МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

“Рекомендаційна система музичних творів на основі колаборативної фільтрації”

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп’ютерні науки»**

Освітня програма **«Комп’ютерні науки»**

Освітній рівень: **бакалавр**

Виконав: студент 4 курсу, групи КН-42
спеціальності – 122 «Комп’ютерні науки»

Кавун Анна Вікторівна
(прізвище та ініціали)

Керівник к.т.н., доц. Іларіонов О.Є.
(наук. ступінь, звання, прізвище та ініціали)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 05.06.2023 р.
зав. кафедри _____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. кафедри інтелектуальних технологій

к.т.н., доц. Іларіонов О.Є.
(звання, прізвище та ініціали)

(підпис)

« *15* » *лютого* 2023 р.

З А В Д А Н Н Я
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кавун Анні Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи: "Рекомендаційна система музичних творів на основі колаборативної фільтрації"
затверджена наказом ректора від "*11*" *листопада* 2022 року №4
2. Термін виконання проекту (роботи):): з 13.02.2023 до 28.05.2023
3. Вихідні дані до роботи: розробити модуль рекомендаційної музичної системи для забезпечення роботи колаборативної фільтрації
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) аналіз принципів побудови систем рекомендацій;
2) моделювання роботи модуля колаборативної фільтрації рекомендаційної музичної системи;
3) описання розробленого модуля фільтрації рекомендаційної музичної системи.
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):
мета, об'єкт та предмет (1 слайд), методи рекомендацій (1 слайд), (1 слайд), завдання на кваліфікаційну роботу (1 слайд), Основні бізнес-процеси в модулі колаборативної фільтрації (1 слайд), дерево функцій (1 слайд), діаграма роботи у форматі IDEF0 (1 слайд), діаграми рівня A1 (2 слайди), діаграма використання (1 слайд), архітектура системи рекомендацій (1 слайд), Таблиці БД (2 слайди) обробка даних лайків і переглядів (2 слайди), робочі вікна системи (2 слайди), висновки (1 слайди).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Іларіонов О.Є.	15.02.2022	15.02.2022
2	Іларіонов О.Є.	15.03.2022	15.03.2022
3	Іларіонов О.Є.	15.04.2022	15.04.2022

7. Дата видачі завдання 15 лютого 2023 року

Керівник дипломної роботи _____ / О.Є. Іларіонов /
(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ / А.В. Кавун /
(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Обговорення постановки завдання та змісту пояснювальної записки	15.02.2023 – 22.02.2023	
2	Вибір та формування теми	23.02.2023 – 25.02.2023	
3	Аналіз предметної області	26.02.2023 – 14.03.2023	
4	Вибір методів рішення задачі	15.03.2023 – 14.04.2023	
5	Створення програмного модулю та його описання	15.04.2023 – 15.05.2023	
6	Оформлення пояснювальної записки	16.05.2023 – 28.05.2023	

Керівник випускної кваліфікаційної роботи _____ / О.Є. Іларіонов /
(підпис) (ініціали та прізвище)

Студент _____ / А.В. Кавун /
(підпис) (ініціали та прізвище)

Анотація

Кавун Анна Вікторівна виконала випускну кваліфікаційну роботу на тему “Рекомендаційна система музичних творів на основі колаборативної фільтрації” за спеціальністю 122 – «Комп’ютерні науки».

Випускна кваліфікаційна робота присвячена розробці та реалізації рекомендаційної системи для музичного сервісу. Метою роботи є покращення користувацького досвіду та забезпечення персоналізованих рекомендацій, що відповідають музичним вподобанням кожного користувача.

У розділі 1 "Аналіз принципів побудови рекомендаційних систем" проведено огляд основних принципів та методів побудови рекомендаційних систем. Розглянуто різні підходи, такі як колаборативна фільтрація, заснована на вмісті фільтрація та гібридні підходи. Визначено ключові проблеми, які виникають при розробці таких систем, а також представлено різноманітні метрики якості та методи їх оцінки.

У розділі 2 "Моделювання роботи модуля колаборативної фільтрації рекомендаційної музичної системи" детально розглянуті основні алгоритми для побудови комп'ютерних систем формування рекомендацій, такі як алгоритми Learning to Rank, моделі оцінки вхідних змінних та приклади реалізації рекомендаційних алгоритмів. Також описана архітектура модуля колаборативної фільтрації, яка включає компоненти комплексного підходу до формування рекомендацій.

Розділ 3 "Опис розробленого модуля фільтрації рекомендаційної музичної системи" присвячений розробці самої рекомендаційної системи. Визначено програмні та технічні компоненти, які використовуються для реалізації модуля. Також надано детальний опис основних режимів роботи розробленої системи рекомендацій та проведено тестування її роботи.

Ключові слова: система рекомендацій, методи фільтрації, нейронна мережа, машинне навчання.

Summary

The degree project: «*Recommender system of musical works based on collaborative filtering*» *has completed by Anna Kavun specialty* 122 – «*Computer Scienses*».

The graduation qualification work is dedicated to the development and implementation of a recommendation system for a music service. The aim of the work is to improve the user experience and provide personalized recommendations that align with the music preferences of each user.

In Chapter 1 "Analysis of Principles for Building Recommendation Systems" an overview of the main principles and methods for building recommendation systems is provided. Various approaches are discussed, such as collaborative filtering, content-based filtering, and hybrid approaches. Key challenges in developing such systems are identified, and a variety of quality metrics and evaluation methods are presented.

Chapter 2 "Modeling the Operation of the Collaborative Filtering Module of the Music Recommendation System" provides a detailed analysis of the main algorithms for building computer-based recommendation systems, such as Learning to Rank algorithms, models for evaluating input variables, and examples of recommendation algorithm implementations. The architecture of the collaborative filtering module is described, which includes components that support a comprehensive approach to recommendation generation.

Chapter 3, "Description of the Developed Recommendation Filtering Module of the Music System" focuses on the development of the recommendation system itself. The software and technical components used for the implementation of the module are defined. A detailed description of the main operating modes of the developed recommendation system is provided, and testing of its functionality is conducted.

Keywords: *recommendation system, filtering methods, neural network, machine learning.*

ЗМІСТ

Перелік умовних позначень і скорочень

ANN	–	*artificial neural networks*
ARM	–	*advanced risc machine*
CMN	–	*Collaborative Memory Networks*
FISSA	–	*Fusing Item Similarity Models with Self-Attention*
ISO	–	*international organization for standardization*
NCF	–	*Neural collaborative filtering*
SASRec	–	*Self-Attentive Sequential Recommendation*

Вступ

Рекомендаційні системи є дуже актуальними в сучасному світі, оскільки дозволяють компаніям забезпечувати персоналізовані рекомендації своїм клієнтам. Це дозволяє збільшити продажі та залучення нових клієнтів, збільшити лояльність та задоволеність клієнтів, а також забезпечити більш ефективну рекламу продуктів та послуг.

Основна ідея рекомендаційних систем полягає в тому, щоб враховувати взаємодію користувачів з продуктами та послугами, щоб надати їм рекомендації, які найбільше підходять саме їм. Це дозволяє забезпечити персоналізацію рекомендацій та підвищити ефективність маркетингових кампаній.

Рекомендаційні системи дозволяють забезпечити кращу обробку великої кількості даних та підвищити точність рекомендацій. Наприклад, застосування методів машинного навчання дозволяє автоматично виявляти зв'язки між користувачами та продуктами, та надавати більш точні рекомендації.

Аналіз принципів побудови рекомендаційних систем починається зі збору даних, які необхідні для їх роботи. Це можуть бути різноманітні дані: інформація про користувачів, історії їхніх дій, а також характеристики об'єктів, що рекомендуються. Далі проводиться попередній аналіз цих даних, зокрема, відбираються параметри, що найбільше впливають на рекомендації.

Одним з ключових принципів роботи рекомендаційної системи є збір інформації про користувачів і об'єкти. Це може бути здійснене за допомогою різних методів, включаючи опитування, аналіз дій користувачів і збір даних з соціальних мереж. Важливим етапом є обробка отриманих даних, що дозволяє відібрати ключові параметри для рекомендацій.

Одним із ключових елементів рекомендаційної системи є алгоритми рекомендацій. Для їх вибору можна використовувати різні методи, включаючи колаборативну фільтрацію, контент-базовані підходи, гібридні моделі і т.д. Важливо вибрати той метод, який найкраще підходить до потреб користувачів і характеристик об'єктів.

Іншим важливим принципом роботи рекомендаційної системи є визначення метрик якості, які дозволяють оцінити її ефективність. Для цього можна використовувати різні метрики, наприклад, точність, покриття, рангова кореляція тощо.

Також важливим принципом є підтримка рекомендаційної системи, включаючи збір інформації про користувачів, оновлення бази даних, а також моніторинг та аналіз результатів. Це дозволяє підтримувати високу якість рекомендацій і забезпечувати їх відповідність змінним потребам користувачів.

Іншим важливим принципом є відкритість системи, що дозволяє використовувати її для інших цілей та інтегрувати з іншими системами. Наприклад, рекомендаційна система, розроблена для кінотеатру, може бути використана і для інших типів мультимедіа, таких як відеоігри або музика.

Останнім принципом є постійне вдосконалення системи. Так як потреби та інтереси користувачів можуть змінюватися з часом, то рекомендаційна система повинна постійно оновлюватися та вдосконалюватися. Нові методи та алгоритми можуть допомогти покращити якість рекомендацій та забезпечити більш точну передбачуваність потреб користувачів. Також важливо не забувати про зворотний зв'язок від користувачів, що може бути використано для вдосконалення системи та забезпечення більш точних та персоналізованих рекомендацій.

Усі ці принципи дозволяють забезпечити ефективність та точність рекомендаційної системи, що є ключовим для забезпечення задоволення користувачів та підвищення їх лояльності до сервісу.

Однак, рекомендаційні системи також стикаються з викликами та проблемами, пов'язаними зі збором та обробкою даних, забезпеченням конфіденційності даних, та питаннями етики. Тому, дослідження та розробка нових методів та алгоритмів рекомендаційних систем дозволить вирішити ці проблеми та забезпечити їх більш ефективне використання.

В межах досліджень рекомендаційних систем слід виділити дослідження колаборативної фільтрації, які є дуже актуальним, оскільки вони допомагають

розв'язати важливу задачу – персоналізацію рекомендацій. Основна ідея колаборативної фільтрації полягає в тому, що алгоритм рекомендацій будується на основі взаємодії користувачів з певними товарами або послугами. Таким чином, алгоритм може враховувати особисті інтереси кожного користувача та давати рекомендації, які найбільше підходять саме йому. Це дозволяє забезпечити персоналізацію рекомендацій та підвищити ефективність маркетингових кампаній.

Дослідження колаборативної фільтрації вимагає окремого розгляду викликів та проблем, пов'язаних з їх застосуванням. Зокрема, існують проблеми зі збором інформації про користувачів та їх взаємодію з продуктами, проблеми зі зберіганням та обробкою великої кількості даних, а також проблеми з недостатньою точністю рекомендацій.

Загалом, розробка та покращення колаборативної фільтрації та рекомендаційних систем вимагає поєднання зусиль з різних галузей, таких як математика, статистика, комп'ютерна наука та соціальні науки. Однак, якщо вдасться подолати виклики, пов'язані з розробкою цих систем, вони можуть значно полегшити життя користувачів та допомогти покращити ефективність бізнесу.

З огляду на вищесказане визначено об’єкт, предмет та мету дослідження:

– об’єкт дослідження – процес обробки даних для підтримки роботи системи рекомендацій;

– предмет дослідження – модуль збору і обробки даних для систем рекомендацій;

– мета дослідження – розробка модулю рекомендацій музичної системи.

Розділ 1
Аналіз принципів побудови рекомендаційних систем

1.1. Особливості використання рекомендаційних систем

Аналіз основних завдань при розробці рекомендаційних систем допомагає розуміти, які цілі ставляться перед такою системою і які проблеми необхідно вирішувати.

Одним з основних завдань є забезпечення якості рекомендацій. Рекомендації повинні бути релевантні та корисні для користувачів, що дозволить збільшити їх задоволеність від використання сервісу та покращити відносини з ним. Для досягнення цього завдання необхідно використовувати ефективні алгоритми рекомендацій, які враховують інтереси користувачів та мають високу точність.

Іншим важливим завданням є забезпечення персоналізації рекомендацій. Кожен користувач має свої унікальні інтереси та вподобання, тому рекомендації повинні бути налаштовані під кожного конкретного користувача. Для цього необхідно збирати та аналізувати інформацію про дії користувачів, що дозволить побудувати модель його інтересів та вподобань.

Також важливим завданням є підтримка рекомендаційної системи, включаючи збір інформації, зберігання даних та оновлення алгоритмів рекомендацій. Для забезпечення ефективності системи необхідно використовувати сучасні технології та інструменти, такі як машинне навчання, аналіз даних та обробка великих обсягів інформації. [3, 4].

Крім того, важливим завданням є забезпечення конфіденційності даних користувачів та захист від зловмисних атак, таких як впровадження некоректних даних або перехоплення комунікацій. Для цього можна використовувати різноманітні методи шифрування та автентифікації даних, а також регулярно оновлювати програмне забезпечення системи для усунення виявлених вразливостей.

Одним з основних завдань при розробці рекомендаційної системи є визначення метрик якості, які дозволяють оцінити ефективність системи. Для цього можна використовувати різноманітні показники, такі як точність рекомендацій, покриття каталогу, час перегляду/прослуховування кліпу та інші. При виборі метрик необхідно враховувати конкретні потреби та цілі рекомендаційної системи.

Також важливим завданням є забезпечення якісної обробки даних, яка передує рекомендаційному алгоритму. Необхідно використовувати методи обробки тексту та аналізу зображень, а також створювати системи збору даних та їх попередньої обробки. Крім того, важливим завданням є забезпечення підтримки різноманітних джерел даних, таких як соціальні мережі, веб-сайти та інші.

Ще одним важливим завданням є забезпечення персоналізації рекомендацій для кожного користувача. Для цього можна використовувати різноманітні підходи, такі як колаборативна та контент-базована фільтрація, гібридні методи та інші.

1.1.1. Аналіз методів вирішення проблем при розробці систем рекомендацій

Розвиток систем рекомендацій стикається з численними викликами, зокрема:

1. Брак якісних та кількісних даних: Рекомендаційні системи потребують великого обсягу даних для точної роботи. Проте, збір та аналіз даних може бути складним та дорогим, і відбуватися з різною якістю.

2. Взаємодія між користувачами та системою: Користувачі можуть не завжди довіряти рекомендаціям системи та не бути готовими сприймати їх. Для підвищення довіри користувачів можуть бути використані різноманітні техніки, такі як пояснення, фільтри етики та інші.

3. Особиста конфіденційність та безпека: Збір та аналіз даних може стати загрозою для приватності та безпеки користувачів, тому важливо забезпечити надійний захист даних та враховувати відповідні норми та закони.

4. Стабільність та надійність системи: Рекомендаційні системи повинні бути стійкими та надійними, забезпечуючи швидку та ефективну роботу без збоїв та перебоїв.

5. Ефективність та точність: Рекомендаційні системи повинні надавати точні та релевантні рекомендації, які відповідають потребам та інтересам користувачів. Для цього можуть бути використані різні методи та алгоритми, які мають бути оптимізовані для досягнення максимальної точності та ефективності.

6. Співпраця між виробниками та користувачами: Виробники повинні встановлювати довгострокові стосунки з користувачами та забезпечувати їх взаємодію з ними для збирання фідбеку та вдосконалення рекомендаційної системи на основі цих даних.

7. Підтримка нових форматів даних: Рекомендаційні системи повинні підтримувати різноманітні формати даних, такі як текстові дані, відео, зображення тощо, щоб забезпечити точні та релевантні рекомендації для користувачів.

8. Аналіз контенту та семантики: Рекомендаційні системи повинні вміти аналізувати контент та його семантику, щоб забезпечити користувачам релевантні рекомендації. Це вимагає використання складних алгоритмів та методів обробки природньої мови та комп'ютерного зору.

9. Прихований контент та диверсифікація: Рекомендаційні системи повинні враховувати наявність прихованого контенту та забезпечувати диверсифікацію рекомендацій для забезпечення більш широкого спектру варіантів для користувачів.

10. Персоналізація та контроль за вмістом: Рекомендаційні системи повинні вміти персоналізувати рекомендації для кожного користувача та забезпечувати контроль за вмістом, що рекомендується. Користувачі повинні мати можливість налаштовувати та контролювати рекомендації, які вони отримують, та визначати, які дані є конфіденційними.

