data. 2022. Т. 9, № 59. URL: https://doi.org/10.1186/s40537-022-00592-5 (дата звернення: 18.12.2024).

22. Burke R. Hybrid Recommender Systems: Survey and Experiments. Modeling and User-Adapted Interaction. 2002. Т. 12. С. 331–370. URL: https://doi.org/10.1023/A:1021240730564 (дата звернення: 22.12.2024).

23. Kumar P., Ramjeevan T. Recommendation system techniques and related issues: a survey. International Journal of Information Technology. 2018. Т. 10. С. 495–501. URL: https://rdcu.be/d4vil (дата звернення: 22.12.2024).

24. Chen Z., Wang S. A review on matrix completion for recommender systems. Knowledge and Information Systems. 2022. Т. 64. С. 1–34. URL: https://doi.org/10.1007/s10115-021-01629-6 (дата звернення: 23.12.2024).

25. Heterogeneous graph inference with range constrainted -collaborative matrix factorization for small molecule-{miRNA} association prediction / Shudong Wang та

ін. Computational Biology and Chemistry. 2024. Т. 110. URL: https://doi.org/10.1016/j.compbiolchem.2024.108078 (дата звернення: 23.12.2024).

26. A collaborative filtering recommendation framework utilizing social networks / Aamir Fareed та ін. Machine Learning with Applications. 2023. Т. 14. URL: https://doi.org/10.1016/j.mlwa.2023.100495 (дата звернення: 23.12.2024).

27. Chris Alvino, Justin Basilico. Learning a Personalized Homepage. Netflix Technology Blog. URL: https://netflixtechblog.com/learning-a-personalized-homepage-aa8ec670359a (дата звернення: 25.12.2024).

28. Pastukhov D. Inside Spotify's Recommender System: A Complete Guide to Spotify Recommendation Algorithms. Music tomorrow. URL: https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022 (дата звернення: 28.12.2024).

29. Multi-Armed Bandits in Recommendation Systems: A survey of the state-of-the-art and future directions / Nícollas Silva та ін. Expert Systems with Applications. 2022. Т. 197. URL: https://doi.org/10.1016/j.eswa.2022.116669 (дата звернення: 26.12.2024).

30. Recommending for Long-Term Member Satisfaction at Netflix. Netflix Technology Blog. URL: https://netflixtechblog.com/recommending-for-long-term-member-satisfaction-at-netflix-ac15cada49ef (дата звернення: 26.12.2024).

31. Find By ID. The Movie Database (TMDB). URL: https://developer.themoviedb.org/reference/find-by-id (дата звернення: 13.01.2025).

32. IMDb: Ratings, Reviews, and Where to Watch the Best Movies & TV Shows. IMDb. URL: https://www.imdb.com/ (дата звернення: 03.06.2025).

33. D J. M. P., Kavlakoglu E. What is content-based filtering? | IBM. IBM - United States. URL: https://www.ibm.com/think/topics/content-based-filtering (дата звернення: 13.01.2025).

34. Bhat T. Recommendation system−Basics and Use cases. Medium. URL: https://medium.com/@tabindabhat/recommendation-system-basics-and-use-cases-6c5a8b26b98 (дата звернення: 13.01.2025).

35. Maklin C. Model Based Collaborative Filtering−SVD. Medium. URL:

https://medium.com/@corymaklin/model-based-collaborative-filtering-svd-19859c764cee (date of access: 15.01.2025).

36. Vincent R. J. Singular Value Decomposition (SVD)–Working Example. Medium. URL: https://medium.com/intuition/singular-value-decomposition-svd-working-example-c2b6135673b5 (дата звернення: 17.01.2025).

37. Vinija's Notes • Recommendation Systems • Multi-Armed Bandits. Vinija Jain. URL: https://vinija.ai/recsys/multi-armed-bandit/ (дата звернення: 12.06.2025).

38. Gomez-Uribe C. A., Hunt N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM Transactions on Management Information Systems (TMIS). 2015. Т. 6, № 4. С. 1–19. URL: https://dl.acm.org/doi/10.1145/2843948 (дата звернення: 01.06.2025).

39. Chowdhury M. E. What's the acceptable value of Root Mean Square Error (RMSE), Sum of Squares due to error (SSE) and Adjusted R-square?. ResearchGate. URL: https://www.researchgate.net/post/Whats-the-acceptable-value-of-Root-Mean-Square-Error-RMSE-Sum-of-Squares-due-to-error-SSE-and-Adjusted-R-square (дата звернення: 14.06.2025).

40. StephanieOgu/2025_M_PI_IPZ-23-1_Ogu_S_I. GitHub. URL: https://github.com/StephanieOgu/2025_M_PI_IPZ-23-1_Ogu_S_I (дата звернення: 15.06.2025).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

11. Temporal modeling of user preferences in recommender system Chalyi, S., Leshchynskyi, V. CEUR Workshop Proceedings, 2020, 2711, с. 518-528.

12. Method of forming recommendations using temp oral constraints in a situation of cyclic cold start of the recommender system Chalyi, S., Leshchynskyi, V., Leshchynska, I. EUREKA, Physics and Engineering, 2019, 2019(4), с. 34-40.

13. Method of constructing explanations for recommender systems based on the temporal dynamics of user preferences Chaly i, S., Leshchynskyi, V. EUREKA, Physics and Engineering, 2020, 2020(3), стр. 43-50 DOI 10.21303/2461-4262.2020.001228.