Серед викликів до методів колаборативної фільтрації можна виділити:

1. Проблема холодного старту: колаборативна фільтрація базується на історичних даних про взаємодію користувачів з продуктами, тому при старті нової рекомендаційної системи виникає проблема недостатньої кількості даних для створення точних рекомендацій.

2. Проблема розмірності даних: у колаборативній фільтрації матриця рейтингів може бути дуже великою і розрідженою, що може збільшувати час обчислення та робити систему менш ефективною.

3. Проблема "кругової рекомендації": колаборативна фільтрація може викликати проблему, коли користувачі отримують тільки ті рекомендації, які вже вони раніше споживали, що обмежує можливість відкриття нових пропозицій.

4. Проблема групової динаміки: колаборативна фільтрація може не враховувати вплив групової динаміки на рекомендації. Наприклад, якщо всі члени групи поділяють однакові інтереси, то система може порекомендувати їм одні й ті ж продукти, незалежно від їх індивідуальних вподобань.

5. Проблема анонімності: у колаборативній фільтрації використовуються дані про поведінку користувачів, що може порушувати їх приватність. Ця проблема може бути вирішена шляхом застосування методів анонімізації даних.

6. Проблема неоднорідності смаків користувачів: колаборативна фільтрація може недостатньо точно враховувати неоднорідність смаків користувачів. Наприклад, коли користувачі можуть віддавати перевагу різним аспектам продукту, таким як ціна, якість, бренд, та інші. Рекомендаційна система повинна бути здатна враховувати ці різні аспекти і ранжувати рекомендації відповідно до індивідуальних вподобань користувача.

7. Проблема кількості рекомендацій: колаборативна фільтрація зазвичай порекомендує лише кілька продуктів, що може бути недостатньо для задоволення потреб користувача. Системи рекомендацій повинні бути здатні рекомендувати достатню кількість продуктів, щоб дати користувачу можливість вибору.

8. Проблема оновлення даних: колаборативна фільтрація базується на історичних даних про взаємодію користувачів з продуктами, що може змінюватися з часом. Рекомендаційні системи повинні бути здатні оновлювати свої дані, щоб враховувати змінені вподобання користувачів і нові продукти на ринку.

9. Проблема рекомендацій для нових користувачів: колаборативна фільтрація може бути недоцільною для рекомендацій новим користувачам, які ще не мали взаємодії з продуктами. Рекомендаційні системи повинні мати можливість використовувати інші джерела даних, такі як демографічні характеристики, щоб зробити персоналізовані рекомендації.

10. Проблема ефективності: колаборативна фільтрація може бути часово витратною та ресурсоємною, особливо при великих масш табах даних. Рекомендаційні системи повинні бути здатні до швидкої обробки та аналізу великих обсягів даних, щоб забезпечити користувачам швидкі та точні рекомендації.

11. Проблема безпеки та приватності: рекомендаційні системи збирають та обробляють великі обсяги особистої інформації про користувачів. Це може створювати проблеми з приватністю та безпекою даних. Рекомендаційні системи повинні мати ефективні механізми захисту даних та забезпечувати конфіденційність користувачів.

12. Проблема оцінки точності: оцінка точності рекомендаційних систем є складним завданням. Колаборативна фільтрація може бути вразлива до різноманітних форм зловживання та маніпуляцій. Рекомендаційні системи повинні мати ефективні механізми оцінки точності рекомендацій та здатні до протидії зловживанням.

1.1.2. Визначення метрик якості системи

Метрики якості систем рекомендацій мають досить велике значення, оскільки на їх основі можна визначити ефективність системи і покращити її функціональність. Основні метрики якості для рекомендаційних систем включають точність, покриття, різноманітність та швидкість.

Точність (precision) визначає, наскільки точно система може рекомендувати користувачеві предмети, які йому дійсно сподобаються. Цю метрику можна вимірювати за допомогою таких показників, як середня або медіана точності рекомендацій.

Покриття (coverage) визначає, який відсоток предметів з БД може бути рекомендований користувачеві. Ця метрика дозволяє визначити, наскільки повністю враховані унікальні інтереси користувача в процесі рекомендацій.

Різноманітність (diversity) оцінює, наскільки різноманітні рекомендації, що надаються користувачеві. Визначення цієї метрики зазвичай базується на показнику різноманітності вхідних даних та способів їх обробки.

Швидкість (speed) визначає, наскільки швидко система може видавати рекомендації користувачеві. Дана метрика важлива для тих сервісів, що працюють в режимі реального часу.

Крім того, можна використовувати метрику "сенситивності", яка вимірює, наскільки сильно змінюються рекомендації при додаванні або видаленні деяких елементів. Ця метрика допомагає визначити, наскільки чутлива система до змін в інтересах користувачів та властивостях предметів.

Також, метрика "досвід" вимірює, наскільки успішно система рекомендацій стимулює користувачів до взаємодії з системою. Наприклад, ця метрика може оцінювати, скільки разів користувачі повертаються до системи після отримання рекомендацій, або скільки разів вони роблять покупки через систему.

Метрика "повноти" вимірює, наскільки повні рекомендації системи, тобто, наскільки вона пропонує всі можливі елементи, які можуть бути цікаві для користувача.

Таким чином метрики якості є важливим інструментом для оцінки ефективності колаборативних систем фільтрації та визначення того, наскільки корисні вони для користувачів. Ось деякі з основних метрик, які можуть бути використані для оцінки системи колаборативної фільтрації:

– точність (accuracy): ця метрика визначає, наскільки точно система колаборативної фільтрації передбачає, які товари будуть подобатися користувачеві. Точність може бути обчислена, порівнюючи рекомендації системи з фактичними покупками або іншими показниками, які вказують на те, чи відповідають рекомендації користувача вибору;

– покриття (coverage): ця метрика визначає, яка частка всіх можливих товарів пропонується користувачам системою. Більші значення цієї метрики означають, що користувачі мають більше можливостей вибору;

– різноманітність (diversity): ця метрика визначає, наскільки різноманітні товари рекомендуються користувачам. Вона може бути обчислена, оцінюючи, наскільки різні товари рекомендує система;

– середня рангова позиція (mean average precision): Ця метрика визначає, як швидко користувачі знаходять товари, які їм сподобалися, серед рекомендацій системи;

– задоволеність користувача (user satisfaction): Ця метрика визначає, наскільки задоволені користувачі системою рекомендацій. Це може бути виміряно через опитування користувачів або відгуки про систему;

– швидкість (speed): Ця метрика визначає, наскільки швидко система колаборативної фільтрації може надавати рекомендації користувачам. Швидкість роботи системи є дуже важливою метрикою якості, оскільки користувачі очікують отримувати рекомендації в режимі реального часу, без зайвої затримки;

– ресурсоємність (resource utilization): ця метрика визначає, скільки ресурсів потрібно для підтримки системи рекомендацій, таких як обсяг даних, кількість користувачів і товарів, потужність серверів тощо;

– інтерпретованість (interpretability): ця метрика визначає, наскільки легко розуміти та пояснювати причини рекомендацій, які надає система колаборативної фільтрації. Наприклад, можливість вказати, які фактори були враховані при рекомендації певного товару;

– робастність (robustness): ця метрика визначає, як система колаборативної фільтрації працює у випадку непередбачуваних ситуацій, наприклад, коли в базі даних відсутні дані про певний товар або користувача.

Важливо вибирати метрики якості, які відповідають конкретним потребам та цілям системи рекомендацій. Також важливо зазначити, що підбір метрик якості може бути викликом, оскільки деякі метрики можуть бути взаємно протирічливими, тому необхідно враховувати весь комплекс метрик при розробці систем рекомендацій.

Важливо, наскільки хороші рекомендації верхніх N елементів списку, де N залежно від завдання варіюється зазвичай у межах від 5 до 20. Тому в метриках ранжування з'являється постфікс @k [5]:

1. Precision@k - ця метрика вимірює відношення кількості релевантних (корисних) рекомендацій, які були серед перших k, до загальної кількості перших k рекомендацій. Значення Precision@k може бути в діапазоні від 0 до 1, де 1 означає, що всі перші k рекомендацій є релевантними, а 0 - що немає жодної релевантної рекомендації серед перших k. Ця метрика особливо важлива в тих випадках, коли важливість порядку рекомендацій важлива.

2. Average Precision@k - ця метрика вимірює середнє значення Precision@k для всіх k, до яких були надані рекомендації. Вона враховує як кількість релевантних рекомендацій, так і їх порядок в списку рекомендацій. Значення Average Precision@k також може бути в діапазоні від 0 до 1, де 1 означає, що всі перші k рекомендацій є релевантними, а 0 - що немає жодної релевантної рекомендації серед перших k.

3. Mean Average Precision@k - ця метрика вимірює середнє значення Average Precision@k для всіх користувачів в системі. Вона дає загальний показник ефективності системи рекомендацій для всієї аудиторії.

4. Normalized Discounted Cumulative Gain - ця метрика враховує не тільки релевантність рекомендацій, але й їх порядок в списку рекомендацій. Вона вимірює вплив порядку рекомендацій на загальну релевантність, тобто

відображає те, що релевантність об'єктів у списку рекомендацій знижується зі збільшенням їх позиції в списку.

Інші метрики, які використовуються в рекомендаційних системах, включають F1-міру, ROC-криву, AUC-ROC (Area Under the ROC Curve) і т.д. Кожна з них вимірює певний аспект ефективності системи рекомендацій і може бути корисною при аналізі її роботи та покращенні результатів.

Кожна з метрик має свої переваги та недоліки, тому вибір метрик для використання залежить від конкретної задачі та особливостей системи рекомендацій.

Метрика Precision@k вимірює частку релевантних предметів серед k рекомендацій. Ця метрика корисна, коли важливим є точне відображення тих предметів, які дійсно цікаві користувачу. Наприклад, в рекомендаційних системах електронної комерції, де важливим є продаж товарів, метрика Precision@k може бути використана для вимірювання точності рекомендацій.

Average Precision@k враховує як порядок, так і частку релевантних предметів серед k рекомендацій. Ця метрика краще підходить для випадків, коли важливим є не тільки точність, але й порядок рекомендацій, наприклад, в музичних сервісах, де порядок виконавців у списку рекомендацій може бути важливим.

Mean Average Precision@k (MAP@k) обчислює середню точність усіх ранжувань серед k рекомендацій для кожного користувача. Ця метрика є більш точною, ніж Average Precision@k, оскільки враховує кількість релевантних предметів серед усіх рекомендацій для кожного користувача. MAP@k може бути використана в рекомендаційних системах, де важливо вимірювати середню точність усіх користувачів.

Normalized Discounted Cumulative Gain (NDCG) враховує не тільки релевантність рекомендацій, але й їх порядок в списку рекомендацій. Вона вимірює вплив порядку рекомендацій на загальну релевантність, тому NDCG може бути використана в системах рекомендацій, де важливим є не тільки пропонувати релевантні предмети, але й те, щоб пропонувати їх у правильному

порядку. Наприклад, в системі музичних рекомендацій користувач може бути зацікавлений у певному жанрі, але бажає послухати не лише відомі хіти, а й менш популярні, проте релевантні композиції. В такому випадку, важливим є те, щоб система пропонувала не лише релевантні композиції, але й правильно їх ранжувала.

Іншим прикладом використання метрики Precision@k може бути система рекомендацій для електронної комерції. Якщо система пропонує користувачеві товари, то важливим є те, щоб користувач купив якомога більше з пропонованих товарів, тобто Precision@k може бути використана для вимірювання ефективності такої системи.

У будь-якій системі рекомендацій важливо визначити метрики якості, які найбільш відповідають її особливостям і завданням. При цьому використання кількох метрик дозволяє отримати більш повну картину про ефективність системи рекомендацій, а також виявити її недоліки та можливості для поліпшення.

Компанії, такі як Google, Facebook, Amazon, Netflix та інші, використовують ці метрики для оцінки своїх систем рекомендацій [11]. У статті [12] представники Airbnb описують використання метрики NDCG для оцінки ефективності своїх рекомендаційних систем у виборі житла для користувачів.

1.2. Принципи роботи колаборативних моделей

Принципи роботи колаборативних моделей ґрунтуються на зборі та аналізі історичних даних взаємодії користувачів з ресурсом. Колаборативні моделі базуються на тому, що користувачі зі схожими інтересами здатні взаємодіяти зі схожими об'єктами, тому якщо один користувач має позитивний досвід з об'єктом, то й інші користувачі зі схожими інтересами мають більшу ймовірність мати такий самий позитивний досвід з цим об'єктом (рис. 1.1).

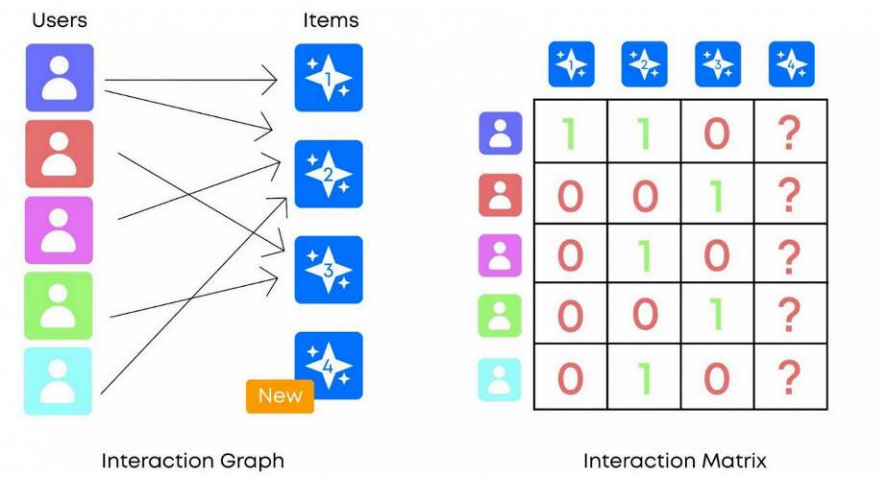

Рисунок 1.1 – Візуалізація користувацького вибору [7]

Основними принципами роботи колаборативних моделей є:

1. Збір даних про взаємодію користувачів з ресурсом. Дані можуть бути зібрані як за допомогою прямого опитування користувачів, так і автоматично з логів взаємодії з ресурсом.

2. Побудова матриці рейтингів. Для кожного користувача та об'єкту будується матриця рейтингів, де в якості значень використовуються оцінки, що виставляються користувачами об'єкту.

3. Визначення схожості користувачів та об'єктів. Для побудови рекомендаційної системи важливо визначити схожість між користувачами та об'єктами. Для цього можуть використовуватися різноманітні методи, наприклад, косинусна схожість, Жаккардова схожість, Евклідова відстань та інші.

4. Побудова моделі. На основі зібраних даних та визначеної схожості користувачів та об'єктів можна побудувати модель, яка буде використовуватися для надання рекомендацій користувачам.

5. Надання рекомендацій. Після побудови моделі, вона може бути використана для надання рекомендацій користувачам. Зазвичай це виконується шляхом вибору певного числа найбільш релевантних або популярних елементів, що були оцінені певним користувачем або групою користувачів.

6. Оцінка точності моделі. Для оцінки точності моделі можуть використовуватися різноманітні метрики, такі як точність, чутливість та

специфічність. Додатково можуть використовуватися метрики, спеціально розроблені для оцінки рекомендаційних систем, такі як Precision@k, Recall@k, MAP@k та NDCG.

7. Оновлення моделі. Колаборативні моделі повинні періодично оновлюватися, оскільки з часом змінюється поведінка користувачів та з'являються нові елементи. Оновлення моделі може включати в себе оновлення даних та перебудову моделі з використанням нових даних.

Узагальнюючи, колаборативні моделі є ефективним інструментом для рекомендаційних систем, оскільки вони дозволяють автоматично визначати відносну релевантність елементів для користувачів на основі їх попередніх дій. При цьому важливими принципами роботи таких моделей є використання матриці оцінок користувачів та елементів, побудова моделі на основі рекомендаційних алгоритмів, надання рекомендацій та оцінка точності моделі.

1.2.1. Класична постановка задачі для рекомендаційної системи

Класична постановка задачі для рекомендаційної системи полягає в тому, щоб запропонувати користувачеві список рекомендованих елементів, які він ще не бачив, на основі його історії споживання або споживання схожих користувачів. Конкретно, система повинна здійснювати наступні кроки:

1. Зібрати вхідні дані про користувачів та елементи. Це можуть бути дані про дії користувачів (наприклад, перегляди відео, покупки товарів тощо) та характеристики елементів (наприклад, жанр відео, категорія товару тощо).

2. Створити матрицю оцінок, яка відображає взаємодію між користувачами та елементами. Кожен елемент може бути оцінений декількома користувачами, а кожен користувач може оцінювати декілька елементів.

3. Розбити матрицю на дві підматриці - тренувальну та тестову. Тренувальна матриця використовується для навчання моделі, тестова матриця використовується для перевірки якості рекомендаційної системи.

4. Навчити модель на тренувальній матриці. Колаборативні моделі можуть бути засновані на методах розкладання матриці (наприклад, Singular Value

Decomposition), методах згорткової нейронної мережі (наприклад, Convolutional Neural Networks) та інших методах.

5. Надати рекомендації на основі навченої моделі. Для кожного користувача рекомендуються ті елементи, для яких модель передбачає найвищу ймовірність за наявною історією взаємодії користувача та елемента.

Узагальнення, яке передбачає відсутність взаємодії з негативним прикладом (рис. 1.2).

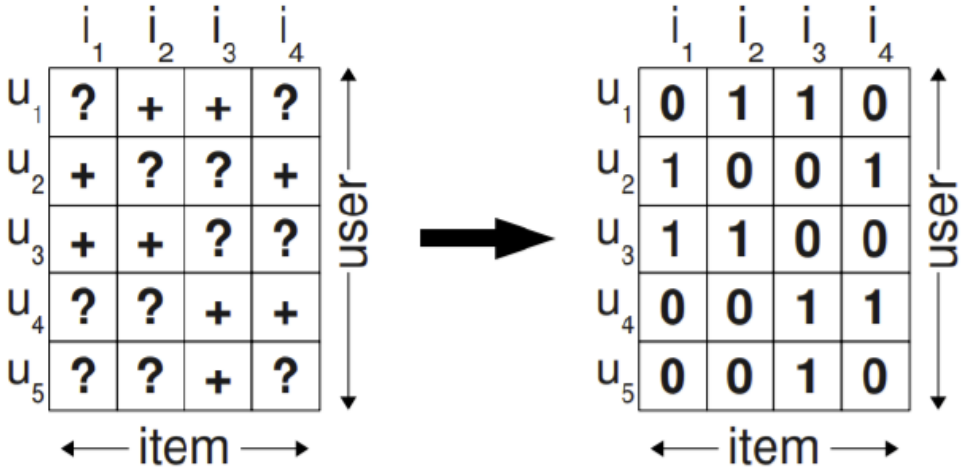

Рисунок 1.2 –Приклад матриці оцінок [8]

Збір трійки параметрів – користувач, позитивний айтем, негативний айтем може бути представлено на рис. 1.3 [9].

Bayesian Personalized Ranking (BPR) є одним з методів навчання рекомендаційних систем на основі implicit feedback. Він ґрунтується на підході байєсівської вибірки та використовує техніку позитивної та негативної зразків для навчання моделі. BPR намагається навчити модель ранжування, яка може відрізняти позитивні приклади від негативних, і при цьому мінімізувати кількість помилкових рекомендацій.

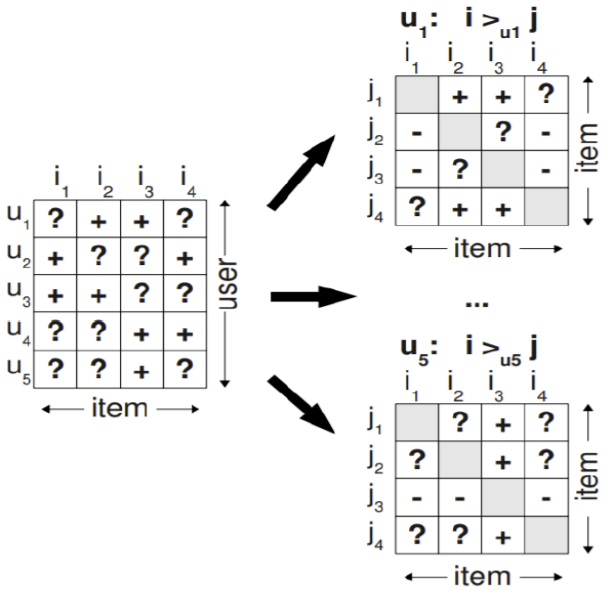

Рисунок 1.3 – Модифікація матриці оцінок [8]

Інші методи навчання для implicit feedback включають:

1. Matrix Factorization (MF): цей метод базується на розкладанні матриці взаємодії користувачів та об'єктів на дві матриці меншого розміру, які репрезентують користувачів та об'єкти у просторі складових. MF намагається навчити модель, яка може передбачити рейтинги, що користувачі присвоюють об'єктам.

2. Factorization Machines (FM): цей метод використовує крос-валідацію та розкладання матриці, щоб побудувати модель, яка може передбачати рейтинги. FM може обробляти як explicit, так і implicit feedback, і навчання відбувається з використанням градієнтного спуску.

3. Neural Networks (NN): цей метод використовує глибокі нейронні мережі для навчання рекомендаційної системи. Це дозволяє моделі враховувати складні залежності між користувачами та об'єктами. NN може обробляти як explicit, так і implicit feedback, і він може бути навчений за допомогою методів навчання, таких як backpropagation.

Кожен з цих методів має свої переваги та недоліки, і вибір підходу для конкретної рекомендаційної системи залежить від багатьох факторів, включаючи характер.

Якщо вважати відсутність взаємодії негативним прикладом, то це може призвести до того, що в матриці оцінок буде багато пропущених значень, які можуть зіскочити з поля зору при розробці рекомендаційної системи. Крім того, це може призвести до зменшення точності рекомендаційної системи, оскільки вона не отримує достатньо інформації про вподобання користувачів. Для зменшення кількості пропущених значень в матриці оцінок можна використовувати методи заповнення пропусків, наприклад, методи ближніх сусідів або методи факторизації матриць. Також можна використовувати альтернативні джерела даних для отримання інформації про користувачів і їхні вподобання, такі як соціальні мережі або інформація про покупки.

Adaptive learning rate - це техніка оптимізації, яка дозволяє динамічно змінювати швидкість навчання моделі в залежності від градієнта функції втрат. Для цього можна використовувати алгоритм Adam (Adaptive Moment Estimation), який забезпечує більш швидку збіжність моделі при навчанні.

Оцінювання навченості системи можна здійснювати за допомогою кількох метрик. Одна з них - це mean average precision (MAP), яка вимірює середню точність рекомендацій. Іншою метрикою може бути normalized discounted cumulative gain (NDCG), яка враховує не тільки релевантність рекомендацій, але й їх порядок в списку рекомендацій.

Для оцінювання навченості системи можна використовувати крос-валідацію, де дані діляться на тренувальну і тестову вибірки. Потім модель навчається на тренувальній вибірці, а її ефективність оцінюється на тестовій вибірці. Цей процес повторюється кілька разів для різних комбінацій тренувальної і тестової вибірок, щоб знизити вплив випадковості на результати.

Для виправлення цієї ситуації можна використовувати алгоритми, що забезпечують диверсифікацію рекомендацій. Один з них - рекомендації на основі змішаної лінійної моделі (mixed linear model recommendations). Цей алгоритм дозволяє забезпечити баланс між популярним і менш популярним контентом, і забезпечує диверсифікацію рекомендацій шляхом використання змішаної моделі рекомендацій. За допомогою цього алгоритму можна забезпечити рівномірну

експозицію різноманітного контенту, що забезпечує якість рекомендацій та забезпечує зростання популярності менш популярного контенту.

Exploitation та exploration - це дві взаємопов'язані стратегії, що використовуються в контексті рекомендаційних систем.

Exploitation - це стратегія, за якою система намагається максимізувати вигоду, рекомендуючи користувачеві ті айтеми, які він вже переглядав або купував раніше. Ця стратегія ґрунтується на ідеї, що користувачі будуть продовжувати використовувати ті айтеми, які їм подобаються, і що рекомендації, які враховують ці уподобання, будуть більш ефективними.

Exploration - це стратегія, за якою система намагається залучити користувача до перегляду нових айтемів, що він може ще не бачив, але які можуть йому сподобатися. Ця стратегія має на меті збільшити різноманітність рекомендацій, щоб залучити користувачів до більш широкого спектру айтемів та збільшити ймовірність знаходження користувачем нових айтемів, які йому сподобаються.

Залежно від контексту та потреб користувачів, рекомендаційні системи можуть використовувати різні комбінації цих двох стратегій для досягнення оптимального результату (рис. 1.4). На початку взаємодії з новим користувачем система може використовувати більш активну стратегію exploration для залучення його до нових айтемів, а після того, як вона збереже достатньо даних про користувача, перейти до стратегії exploitation, щоб рекомендувати йому ті айтеми, які йому сподобаються найбільше [10, 11].

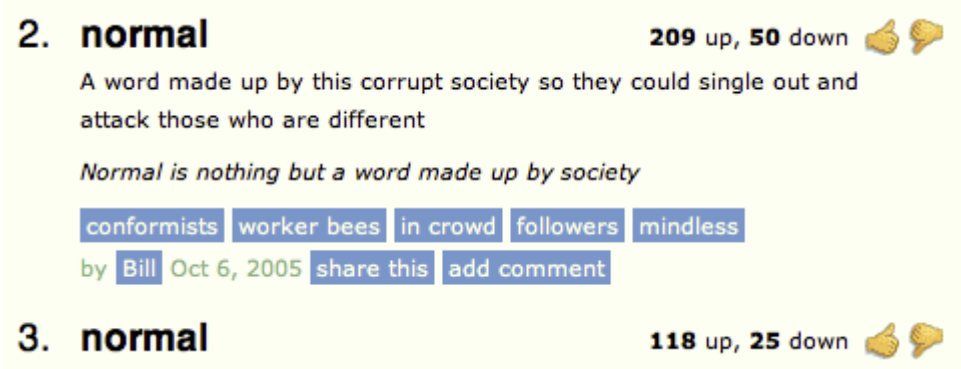

Рисунок 1.4 – Сортування за ознакою «популярність»

$$\text{Score} = (\text{Positive ratings}) - (\text{Negative ratings}) \quad (1.1)$$

$$\text{Score} = (\text{Positive ratings}) / (\text{Total ratings}) \quad (1.2)$$

Формули (1.1) та (1.2) є методами рейтингування товарів на основі оцінок користувачів. Формула (1.1) розраховує рейтинг товару на основі різниці між кількістю позитивних та негативних оцінок. Значення рейтингу буде вище, якщо товар має більше позитивних оцінок і менше негативних. Формула (1.2) розраховує рейтинг товару на основі кількості позитивних оцінок, поділеної на загальну кількість оцінок. Значення рейтингу буде вище, якщо товар має більше позитивних оцінок від загальної кількості оцінок.

Для врахування кількості оцінок можна використовувати інший показник рейтингу. Наприклад, такий показник можна розрахувати як середнє значення оцінок, взяте з коефіцієнтом ваги, що залежить від кількості оцінок. Це дозволяє враховувати як якість оцінок, так і їх кількість, що дає більш об'єктивну оцінку товару.

1.2.2. Принципи персоналізованої рекомендації

Персоналізовані рекомендації - це рекомендації, які враховують індивідуальні потреби та інтереси кожного користувача. Це робиться шляхом аналізу взаємодії користувача з системою та збору додаткової інформації про користувача.

У персоналізованих рекомендаційних системах використовуються різні методи та алгоритми, щоб зробити рекомендації якнайбільш релевантними для кожного користувача. Для цього можуть бути використані такі методи, як:

1. Колаборативний фільтринг: метод, який використовує спільність відгуків користувачів для визначення рекомендацій. Цей метод шукає схожість між користувачами та рекомендує їм товари, які сподобалися їхнім схожим користувачам.

2. Content-based підхід: метод, який використовує аналіз властивостей предметів та відповідних їм властивостей користувачів. Цей метод рекомендує

користувачеві товари на основі співставлення властивостей предметів та відповідних їм властивостей користувача.

3. Гібридний підхід: комбінація колаборативного та content-based підходів для отримання кращих результатів. Цей метод використовує інформацію про користувача, їхню взаємодію з системою та властивості предметів для надання персоналізованих рекомендацій.

Персоналізовані рекомендації є ефективним інструментом для поліпшення взаємодії користувача з системою та підвищення рівня задоволення користувача (рис. 1.5).

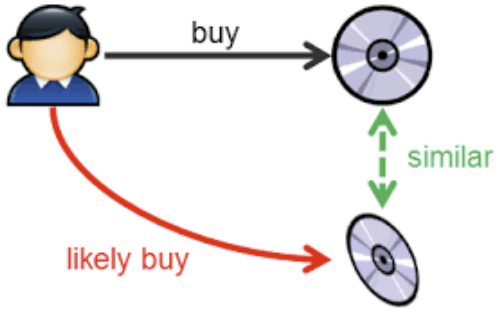

Рисунок 1.5 – Візуалізація персоналізованої рекомендації

1.2.3. Матриці факторизації

Матрична факторизація - це метод розбиття матриці на декілька складових частин, зазвичай дві, з метою зменшення розмірності матриці та виявлення схожості між рядками та стовпцями. У контексті рекомендаційних систем, матрична факторизація використовується для розкладання матриці оцінок користувачів та айтемів на складові частини, що дозволяє знайти складові, які пов'язують ці дві матриці.

Метод матричної факторизації може бути застосований як для колаборативної фільтрації, так і для контент-базованих рекомендаційних систем. У випадку колаборативної фільтрації, це дозволяє знайти приховані характеристики користувачів та айтемів, які впливають на їхню взаємодію. У випадку контент-базованих рекомендаційних систем, це дозволяє знайти приховані характеристики айтемів та користувачів, які можуть бути використані для знаходження рекомендацій.

Метод матричної факторизації може бути реалізований за допомогою різних алгоритмів, таких як Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), Alternating Least Squares (ALS) та інших. Він є ефективним методом для розробки персоналізованих рекомендаційних систем, оскільки може обробляти великі обсяги даних та здійснювати швидкі рекомендації з високим рівнем точності.

Можна розкладати на дві матриці цей обрахунок (рис. 1.6).

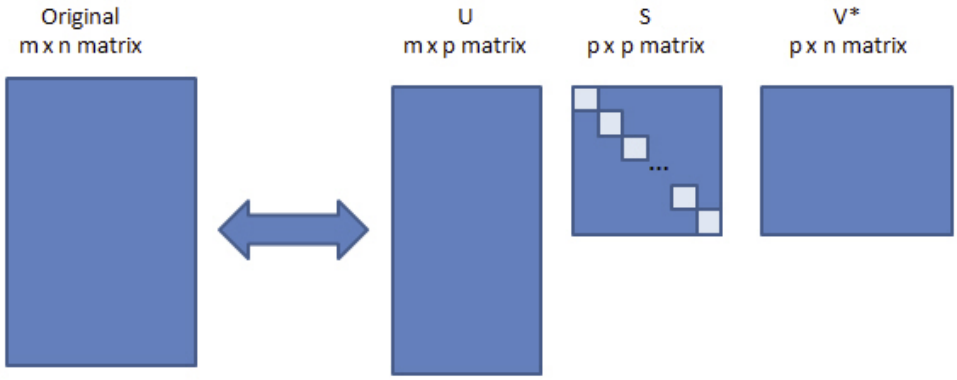

Рисунок 1.6 – Візуалізація розкладання матриці оцінок

1.2.4. Аналіз використання нейромереж в системах рекомендацій

Нейромережі є популярним інструментом у сучасних системах рекомендацій. Застосування нейромереж в рекомендаційних системах може покращити точність рекомендацій, оскільки вони можуть виявляти складні шаблони і залежності між даними.

Одним з найпоширеніших методів з використанням нейромереж є глибокі нейронні мережі (deep neural networks, DNN). DNN зазвичай використовуються для побудови складних моделей, які включають багатошарові архітектури, що дозволяє враховувати багато параметрів.

Також існують різноманітні варіації нейромереж, які можуть бути застосовані в рекомендаційних системах, наприклад, рекурентні нейронні мережі (recurrent neural networks, RNN), надглибокі нейронні мережі (very deep neural networks, VDNN), нейронні мережі з підсиленням (reinforcement neural networks, RNN) та інші.

Нейромережі можуть бути використані для різних задач, таких як ранжування, класифікація, кластеризація та прогнозування. Для рекомендаційних систем, можна використовувати нейромережі для моделювання складних залежностей між користувачами та предметами, що дозволяє точніше передбачати інтереси користувачів та відповідні для них товари.

1.3 Постановка завдання на випускну кваліфікаційну роботу

Постановка завдання на випускну кваліфікаційну роботу полягає у розробці та впровадженні рекомендаційної системи для музичного сервісу. Метою проекту є покращення користувацького досвіду та збільшення задоволення користувачів, шляхом надання персоналізованих рекомендацій на основі їхніх музичних вподобань.

Основні завдання проекту включають:

1. Аналіз вимог та потреб користувачів музичного сервісу: дослідження музичних переваг користувачів, стилів, жанрів та інших важливих факторів, які впливають на вподобання.

2. Розробка алгоритмів колаборативної фільтрації та гібридного підходу: реалізація методу Slope One, який дозволє ефективно побудовувати рекомендації на основі взаємодії користувачів.

3. Розробка модуля фільтрації рекомендацій: створення програмного модуля, який інтегрує розроблені алгоритми та забезпечує роботу системи рекомендацій на музичному сервісі. Модуль повинен мати зручний інтерфейс для користувачів та надавати персоналізовані рекомендації.

4. Валідація та тестування системи: проведення випробувань та аналіз результатів рекомендаційної системи з використанням реальних даних та користувачів. Оцінка якості рекомендацій, порівняння з існуючими методами та аналіз задоволення користувачів.

5. Документування та підготовка звіту: створення документації, яка описує процес розробки та реалізації рекомендаційної системи, включаючи детальні описи алгоритмів, використані технології, аналіз результатів, тестування та оцінку ефективності системи. Також підготовка презентації та захисту проекту перед комісією.

Основна цінність розробленої рекомендаційної системи полягає в поліпшенні користувацького досвіду і забезпеченні персоналізованих рекомендацій, що відповідають індивідуальним музичним вподобанням кожного користувача.

Висновки за розділом

У розділі 1 було досліджено принципи побудови рекомендаційних систем. Було розглянуто різні методики рекомендацій, зокрема колаборативні та контентні, а також їхні переваги та недоліки. Також було розглянуто методи формування train-набору даних, зокрема skip-gram та CBOW. Було детально розглянуто такі методи як матрична факторизація та neural collaborative filtering.

Також були висвітлені питання щодо використання нейромереж в системах рекомендацій та методики "багаторукі бандити". Було досліджено проблему рідкісних множин та показано, як вона може бути вирішена за допомогою алгоритмів Apriori та FP-growth.

Показано, що побудова ефективної рекомендаційної системи вимагає збору великої кількості даних та застосування різних методик та алгоритмів машинного навчання. Найкращі результати можуть бути досягнуті шляхом поєднання декількох підходів та аналізу результатів за допомогою метрик якості.

На основі проведеного аналізу було визначено завдання на випускну кваліфікаційну роботу.

Розділ 2
Моделювання роботи модуля колаборативної фільтрації рекомендаційної музичної системи

За допомогою аналізу та експериментів, покажемо, як можливо поліпшити якість рекомендацій за допомогою різних методів та параметрів моделювання. У цьому розділі розглянемо відповідні моделі, що використовуються для рекомендацій та способи їх вдосконалення, враховуючи особливості музичного контенту та проведемо базове модулювання роботи модуля колаборативної фільтрації.

Цінність моделювання полягає у поглибленні знань про роботу колаборативної фільтрації та впровадженні її в рекомендаційну музичну систему, що дозволить поліпшити якість рекомендацій та підвищити рівень задоволення користувачів. Крім того, результати дослідження можуть бути застосовані в інших галузях, де використовуються рекомендаційні системи на базі колаборативної фільтрації.

Метою моделювання роботи модуля колаборативної фільтрації рекомендаційної музичної системи є дослідження і розробка алгоритмів колаборативної фільтрації для рекомендацій музичних композицій, що дозволяє покращити якість рекомендацій за рахунок використання інформації про попередні взаємодії користувачів з системою.

2.1. Основні алгоритми для побудови комп’ютерних систем формування рекомендацій

Основні алгоритми для побудови комп’ютерних систем формування рекомендацій можуть бути класифіковані за різними критеріями, наприклад, за типом вхідних даних, за типом рекомендацій (контентні або колаборативні) тощо. У цьому розділі ми розглянемо деякі з основних алгоритмів для побудови систем рекомендацій.

1. Learn to Rank - це метод машинного навчання, який використовується для вирішення задач ранжування. Він зазвичай використовується для побудови рекомендаційних систем, де метою є забезпечення користувачам списку рекомендованих елементів в порядку спадання ймовірності зацікавлення.

2. Sequential - це метод побудови рекомендацій, який використовує інформацію про послідовність взаємодій користувача з системою для формування рекомендацій. Він використовується в системах рекомендацій для забезпечення персоналізованих пропозицій на основі історії взаємодій користувача з системою.

3. Variational Autoencoders - це метод глибинного навчання, який використовується для автоматичного екстрагування факторів, що визначають кореляції між об'єктами. Використовується для побудови рекомендаційних систем, де метою є знаходження подібних елементів на основі семантичних зв'язків.

4. Reinforcement Learning - це метод машинного навчання, який використовується для навчання системи рекомендацій, яка взаємодіє з користувачем в режимі реального часу. Він використовується для забезпечення персоналізованих рекомендацій, які відповідають поточному стану користувача. Цей метод машинного навчання використовується для навчання систем рекомендацій, які взаємодіють з користувачем в режимі реального часу. Він базується на ідеї навчання агента приймати рішення на основі нагороди (reward), яку він отримує за свої дії.

У контексті рекомендаційних систем, агентом може бути модель, яка відповідає за підбір рекомендацій для користувача, а нагородою може бути задоволення користувача від отриманих рекомендацій. Задача моделі полягає в тому, щоб максимізувати загальну нагороду, отримуючи відгуки від користувача за кожну рекомендацію.

Reinforcement Learning може використовуватися для забезпечення персоналізованих рекомендацій, які відповідають поточному стану користувача. Наприклад, якщо користувач виявляє інтерес до певного жанру музики після

декількох отриманих рекомендацій, модель може змінити свої підходи до рекомендацій і більше уваги приділяти рекомендаціям з цього жанру.

Однак, використання Reinforcement Learning для побудови систем рекомендацій вважається складним завданням, оскільки вимагає багато даних і складних алгоритмів для оцінки нагороди. Також, необхідно враховувати етичні та приватність питання, що стосуються взаємодії з користувачами в режимі реального часу.

2.1.1. Алгоритми Learning to Rank

Алгоритми Learning to Rank (LTR) - це методи машинного навчання, які використовуються для ранжування великої кількості елементів відповідно до їхньої важливості. У контексті систем рекомендацій, LTR може бути використаний для забезпечення кращого ранжування рекомендацій в залежності від персоналізованих вимог користувача.

Головна відмінність LTR від традиційного ML полягає в тому, що в традиційному ML завдання полягає в передбаченні точної величини цільової змінної, тоді як в LTR сподіваємося отримати список об'єктів у відповідності з їхньою важливістю для користувача. Іншими словами, LTR не передбачає точну величину цільової змінної, а замість цього він навчається виробляти правильний порядок ранжування для кожного користувача.

Для досягнення цього, LTR використовує різні фактори, такі як популярність, частоту взаємодії користувача з об'єктом, або контекстуальні фактори, щоб визначити важливість кожного об'єкту для користувача. Після навчання модель може бути використана для ранжування об'єктів для кожного користувача, що дає змогу забезпечити більш персоналізовані рекомендації.

Одним з популярних алгоритмів LTR є LambdaMart, який використовує градієнтний бустинг для навчання моделі ранжування. Для цього він використовує функцію втрати, яка відображає відстань між реальним порядком об'єктів і порядком, передбаченим моделлю [19].

Одна з головних відмінностей LTR від традиційного машинного навчання полягає в тому, що вона не навчає модель передбачати значення цільової змінної, а навчає модель передбачати порядок об'єктів за їхнім рівнем релевантності. Це дозволяє отримувати більш точні рекомендації, оскільки порядок, у якому пропонуються об'єкти, може бути важливим для користувача.

Загальна схема роботи алгоритму LambdaMart полягає в тому, що спочатку для кожного користувача створюється список об'єктів з їх рівнем релевантності. Потім використовуючи цей список, алгоритм навчає модель передбачати порядок об'єктів за рівнем їхньої релевантності. Після навчання моделі за допомогою градієнтного бустингу, вона може бути застосована для ранжування нових об'єктів для користувачів.

На рис. 2.1 представлені списки популярних LTR-алгоритмів.

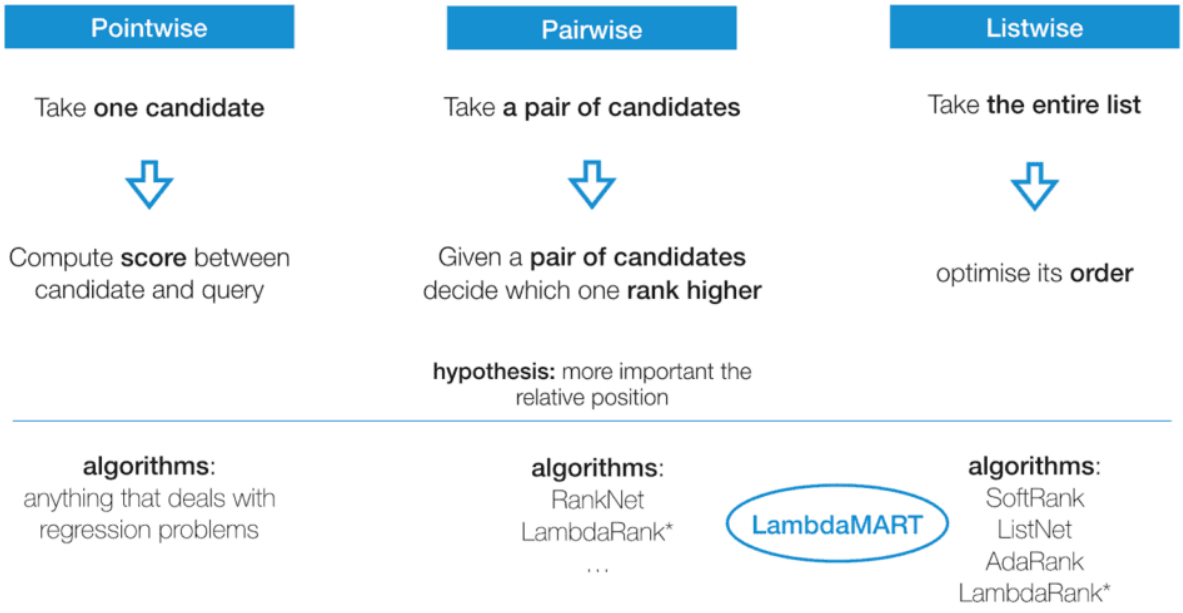

Рисунок 2.1 – Списки популярних LTR-алгоритмів [21]

Отже, є три основні підходи до побудови ранжування:

1. Pointwise - цей підхід перетворює задачу ранжування в задачу регресії або класифікації, де кожен об'єкт ранжується окремо. У цьому випадку модель приймає на вхід властивості окремого об'єкта та виводить його рейтинг.

2. Pairwise - цей підхід розглядає пари об'єктів і порівнює їх за якоюсь метрикою. Метрика використовується для порівняння рейтингів двох об'єктів, і модель навчається визначати, який з об'єктів має вищий рейтинг.

3. Listwise - цей підхід вважає, що ранжування повинно відбуватися як список, а не окремі об'єкти або пари. Модель приймає на вхід список об'єктів та виводить їх порядок за якими рекомендується їх споживачеві.

Компанія Google розробила комбіновану модель для рекомендацій додатків у магазині Google Play в 2016 році. Ця модель об'єднує в собі як колаборативну фільтрацію, так і контентні методи рекомендацій.

Колаборативна фільтрація заснована на спільній історії взаємодії користувачів з додатками. Контентні методи, з іншого боку, використовують описи додатків та іншу інформацію про них для знаходження спільних рис між додатками та користувачами.

В основі цієї комбінованої моделі лежить нейромережа з архітектурою Wide & Deep, яка складається з двох частин. Wide-частина містить лінійну модель з декількома взаємодіючими ознаками, що дозволяє моделі враховувати спільні риси між користувачами та додатками. Deep-частина містить нейромережу, що навчається на основі інформації про користувачів та додатки, що дозволяє моделі враховувати контекстну інформацію та особисті вподобання користувачів.

Ця комбінована модель рекомендацій дозволяє отримувати якісні та персоналізовані рекомендації додатків для користувачів магазину Google Play. Додатково, вона дозволяє покращити збір даних про користувачів та їхні вподобання, що забезпечує змогу додаткового аналізу та вдосконалення рекомендацій в майбутньому. (рис. 2.2).

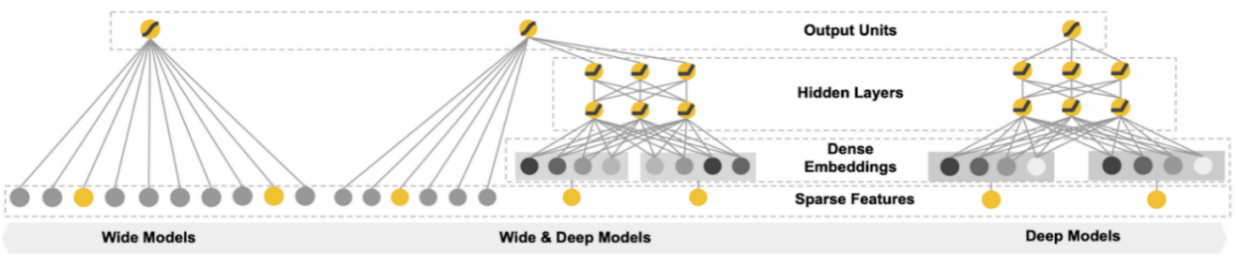

Рисунок 2.2 – Візуалізація моделі Google Play

2.1.2. Моделі оцінки вхідних змінних

Wide & Deep Learning - це модель машинного навчання, яка поєднує в собі лінійну регресію та нейронні мережі з декількома шарами. Модель була розроблена Google для вирішення задач рекомендацій в Google Play.

Основною ідеєю Wide & Deep Learning є комбінування моделі, яка заснована на вивченні зв'язків між ознаками (wide component) та глибокою нейронною мережею, яка вивчає складні залежності між вхідними змінними (deep component). Wide & Deep Learning є розширенням логістичної регресії та використовується для розробки персоналізованих рекомендацій в широкому спектрі додатків та сервісів.

Wide Component - це модель, яка заснована на вивченні зв'язків між ознаками. Wide Component навчається за допомогою логістичної регресії та використовується для визначення кореляції між двома різними елементами. Наприклад, в залежності від того, чи купував користувач попередньо піцу, система може порекомендувати йому гарячі напої.

Deep Component - це глибока нейронна мережа, яка вивчає складні залежності між вхідними змінними. Deep Component використовується для вивчення залежностей між елементами з великою кількістю параметрів. Система може використовувати Deep Component, щоб вивчити залежності між кількістю годин, проведених користувачем в спортзалі, та його вагою. Комбінування Wide Component та Deep Component дозволяє моделі Wide & Deep Learning використовувати лінійну регресію та нейронні мережі одночасно, що дозволяє отримувати більш точні та персоналізовані результати (рис. 2.3).

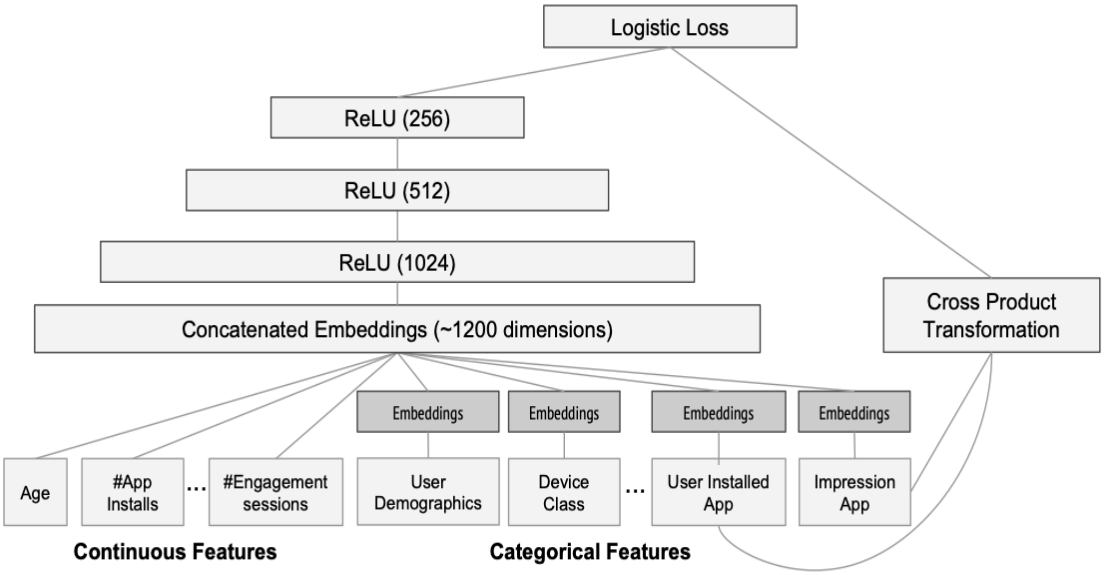

Рисунок 2.3 – Візуалізація моделі Deep Component [21]

2.1.3. Приклади реалізації алгоритмів для систем рекомендацій

Приклади реалізації рекомендаційних алгоритмів Self-Attentive Sequential Recommendation (SASRec) , Fusing Item Similarity Models with Self-Attention (FISSA), Collaborative Memory Networks (CMN), Collaborative Variational Autoencoder (CVAE) можна описати наступним чином:

1. Self-Attentive Sequential Recommendation (SASRec): Цей алгоритм використовує механізм self-attention для побудови рекомендацій на основі послідовності попередніх дій користувача. Він використовує підходи з області нейромереж та навчання без вчителя, щоб розуміти поведінку користувача. SASRec показав дуже хороші результати на деяких даних порівняно з іншими методами (рис. 2.4).

2. Fusing Item Similarity Models with Self-Attention (FISSA): Цей алгоритм поєднує в собі методи рекомендації на основі подібності предметів з механізмом self-attention. Він розглядає кожен предмет з точки зору його подібності до інших предметів, а також з точки зору його важливості для користувача. В результаті, FISSA (рис. 2.5) здатний надавати більш персоналізовані рекомендації, які враховують як подібність предметів, так і особисті інтереси користувача.

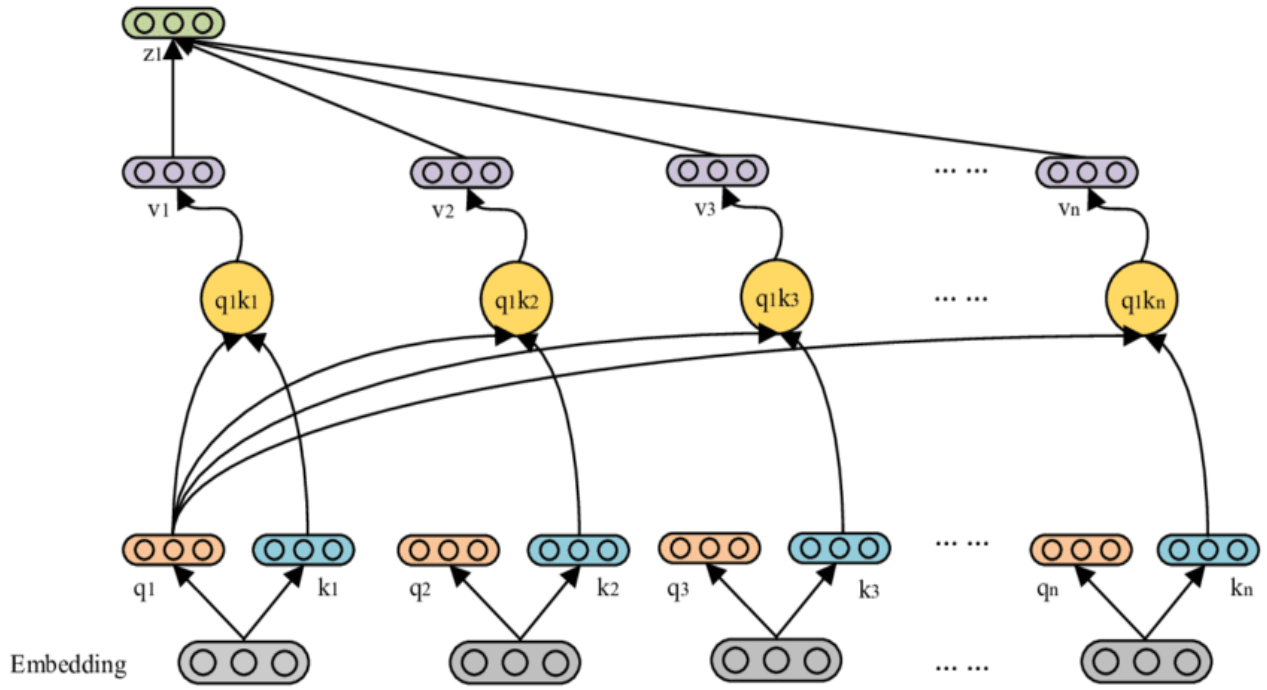

Рисунок 2.4 – Візуалізація роботи алгоритму SASRec [24]

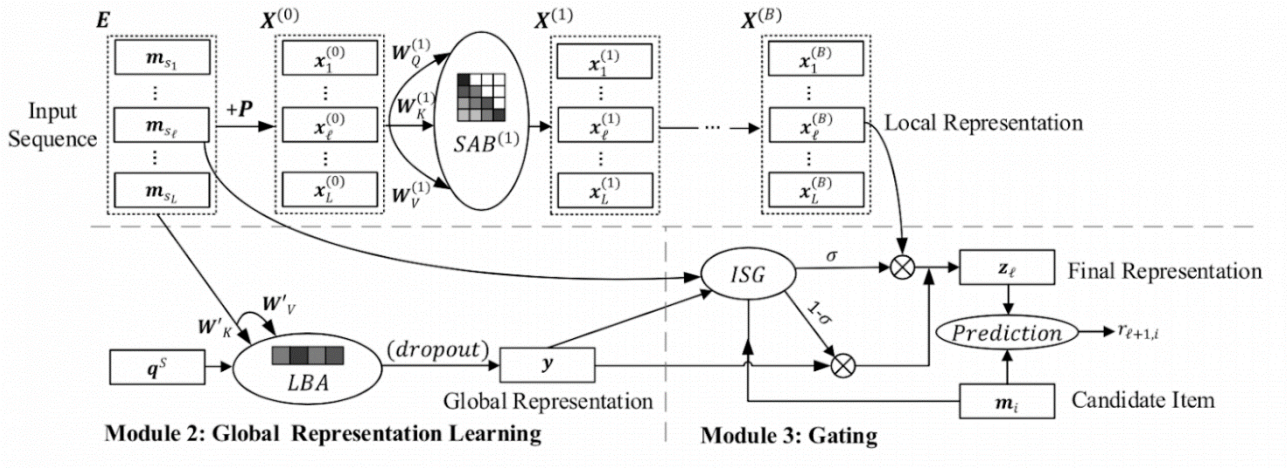

Рисунок 2.5 – Метод реалізації моделі FISSA [25]

3. Collaborative Memory Networks (CMN): Цей алгоритм використовує пам'ять, щоб зберігати інформацію про взаємодію користувачів з предметами. Він здатен зберігати інформацію про те, які предмети користувачі спільно використовують, та враховувати цю інформацію при рекомендації нових предметів. CMN може підтримувати більш складні зв'язки між користувачами та предметами, що дозволяє забезпечити більш точні та персоналізовані рекомендації.

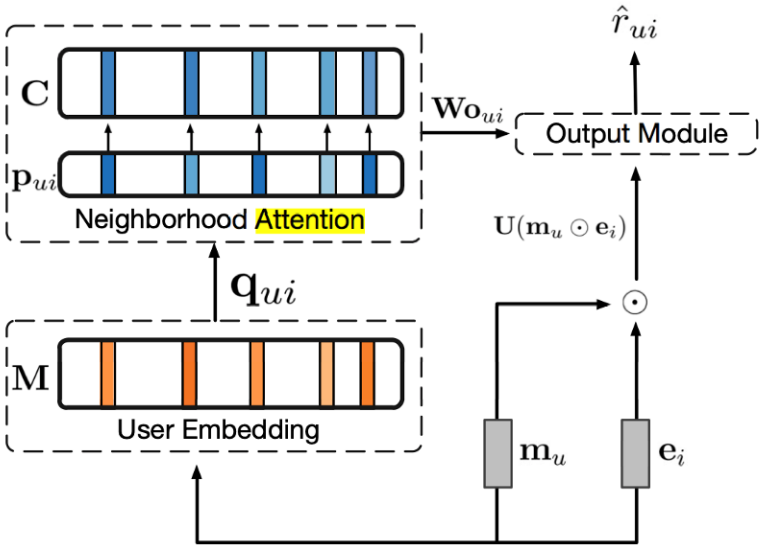

Рисунок 2.6 – Метод реалізації моделі CMN [27]

4. Collaborative Variational Autoencoder (CVAE) - цей алгоритм використовує варіаційний автоенкодер для побудови рекомендацій. Він навчається моделювати складні залежності між користувачами та предметами, що дозволяє забезпечити більш персоналізовані рекомендації. Зокрема, CVAE може робити рекомендації для користувачів, які мають дуже мало інформації про свої вподобання.

Багато рекомендаційних моделей так чи інакше приходять до ідеї гібридних моделей. Ймовірно, причина в тому, що хороша рекомендаційна система повинна підходити великому числу різних у поведінці користувачів і вирішувати одночасно кілька завдань (які вже перераховувалися в першій частині).

2.2. Моделювання бізнес-процесів і архітектури модуля колаборативної фільтрації

Проведений аналіз існуючих систем показав, що модуль колаборативної фільтрації повинен містити наступні бізнес-процеси:

1. Збір даних: модуль колаборативної фільтрації має збирати дані вподобання об'єктів.

2. Обробка даних: після збору даних, модуль колаборативної фільтрації повинен провести попередню обробку даних. Це може включати чистку та нормалізацію даних, виявлення аномалій, статистичний аналіз даних та вибір методів аналізу.

3. Знаходження схожих користувачів або об'єктів: для здійснення рекомендацій модуль колаборативної фільтрації повинен визначити, які користувачі та об'єкти є схожими між собою. Це можна зробити, використовуючи методи порівняння даних, які можуть включати визначення схожості профілів користувачів.

4. Генерація рекомендацій: після знаходження схожих користувачів або об'єктів, модуль колаборативної фільтрації генерує рекомендації для користувачів. Це може бути здійснено за допомогою методів, таких як ранжування та фільтрація результатів, щоб забезпечити користувачам оптимальні рекомендації.

5. Оновлення моделі: модуль колаборативної фільтрації повинен періодично оновлювати модель, використовуючи нові дані та виправляючи помилки. Це може включати періодичну перетренування моделі на нових даних та використання алгоритмів для знаходження невідповідностей в даних та їх виправлення.

6. Розгортання та моніторинг: після побудови моделі колаборативної фільтрації та здійснення перевірки її ефективності, модуль повинен бути розгорнутий в реальному середовищі та підтримуватися. Важливо моніторити ефективність модуля та виявляти можливі проблеми, щоб оперативно їх виправити та забезпечити максимальну ефективність системи.

Загалом, модуль колаборативної фільтрації повинен бути розроблений з урахуванням потреб користувачів та бізнес-вимог, забезпечуючи максимальну ефективність та точність рекомендацій. Тому важливо ретельно проаналізувати бізнес-процеси та вимоги перед розробкою модуля, щоб забезпечити максимальну ефективність та корисність системи для користувачів та бізнесу.

Основними компонентами модуля колаборативної фільтрації є:

1. База даних: компонент для зберігання даних про оцінки користувачів на об'єкти. База даних може бути реляційною або нереляційною, залежно від потреб проекту.

2. Алгоритми знаходження схожості: компонент, який використовується для знаходження схожих користувачів або об'єктів. Цей компонент може включати алгоритми порівняння даних, такі як косинусна схожість, евклідова відстань тощо.

3. Алгоритми ранжування та фільтрації: компонент для генерації рекомендацій та їх відображення користувачам. Цей компонент може використовувати алгоритми ранжування, такі як рейтинговий метод, або фільтрації, такі як колаборативна фільтрація з виключенням.

4. Модуль оновлення моделі: компонент для періодичного оновлення моделі на основі нових даних та виправлення помилок. Цей компонент може використовувати алгоритми машинного навчання та статистичного аналізу для покращення ефективності моделі.

5. Модуль моніторингу: компонент для моніторингу ефективності та виявлення можливих проблем у системі. Цей компонент може включати механізми логування та аналізу даних, щоб виявляти аномалії та допомагати у вирішенні проблем.

Зв'язки між компонентами наступні:

– Компонент бази даних використовується для зберігання даних про оцінки користувачів на об'єкти, які використовуються іншими компонентами.

– Компонент алгоритмів знаходження схожості використовує дані з бази даних, щоб знаходити схожі користувачів або об'єкти, які потім використовуються компонентами алгоритмів ранжування та фільтрації для генерації рекомендацій.

– Компонент алгоритмів ранжування та фільтрації отримує вхідні дані від компонента алгоритмів знаходження схожості та використовує їх для генерації рекомендацій для користувачів.

– Компонент модуля оновлення моделі використовує дані з бази даних та алгоритмів знаходження схожості, щоб періодично оновлювати модель та виправляти помилки.

– Компонент модуля моніторингу використовує дані з компонента логування, щоб моніторити ефективність системи та виявляти можливі проблеми, які потрібно вирішувати.

Загалом, зв'язки між компонентами модуля колаборативної фільтрації залежать від конкретної реалізації системи та можуть бути різними. Проте, важливо мати на увазі, що всі компоненти повинні працювати разом, щоб забезпечити ефективну та точну систему рекомендацій.

2.2.1. Дерево функцій модуля колаборативної фільтрації в системі рекомендацій

Дерево функцій модуля колаборативної фільтрації включає наступні функції:

1. Збір даних:

– Збір вподобань користувачів на об'єкти;

– Збір інформації про перегляди кліпів;

2. Обробка даних:

– Чистка та нормалізація даних;

– Виявлення аномалій у даних;

– Статистичний аналіз даних;

– Вибір методів аналізу даних;

3. Знаходження схожих користувачів та об'єктів:

– Аналіз схожих оцінок користувачів;

– Визначення схожості профілів користувачів;

– Аналіз інформації про перегляди кліпів;

4. Генерація рекомендацій:

– Ранжування результатів на основі схожості користувачів та об'єктів;

– Фільтрація результатів з урахуванням інших факторів (рейтинг об'єктів);

5. Оновлення моделі:

– Періодичне перетренування моделі на нових даних;

– Виправлення помилок у даних та моделі;

6. Розгортання та моніторинг:

– Розгортання модуля в реальному середовищі;

– Моніторинг ефективності та виявлення проблем.

Кожна з функцій може мати додаткові підфункції та вимоги до даних, що потрібні для її виконання. Окрім того, деякі функції можуть бути виконані паралельно або у зворотньому порядку в залежності від конкретної реалізації системи. Загалом, дерево функцій допомагає краще розуміти функціональні вимоги до модуля колаборативної фільтрації та забезпечити його ефективну та точну роботу (рис. 2.7).

Розпишемо більш детально підфункції для функції «1. Збір даних»:

1.1. Збір оцінок користувачів на об'єкти:

– Перевірка доступності даних для збору

– Визначення формату та типу даних

– Визначення методу збору даних (через веб-інтерфейс або API)

– Збір оцінок з додатковою інформацією, такою як ідентифікатор користувача, ідентифікатор об'єкту, дата та час оцінки

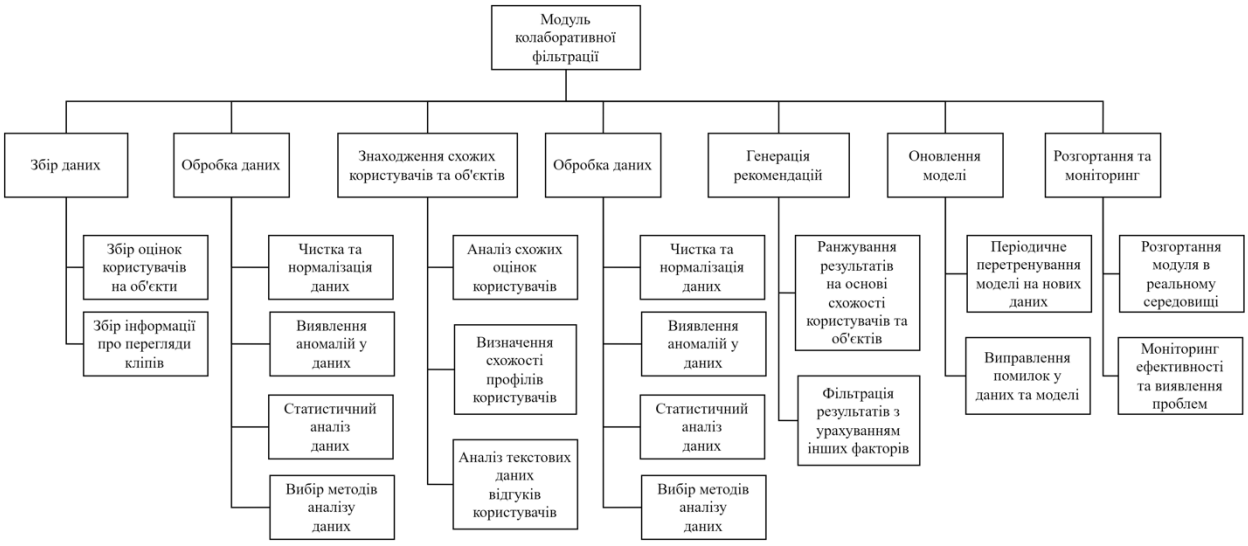

Рисунок 2.7 – Дерево функцій модуля колаборативної фільтрації

1.2. Збір текстових інформації про перегляди кліпів:

– Перевірка доступності даних для збору

– Визначення методу збору даних (наприклад, через веб-інтерфейс або API)

– Збір інформації про перегляди кліпів з додатковою інформацією, такою як ідентифікатор користувача, ідентифікатор об'єкту, дата та час перегляду.

У процесі збору даних можуть виникати додаткові завдання, такі як виявлення дублікатів даних, фільтрація непридатних оцінок. Крім того, необхідно забезпечити конфіденційність даних та виконувати вимоги законодавства щодо захисту персональних даних.

Розпишемо детально функцію «2. Обробка даних»:

2.1. Чистка та нормалізація даних:

– Перевірка даних на наявність помилок та відсутність необхідних даних

– Видалення дублікатів даних

– Конвертація даних в необхідний формат

– Заміна відсутніх значень на значення за замовчуванням або іншими значеннями

– Нормалізація даних для забезпечення однакового масштабу

2.2. Виявлення аномалій у даних:

– Визначення стандартів та критеріїв для визначення аномалій

– Використання статистичних методів для виявлення аномалій, таких як знаходження відхилень від середнього значення, знаходження екстремальних значень тощо

– Видалення аномальних даних або використання спеціальних алгоритмів для їх врахування

2.3. Статистичний аналіз даних:

– Визначення основних статистичних показників, таких як середнє значення, медіана, дисперсія тощо

– Визначення кореляцій між різними даними та виявлення залежностей між ними

– Використання графічних методів для візуалізації даних та їх аналізу

2.4. Вибір методів аналізу даних:

– Визначення мети та завдань аналізу даних

– Вибір методів аналізу даних відповідно до поставлених завдань та доступних даних

– Перевірка ефективності та точності вибраних методів аналізу даних

– Вибір кращих методів аналізу даних для використання в модулі колаборативної фільтрації.

У процесі обробки даних також необхідно враховувати вимоги до конфіденційності даних та виконувати вимоги законодавства. Зокрема, для зберігання та обробки персональних даних можуть бути встановлені обмеження та правила, які необхідно виконувати.

Крім того, важливим етапом обробки даних є визначення якості даних та виявлення помилок у даних. Це може включати перевірку правильності та повноти даних, а також виявлення та виправлення помилок у даних.

Одним з ключових завдань обробки даних є виявлення кореляцій між різними даними та визначення залежностей між ними. Наприклад, у випадку колаборативної фільтрації, важливим є виявлення схожості між користувачами та об'єктами на основі їх оцінок.

Вибір методів аналізу даних є важливим етапом обробки даних, оскільки він визначає точність та ефективність моделі колаборативної фільтрації. Для вибору оптимальних методів аналізу даних можуть використовуватися різні критерії, такі як точність, швидкість роботи, складність алгоритмів тощо.

Загалом, обробка даних є важливим етапом розробки модуля колаборативної фільтрації, оскільки вона забезпечує якість та точність вихідних даних та визначає ефективність моделі.

Розпишемо більш детально підфункції для функції «3. Знаходження схожих користувачів та об'єктів»:

3.1. Аналіз схожих оцінок користувачів:

– Визначення метрики для вимірювання схожості оцінок користувачів, наприклад, косинусна схожість, евклідова відстань тощо

– Порівняння оцінок користувачів з метою знаходження користувачів, які мають схожі оцінки на об'єкти

– Визначення порогу для схожості оцінок користувачів, щоб відфільтрувати менш схожих користувачів

3.2. Визначення схожості профілів користувачів:

– Визначення метрики для вимірювання схожості профілів користувачів, наприклад, косинусна схожість, евклідова відстань тощо

– Аналіз профілів користувачів з метою знаходження користувачів, які мають схожі інтереси та поведінку

– Визначення порогу для схожості профілів користувачів, щоб відфільтрувати менш схожих користувачів

3.3. Аналіз інформації про перегляди кліпів:

– Визначення методів обробки інформації про перегляди кліпів;

– Використання алгоритмів інформації про перегляди кліпів;

– Визначення метрики для вимірювання інформації про перегляди кліпів;

– Порівняння інформації про перегляди кліпів з метою знаходження користувачів, які мають схожі перегляди;

– Визначення порогу для схожості інформації про перегляди кліпів користувачами, щоб відфільтрувати менш схожих користувачів.

За допомогою аналізу схожих оцінок користувачів, схожості профілів користувачів та аналізу інформації про перегляди кліпів можна знайти схожих користувачів та об'єкти. Це дає можливість використовувати знання про схожих користувачів для рекомендацій об'єктів.

Знайдені схожі користувачі можуть бути використані для створення груп користувачів, які мають схожі інтереси та поведінку. Наприклад, у випадку колаборативної фільтрації, групи користувачів можуть бути використані для створення рекомендацій для користувачів на основі інтересів групи.

Знайдені схожі об'єкти можуть бути використані для створення груп об'єктів, які мають схожі характеристики. Наприклад, у випадку колаборативної фільтрації, групи об'єктів можуть бути використані для створення рекомендацій для користувачів на основі схожості з об'єктами з групи.

Загалом, знаходження схожих користувачів та об'єктів є важливим етапом розробки модуля колаборативної фільтрації, оскільки воно допомагає знайти схожість між користувачами та об'єктами, що дозволяє покращити якість та точність рекомендацій.

Крім того, знаходження схожих користувачів та об'єктів може бути складним завданням через велику кількість користувачів та об'єктів, а також через різноманітність оцінок та оцінки переглядів. Тому важливо використовувати ефективні алгоритми та методи обробки даних, щоб знайти схожість користувачів та об'єктів.

Крім знаходження схожих користувачів та об'єктів, можна використовувати інші методи для покращення рекомендацій. Наприклад, можна використовувати методи контентної фільтрації, які використовують характеристики об'єктів для рекомендацій. Також можна використовувати гібридні методи, які комбінують колаборативну та контентну фільтрації.

Загалом, знаходження схожих користувачів та об'єктів є важливим етапом розробки модуля колаборативної фільтрації, оскільки воно дозволяє знайти схожість між користувачами та об'єктами та покращити якість та точність рекомендацій. Важливо використовувати ефективні алгоритми та методи обробки даних, щоб знайти схожість користувачів та об'єктів та забезпечити високу якість рекомендацій.

Розпишемо більш детально підфункції для функції «4. Генерація рекомендацій»:

4.1. Ранжування результатів на основі схожості користувачів та об'єктів:

– Визначення метрики для вимірювання схожості користувачів та об'єктів, наприклад, косинусна схожість, евклідова відстань тощо

– Визначення ваги схожості користувачів та об'єктів відповідно до їх важливості для рекомендацій

– Ранжування результатів за спаданням схожості користувачів та об'єктів

4.2. Фільтрація результатів з урахуванням інших факторів:

– Використання додаткових факторів для фільтрації результатів, наприклад, рейтингу об'єктів, наявності знижок, розташування тощо

– Визначення ваги додаткових факторів відповідно до їх важливості для рекомендацій

– Використання алгоритмів ранжування, наприклад, PageRank, для фільтрації результатів

Після знаходження схожих користувачів та об'єктів та врахування додаткових факторів можна згенерувати список рекомендацій для користувача. Ранжування результатів на основі схожості користувачів та об'єктів дозволяє забезпечити більш персоналізовані рекомендації, оскільки вони базуються на відповідності між користувачем та об'єктом. Фільтрація результатів з урахуванням інших факторів дозволяє покращити якість рекомендацій, оскільки враховується додаткова інформація про об'єкти.

Важливо враховувати, що генерація рекомендацій є ітеративним процесом, який потребує постійного оновлення та вдосконалення моделі. Для покращення якості рекомендацій можна використовувати методи машинного навчання, які дозволяють автоматично вивчати залежності між користувачами та об'єктами на основі історії їх взаємодії.

Також важливо враховувати різноманітність користувачів та об'єктів, що може призводити до проблеми холодного старту, коли немає достатньої кількості даних для рекомендацій новим користувачам або об'єктам. Для розв'язання цієї проблеми можна використовувати додаткові джерела інформації про користувачів та об'єкти, такі як соціальні мережі або інтернет-форуми.

Загалом, генерація рекомендацій є важливим етапом розробки модуля колаборативної фільтрації, оскільки вона дозволяє забезпечити персоналізовані та якісні рекомендації для користувачів. Важливо використовувати ефективні

алгоритми та методи ранжування результатів, а також враховувати додаткові фактори для фільтрації результатів та покращення якості рекомендацій. Також важливо враховувати різноманітність користувачів та об'єктів, щоб уникнути проблеми холодного старту.

Розпишемо більш детально підфункції для функції «5. Оновлення моделі»:

5.1. Періодичне перетренування моделі на нових даних:

– Збір та підготовка нових даних для тренування моделі

– Використання нових даних для перетренування моделі з метою покращення точності та якості рекомендацій

– Визначення частоти перетренування моделі в залежності від обсягу та складності даних

5.2. Виправлення помилок у даних та моделі:

– Виявлення та аналіз помилок у даних та моделі

– Корекція помилок у даних та моделі з метою покращення точності та якості рекомендацій

– Визначення частоти виправлення помилок в залежності від обсягу та складності даних

Розпишемо більш детально підфункції для функції «6. Розгортання та моніторинг»:

6.1. Розгортання модуля в реальному середовищі:

– Підготовка середовища для розгортання модуля, включаючи налаштування серверів та баз даних

– Розгортання модуля в реальному середовищі та підключення до системи рекомендацій

– Перевірка роботи модуля та відладка помилок, які можуть виникнути при роботі в реальному середовищі

6.2. Моніторинг ефективності та виявлення проблем:

– Моніторинг ефективності модуля та виявлення проблем, які можуть виникнути при роботі в реальному середовищі

– Аналіз та виявлення причин проблем з метою їх виправлення та покращення роботи модуля

– Визначення частоти моніторингу та виявлення проблем в залежності від обсягу та складності даних

Оновлення моделі та розгортання модуля в реальному середовищі є важливими етапами розробки модуля колаборативної фільтрації, оскільки вони дозволяють забезпечити актуальні та якісні рекомендації для користувачів. Періодичне перетренування моделі дозволяє враховувати зміни у поведінці користувачів та об'єктів, а також враховувати нові дані, що впливають на рекомендації. Виправлення помилок у даних та моделі дозволяє забезпечити більш точні та якісні рекомендації та уникнути помилкових рекомендацій.

Розгортання модуля в реальному середовищі дозволяє перевірити, як модуль працює в реальних умовах, та вирішити проблеми, які можуть виникнути при роботі в реальному середовищі. Моніторинг ефективності та виявлення проблем дозволяє вчасно виявляти та вирішувати проблеми з роботою модуля, що дозволяє забезпечити стабільну та надійну роботу модуля.

Загалом, оновлення моделі та розгортання модуля в реальному середовищі є важливими етапами в розробці модуля колаборативної фільтрації. Вони дозволяють забезпечити актуальні та якісні рекомендації для користувачів, а також підвищити ефективність та надійність роботи модуля.

2.2.2. Діаграми роботи колаборативної системи рекомендацій у форматі IDEF0

Діаграма роботи колаборативної системи може бути побудована у форматі IDEF0, що дозволяє структурувати та описати функції системи та їх взаємозв'язки.

Загальний опис діаграми роботи колаборативної системи рекомендацій у форматі IDEF0 для описаного функціоналу (рис. 2.8):

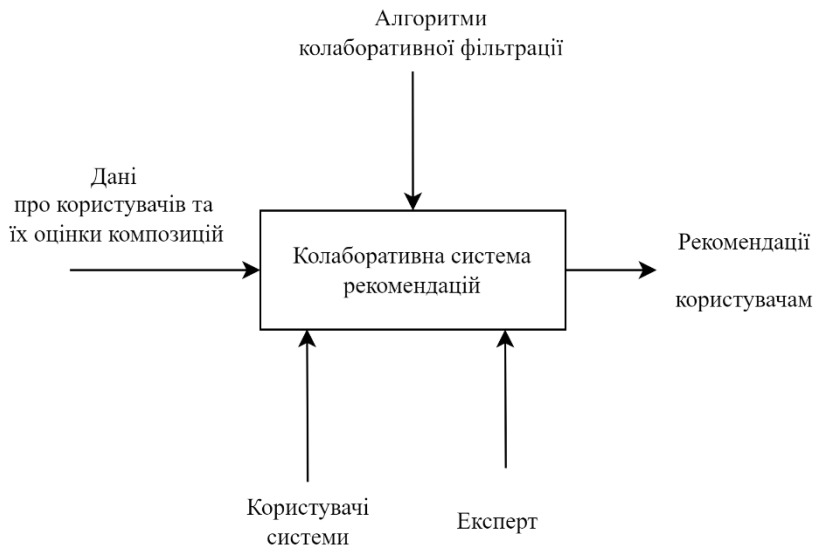

Рисунок 2.8 – Діаграма роботи колаборативної системи рекомендацій у форматі IDEF0

Декомпозиція процесів колаборативної системи рекомендацій у форматі IDEF0 на рис. 2.9.

A0. Колаборативна система рекомендацій;

A1. Збір оцінок користувачів:

A1.1 Збір оцінок користувачів;

A1.2 Збір додаткових даних про користувачів та об'єкти;

A1.3 Чистка, нормалізація та статистичний аналіз даних;

A2. Збір додаткових даних про користувачів:

A2.1 Аналіз схожих оцінок користувачів;

A2.2 Визначення схожості профілів користувачів;

A2.3 Аналіз активації композицій користувачам;

A3. Чистка, нормалізація та статистичний аналіз даних:

A3.1 Ранжування результатів на основі схожості користувачів та об'єктів;

A3.2 Фільтрація результатів з урахуванням інших факторів;

A4. Нейронна мережа для доповнення оцінок:

A4.1 Періодичне тренування моделі на нових даних;

A4.2 Виправлення помилок у даних та моделі.

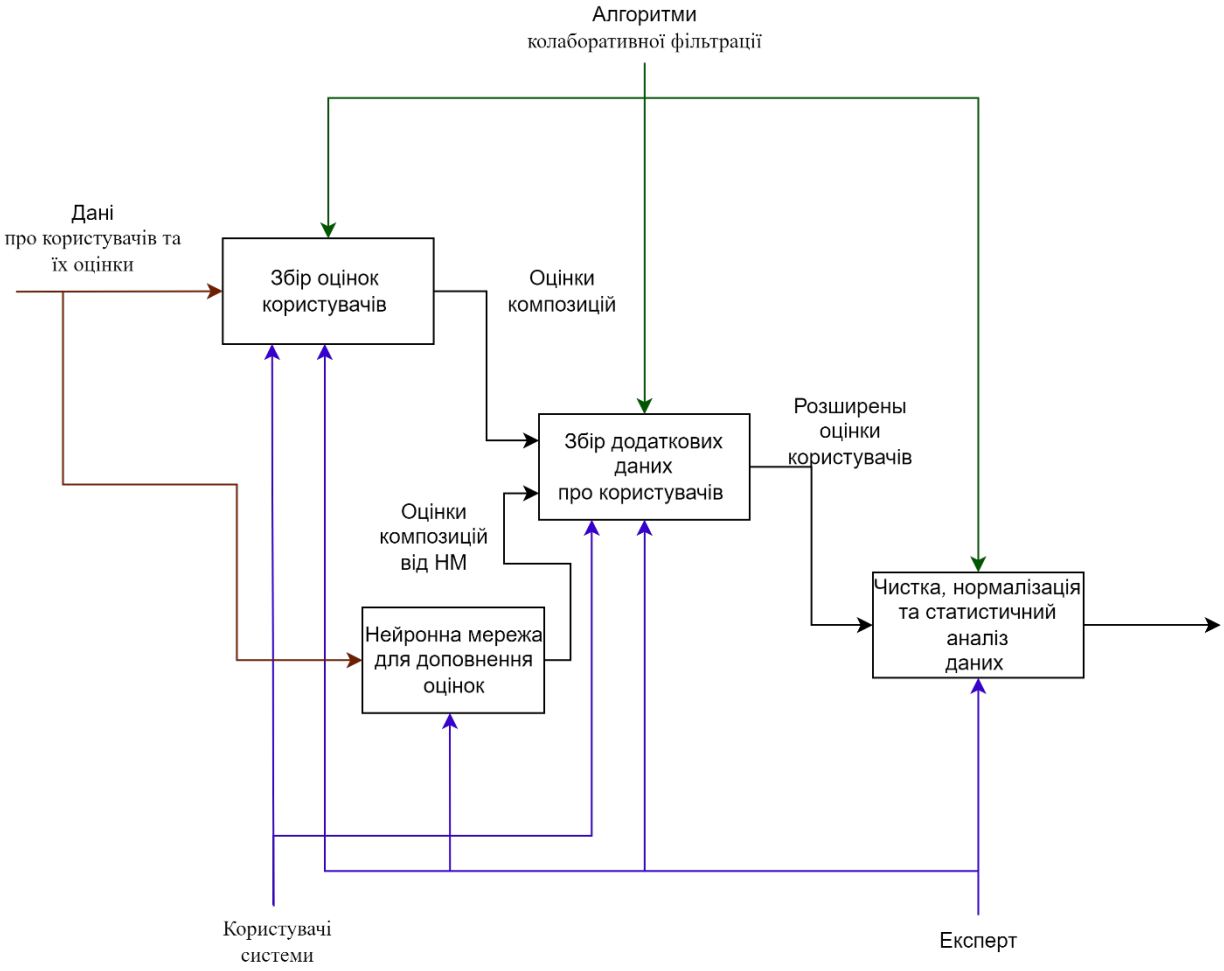

Рисунок 2.9 – Діаграма декомпозиції процесів колаборативної системи рекомендацій у форматі IDEF0

На діаграмі можна виділити основні функції системи та їх взаємозв'язки, а також визначити додаткові елементи, які необхідні для роботи системи, такі як база даних, сервери, алгоритми аналізу даних.

Для діаграми роботи колаборативної системи у форматі IDEF0 можна виокремити вхідні та вихідні потоки. Вхідні потоки - це дані та інформація, які потрібні системі для її роботи, тобто вони є вхідними параметрами функцій системи. Вихідні потоки - це результати роботи функцій системи, які передаються далі для подальшої обробки чи використання.

Отже, на діаграмі роботи колаборативної системи можна виділити наступні вхідні та вихідні потоки:

Вхідні потоки:

– Дані про користувачів та їх оцінки на об'єкти;

– Додаткові дані про користувачів та об'єкти.

Вихідні потоки:

– Рекомендації користувачам щодо об'єктів, які їм можуть сподобатись

– Оновлена модель рекомендаційної системи після перетренування на нових даних

– Повідомлення про проблеми та помилки в роботі системи, які потребують виправлення

– Додаткові дані для моніторингу ефективності системи та виявлення проблем.

Вхідні та вихідні потоки можуть бути подальшими вхідними параметрами чи результатами роботи інших функцій системи. Наприклад, результати генерації рекомендацій можуть бути використані для відображення на веб-сторінках або мобільних додатках, або ж вони можуть стати вхідними параметрами для інших функцій системи, наприклад, для виправлення помилок в даних та моделі.

Для подальшої деталізації та декомпозиції функції А1. Збір та обробка даних можна використати діаграму рівня А1 (рис. 2.9):

А1. Збір та обробка даних

А1.1 Збір оцінок користувачів

А1.1.1 Збір оцінок користувачів на об'єкти

А1.1.2 Збір інформації про перегляди кліпів

А1.2 Збір додаткових даних про користувачів та об'єкти

А1.2.1 Збір інформації про користувачів

А1.2.2 Збір інформації про об'єкти

А1.3 Чистка, нормалізація та статистичний аналіз даних

А1.3.1 Чистка та попередня обробка даних

А1.3.2 Нормалізація даних

А1.3.3 Статистичний аналіз даних

А1.4 Вибір методів аналізу даних

A1.4.1 Вибір методів аналізу схожості користувачів та об'єктів

A1.4.2 Вибір методів рекомендації користувачам

Діаграма рівня A1 дозволяє розкрити деталізацію функції A1 на окремі підфункції та їх взаємозв'язки. Для кожної підфункції можна провести подальшу декомпозицію та деталізацію.

Для функції «A1.1. Збір оцінок та інформаціїї про перегляди кліпів» можна провести наступну деталізацію:

A1.1 Збір оцінок та інформаціїї про перегляди кліпів

A1.1.1 Збір оцінок користувачів на об'єкти

A1.1.1.1 Встановлення зв'язку з базою даних оцінок

A1.1.1.2 Отримання списку користувачів та композицій для збору оцінок

A1.1.1.3 Збір та Збереження оцінок користувачів

A1.1.2 Збір інформації про перегляди кліпів

A1.1.2.1 Встановлення зв'язку з базою даних

A1.1.2.2 Отримання списку користувачів та об'єктів для збору інформації про перегляди кліпів

A1.1.2.3 Збір інформації про перегляди кліпів

A1.1.2.4 Збереження інформації про перегляди кліпів у базі даних

Для функції A1.2. Збір додаткових даних про користувачів та об'єкти можна провести подібну деталізацію та декомпозицію, виокремивши окремі підфункції, наприклад, вибір методів збору додаткової інформації, розробку запитів до баз даних для отримання необхідної інформації, збереження даних у базі даних тощо.

Для функції A1.3. Чистка, нормалізація та статистичний аналіз даних можна провести наступну деталізацію та декомпозицію:

A1.3 Чистка, нормалізація та статистичний аналіз даних

A1.3.1 Чистка та попередня обробка даних

A1.3.1.1 Видалення дублікатів

A1.3.1.2 Видалення відсутніх даних

A1.3.1.3 Видалення викидів та аномалій

A1.3.2 Нормалізація даних

A1.3.2.1 Нормалізація оцінок користувачів

A1.3.2.2 Нормалізація інформації про об'єкти та користувачів

A1.3.3 Статистичний аналіз даних

A1.3.3.1 Розрахунок статистичних показників (середнє значення, медіана, дисперсія тощо)

A1.3.3.2 Візуалізація даних за допомогою графіків, діаграм та інших методів

Для функції A1.4. Вибір методів аналізу даних можна провести подібну деталізацію та декомпозицію, виокремивши окремі підфункції, наприклад, визначення схожості між користувачами та об'єктами, ранжування результатів та вибір методів фільтрації результатів. Детальна декомпозиція та деталізація кожної функції допомагає зрозуміти процеси, що відбуваються у системі та забезпечує більш ефективну розробку та управління системою рекомендацій.

Для подальшої деталізації та декомпозиції функції «A2. Знаходження схожих користувачів та об'єктів» можна використати діаграму рівня A1 (рис. 2.10):

A2. Знаходження схожих користувачів та об'єктів

A2.1 Аналіз схожих оцінок користувачів

A2.1.1 Вибір алгоритмів для аналізу схожості оцінок

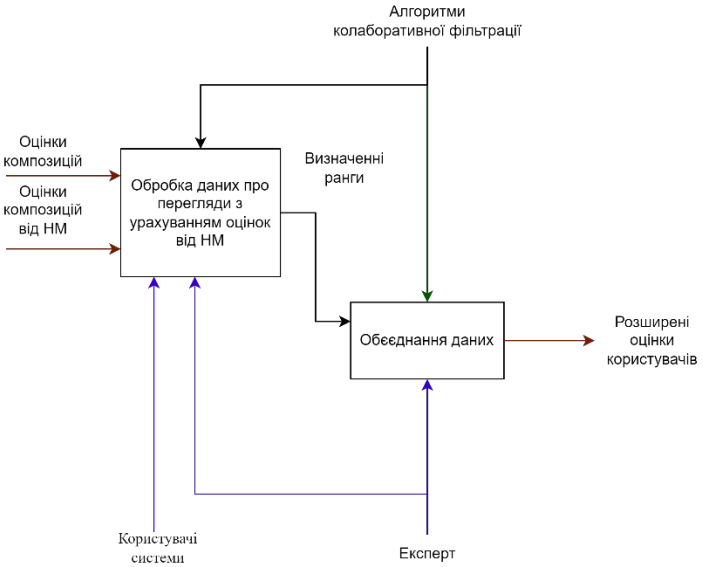

Рисунок 2.10 – Діаграма рівня A1 для «Збір додаткових данихпро користувачів»

А2.1.2 Побудова матриці схожості між користувачами на основі оцінок

А2.1.3 Вибір порогового значення для визначення схожості між користувачами

А2.1.4 Відбір схожих користувачів для подальшого аналізу

А2.2 Визначення схожості профілів користувачів

А2.2.1 Вибір алгоритмів для визначення схожості профілів користувачів

А2.2.2 Побудова матриці схожості між користувачами на основі профілів

А2.2.3 Вибір порогового значення для визначення схожості між користувачами

А2.2.4 Відбір схожих користувачів для подальшого аналізу

А2.3 Аналіз інформації про перегляди кліпів

А2.3.1 Вибір алгоритмів для аналізу інформації про перегляди кліпів

А2.3.2 Побудова моделі векторного простору для інформації про перегляди кліпів

А2.3.3 Вибір порогового значення для визначення схожості інформації про перегляди кліпів

А2.3.4 Відбір схожих користувачів у відповідності до інформації про перегляди кліпів

Кожна з цих підфункцій може мати свої власні підфункції, які необхідні для їх виконання, наприклад, для функції А2.1.1 Вибір алгоритмів для аналізу схожості оцінок можна виокремити підфункції, які включають аналіз даних, порівняння різних алгоритмів, вибір оптимального алгоритму для певної задачі тощо.

Також для функції А2.2.2 Побудова матриці схожості між користувачами на основі профілів можна виокремити наступні підфункції:

А2.2.2.1 Виділення ключових характеристик профілів користувачів

А2.2.2.2 Побудова векторного простору профілів користувачів

А2.2.2.3 Визначення схожості між векторами профілів користувачів

А2.2.2.4 Побудова матриці схожості між користувачами на основі векторного простору профілів

Для функції А2.3.2 Побудова моделі векторного простору для інформації про перегляди кліпів користувачами можна виокремити такі підфункції:

А2.3.2.1 Токенізація інформації про перегляди кліпів

А2.3.2.2 Побудова індексу термів для інформації про перегляди кліпів

А2.3.2.3 Застосування методів векторної моделі для інформації про перегляди кліпів

Такі підфункції допомагають зрозуміти процеси, які відбуваються у системі рекомендацій та забезпечують більш ефективну розробку та управління системою.

Для подальшої деталізації та декомпозиції функції «А3. Генерація рекомендацій» можна використати діаграму рівня А1 (рис. 2.11):

А3. Чистка, нормалізація та статистичний аналіз даних

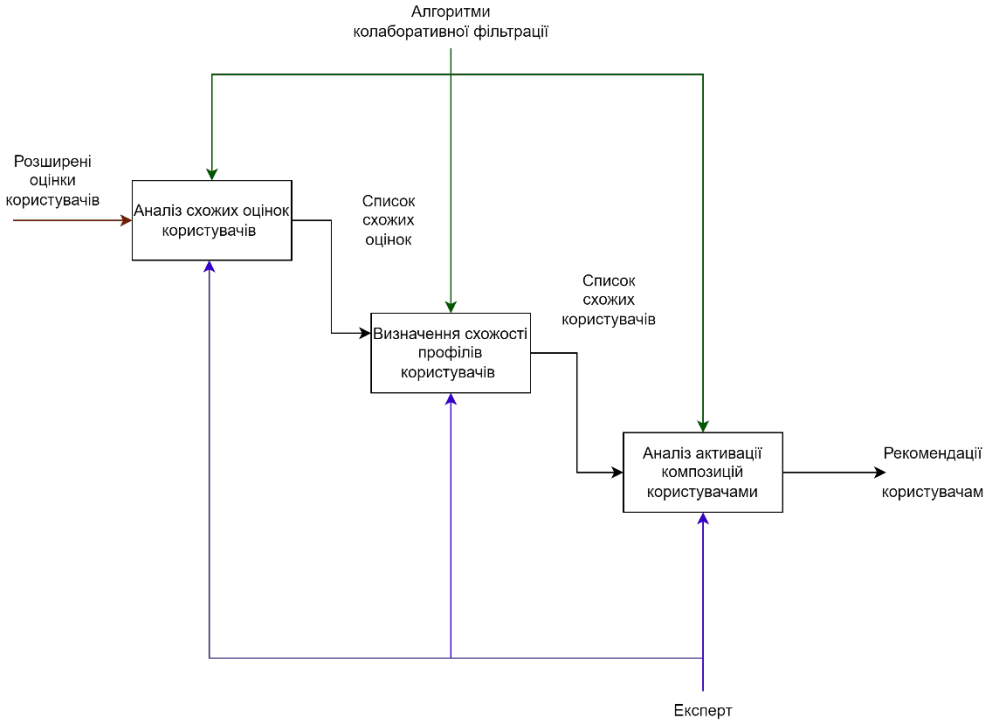

Рисунок 2.11 – Діаграма рівня А1 для «А3. Генерація рекомендацій»

А3.1 Ранжування результатів на основі схожості користувачів та об'єктів

А3.1.1 Вибір алгоритмів для ранжування результатів

А3.1.2 Побудова рейтингу рекомендованих об'єктів на основі схожості користувачів та об'єктів

A3.1.3 Вибір порогового значення для визначення кількості рекомендованих об'єктів

A3.1.4 Виведення рекомендованих об'єктів для користувачів

A3.2 Фільтрація результатів з урахуванням інших факторів (наприклад, рейтингу об'єктів)

A3.2.1 Визначення інших факторів, які можуть впливати на рекомендації

A3.2.2 Побудова рейтингу об'єктів на основі інших факторів

A3.2.3 Врахування рейтингу об'єктів у фільтрації результатів рекомендацій

A3.2.4 Виведення рекомендованих об'єктів з урахуванням рейтингу для користувачів

Кожна з цих підфункцій може мати свої власні підфункції, які необхідні для їх виконання. Наприклад, для функції A3.1.1 Вибір алгоритмів для ранжування результатів можна виокремити підфункції, які включають аналіз даних, порівняння різних алгоритмів, вибір оптимального алгоритму для певної задачі тощо.

Також для функції A3.2.2 Побудова рейтингу об'єктів на основі інших факторів можна виокремити наступні підфункції:

A3.2.2.1 Визначення інших факторів, які можуть впливати на рейтинг об'єктів

A3.2.2.2 Збір даних про інші фактори та оцінок користувачів

A3.2.2.3 Відбір та відфільтрування необхідних даних

A3.2.2.4 Обчислення вагових коефіцієнтів для кожного фактора

A3.2.2.5 Побудова рейтингу об'єктів на основі вагових коефіцієнтів та оцінок користувачів

Декомпозиція функцій A3.1 та A3.2 допомагає зрозуміти, які процеси відбуваються при генерації рекомендацій, як вони пов'язані між собою та які додаткові фактори можуть впливати на результати.

Для подальшої деталізації та декомпозиції функції «A4. Оновлення моделі» можна використати діаграму рівня A1 (рис.. 2.12):

A4. Оновлення моделі

A4.1 Періодичне тренування моделі на нових даних

A4.1.1 Збір нових даних та оновлення бази даних

A4.1.2 Визначення параметрів моделі, які потребують перетренування

A4.1.3 Вибір алгоритму для перетренування моделі

A4.1.4 Перетренування моделі на нових даних

A4.1.5 Тестування та оцінка ефективності оновленої моделі

A4.2 Виправлення помилок у даних та моделі

A4.2.1 Виявлення помилок та неточностей у даних та моделі

A4.2.2 Аналіз та виправлення помилок у даних та моделі

A4.2.3 Оновлення бази даних та моделі з урахуванням виправлень

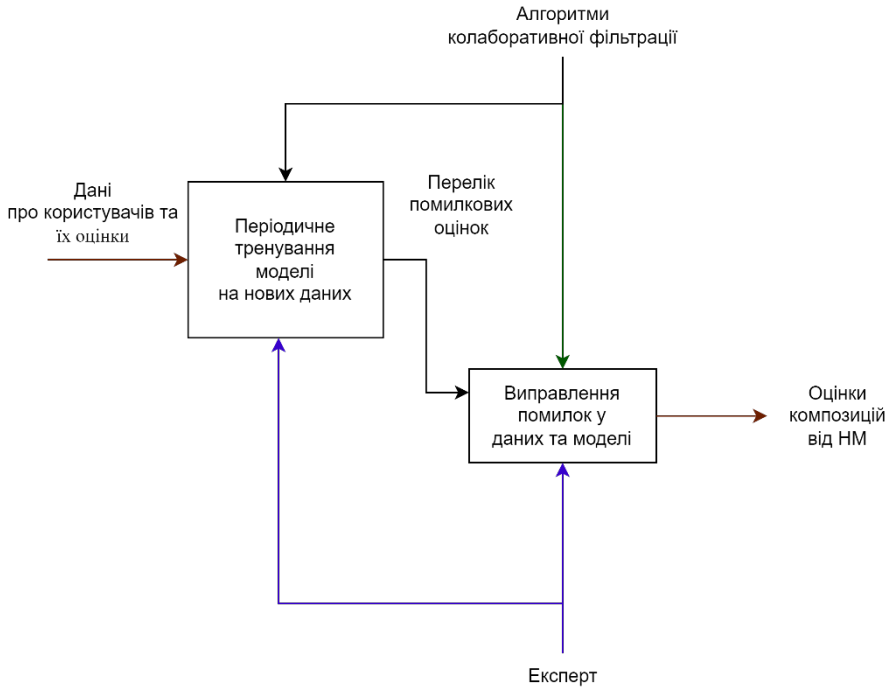

Рисунок 2.12 – Діаграма рівня A1 для «Нейронна мережа для доповнення оцінок»

Декомпозиція функцій A4.1 та A4.2 допомагає зрозуміти, які процеси потрібно виконувати для забезпечення оновлення та підтримки ефективності моделі. Вона також відображає важливість постійного аналізу та оновлення даних та моделі для забезпечення точності та ефективності системи рекомендацій.

2.2.3. Діаграма використання системи рекомендацій

Діаграма використання системи рекомендацій складається з двох головних блоків - користувачів та системи. Користувачі взаємодіють з системою, щоб отримати персоналізовані рекомендації на основі їхнього профілю та взаємодії з об'єктами, а система здійснює відповідні обчислення та видає рекомендації.

На діаграмі показані наступні етапи взаємодії:

1. Користувачі надсилають запити до системи з метою отримання рекомендацій.
2. Система приймає запит та виконує обробку даних згідно функціоналу, описаному на діаграмах IDEF0.
3. Система генерує список рекомендацій на основі аналізу даних користувача та об'єктів.
4. Система повертає рекомендації користувачам.
5. Користувачі отримують рекомендації та можуть взаємодіяти з системою, надсилаючи додаткові запити.
6. Система зберігає дані про взаємодію користувачів та об'єктів для подальшого аналізу та оновлення моделі.

В системі рекомендацій можуть бути різні типи користувачів з різними ролями та доступами. Основними типами користувачів можуть бути:

1. Звичайний користувач - це основний тип користувача, який отримує персоналізовані рекомендації та може взаємодіяти з системою, шукаючи об'єкти та надсилаючи запити на рекомендації.
2. Адміністратор - користувач з підвищеними правами доступу, який має змогу управляти системою та її функціоналом. Адміністратор може додавати та видаляти об'єкти, керувати настройками та виконувати інші функції.
3. Експерт - користувач з підвищеними знаннями в певній галузі, який може додавати та редагувати об'єкти, проводити аналіз та відповідати на запити користувачів щодо певних об'єктів.

4. Розробник - користувач, який займається розробкою та підтримкою системи. Розробник має доступ до внутрішніх налаштувань та коду системи та може проводити розробку нових функцій.

Різні типи користувачів мають різні доступи та функції у системі рекомендацій. Наприклад, звичайний користувач може тільки отримувати рекомендації та надсилати запити, тоді як адміністратор може управляти всіма налаштуваннями та функціями системи. Це дозволяє краще керувати та підтримувати систему, а також забезпечує більшу безпеку та захист від неправомірного доступу до системи.

Для кожного типу користувача можна розглянути функції, які він може виконувати у системі рекомендацій:

1. Звичайний користувач:

– Перегляд персоналізованих рекомендацій

– Пошук об'єктів за ключовими словами або категоріями

– Додавання об'єктів у список обраних або улюблених

2. Адміністратор:

– Управління користувачами та їх доступами

– Додавання та видалення об'єктів

– Налаштування параметрів рекомендаційної системи

– Моніторинг ефективності системи та виявлення проблем

– Виконання інших функцій, пов'язаних з керуванням та підтримкою системи

3. Експерт:

– Додавання та редагування об'єктів

– Аналіз та відповідь на запити користувачів щодо певних об'єктів

– Виконання інших функцій, пов'язаних з редагуванням та підтримкою об'єктів у системі

4. Розробник:

– Розробка нових функцій та покращень системи

– Підтримка та оновлення існуючого коду та функціоналу

– Розробка та тестування нових алгоритмів та методів рекомендацій

Ці функції можна представити на діаграмі використання, яка відображатиме можливість користувачів здійснювати певні дії в системі (рис. 2.13.

Рисунок 2.13 – Діаграма використання

2.3. Проєктування бази даних для підтримки роботи системи рекомендацій

Для побудови БД розпишемо таблиці, які повинні бути в БД, які по суті є концептуальною моделлю БД для системи рекомендацій:

1. Таблиця "authors" (автори пісень):

– id: унікальний ідентифікатор автора (тип: bigint(20) unsigned);

– name: ім'я автора (тип: varchar(255));

– created_at: дата і час створення запису (тип: timestamp);

– updated_at: дата і час оновлення запису (тип: timestamp).

2. Таблиця "songs" (пісні):

– id: унікальний ідентифікатор пісні (тип: bigint(20) unsigned);

– name: назва пісні (тип: varchar(255));

– url: URL-адреса пісні (тип: varchar(255));

– likes: кількість лайків пісні (тип: int);

– plays: кількість програвань пісні (тип: int);

– created_at: дата і час створення запису (тип: timestamp);

– updated_at: дата і час оновлення запису (тип: timestamp).

3. Таблиця "genres" (жанри пісень):

– id: унікальний ідентифікатор жанру (тип: bigint(20) unsigned);

– name: назва жанру (тип: varchar(255));

– created_at: дата і час створення запису (тип: timestamp);

– updated_at: дата і час оновлення запису (тип: timestamp).

4. Таблиця "users" (користувачі):

– id: унікальний ідентифікатор користувача (тип: bigint(20) unsigned);

– name: ім'я користувача (тип: varchar(255));

– email: електронна пошта користувача (тип: varchar(255));

– created_at: дата і час створення запису (тип: timestamp);

– updated_at: дата і час оновлення запису (тип: timestamp).

Така структура даних дозволяє зберігати інформацію про авторів пісень, самі пісні, жанри, а також про користувачів, які взаємодіють з системою рекомендацій. За допомогою зв'язків між таблицями можна встановлювати зв'язки між піснями, авторами, жанрами та користувачами, які їх прослуховують/дивляться кліпи.

Тепер можна описати зв’язки між таблицями:

– зв'язок між таблицями "authors" та "songs": в таблиці "songs" є поле "author_id", яке посилається на поле "id" таблиці "authors". Це вказує на те, що один автор може мати багато пісень;

– зв'язок між таблицями "songs" та "genres": в таблиці "songs" є поле "genre_id", яке посилається на поле "id" таблиці "genres". Це вказує на те, що одна пісня може належати до одного жанру;

– зв'язок між таблицями "users" та "songs": через додаткову таблицю "likes" з полями "user_id" та "song_id", які посилаються на поля "id" таблиць "users" та

"songs", вказується, що один користувач може лайкати багато пісень, і одна пісня може мати багато лайків від різних користувачів;

– зв'язок між таблицями "users" та "authors": через додаткову таблицю "follows" з полями "user_id" та "author_id", які посилаються на поля "id" таблиць "users" та "authors", вказується, що один користувач може стежити за багатьма авторами, і один автор може мати багато фоловерів (користувачів, які його стежать).

Це дозволяє представити концептуальну модель БД (рис. 2.14).

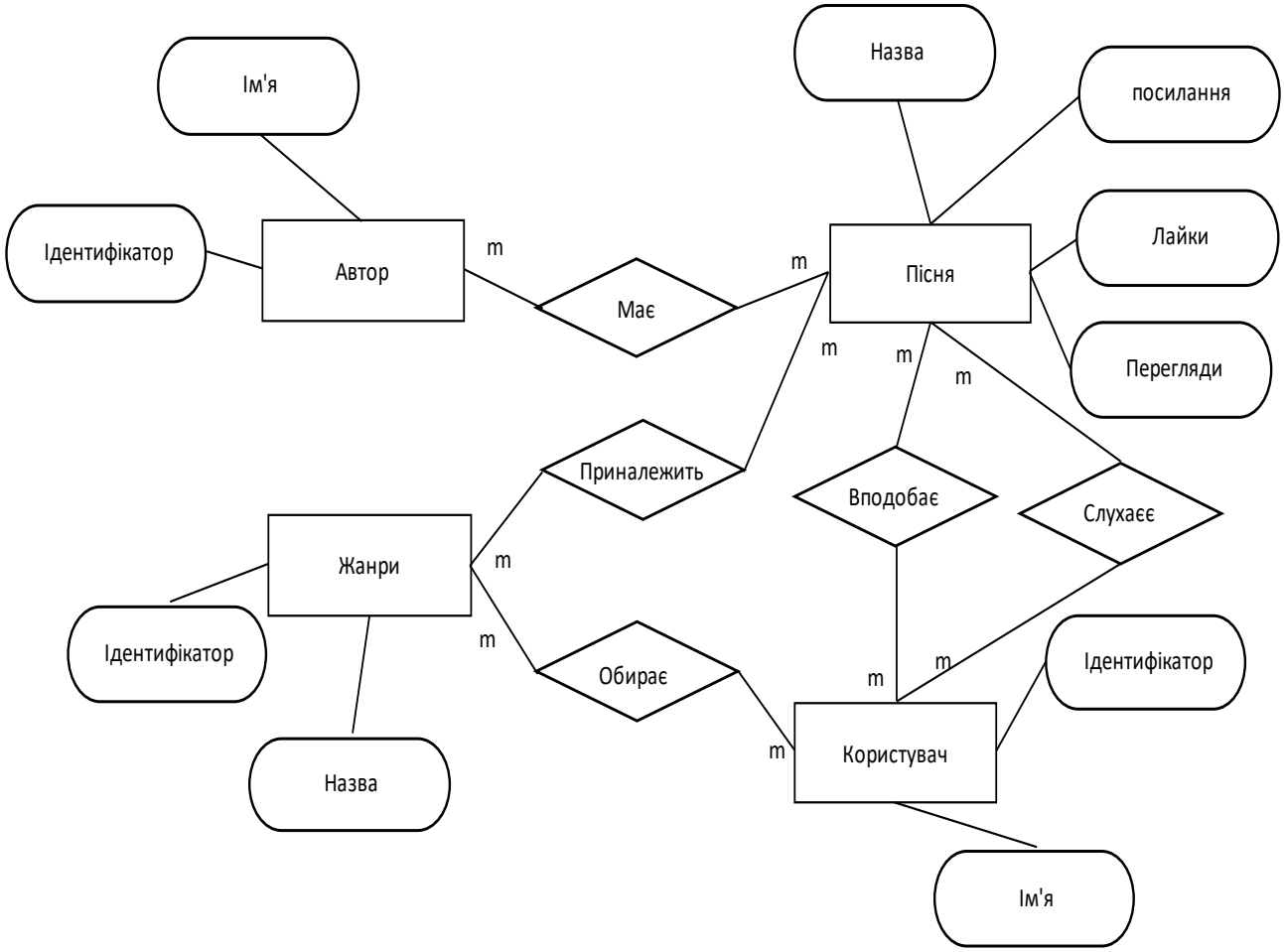

Рисунок 2.14 – Концептуальна модель БД

З урахуванням проведеного аналізу роботи системи рекомендацій та фреймворку Laravel для створення ситсеми було спроєктовано наступну базу даних, яка містить таблиці:

– authors;

– failed_jobs;

– genres;

– migrations;

– password_resets;

– personal_access_tokens;

– songables;

– songs;

– userables;

– users.

В БД передбачено таблиці для збереження даних та службові таблиці.

Таблиці мають наступне призначення.

– Таблиця authors містить інформацію про авторів пісень (табл. 2.1);

Таблиця 2.1

Структура таблиці authors

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PK	NULL	auto_increment
name	varchar(255)	NO		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

– Таблиця songs містить інформацію про пісні, які можуть бути рекомендовані користувачам (табл. 2.2).

Таблиця 2.2

Структура таблиці songs

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PK	NULL	auto_increment
name	varchar(255)	NO		NULL	
url	varchar(255)	NO		NULL	
likes	int	YES		NULL	
plays	Int	YES		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

– Таблиця genres містить список жанрів, які можуть бути присвоєні пісням (табл. 2.3).

Таблиця 2.3

Структура таблиці genres

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PK	NULL	auto_increment
name	varchar(255)	NO		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

– Таблиця songables містить прив’язку пісень до авторів і до жанрів, що визначається полем songable_type (табл. 2.4).

Таблиця 2.4

Структура таблиці songables

Field	Type	Null	Key	Default	Extra
song_id	int(20) unsigned	NO	FK	NULL	
songable_id	int(20) unsigned	NO	FK	NULL	
songable_type	varchar(255)	NO		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

– Таблиця userables містить прив’язку користувачів до жанрів і до пісень, що визначається полем userable_type (табл. 2.5).

Таблиця 2.5

Структура таблиці userables

Field	Type	Null	Key	Default	Extra
user_id	int(20) unsigned	NO	FK	NULL	
userable_id	int(20) unsigned	NO	FK	NULL	
userable_type	varchar(255)	NO		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

Службові таблиці:

– Таблиця users містить інформацію про користувачів, які взаємодіють з системою рекомендацій.

– Таблиця failed_jobs містить інформацію про неуспішні завдання (табл. 2.6);

Таблиця 2.6

Структура таблиці failed_jobs

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PK	NULL	auto_increment
uuid	varchar(255)	NO		NULL	
connection	text	NO		NULL	
queue	text	NO		NULL	
payload	longtext	NO		NULL	
exception	longtext	NO		NULL	
failed_at	timestamp	NO		CURRENT_TIMESTAMP	

– Таблиця migrations містить інформацію про міграцію даних в системі (табл. 2.7);

Таблиця 2.7

Структура таблиці migrations

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto_increment
migration	varchar(255)	NO		NULL	
batch	int(11)	NO		NULL	

– Таблиця password_resets містить інформацію про скидання паролів користувачів (табл. 2.8);

Таблиця 2.8

Структура таблиці password_resets

Field	Type	Null	Key	Default	Extra
email	varchar(255)	NO	MUL	NULL	
token	varchar(255)	NO		NULL	
created_at	timestamp	YES		NULL	

– Таблиця personal_access_tokens містить інформацію про токени доступу, які використовуються для аутентифікації користувачів (табл. 2.9).

Таблиця 2.9

Структура таблиці personal_access_tokens

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
tokenable_type	varchar(255)	NO	MUL	NULL	
tokenable_id	bigint(20) unsigned	NO		NULL	
name	varchar(255)	NO		NULL	
token	varchar(64)	NO	UNI	NULL	
abilities	text	YES		NULL	
last_used_at	timestamp	YES		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

На основі представлених описів таблиць і її моделі побудовано структуру БД (рис. 2.15).

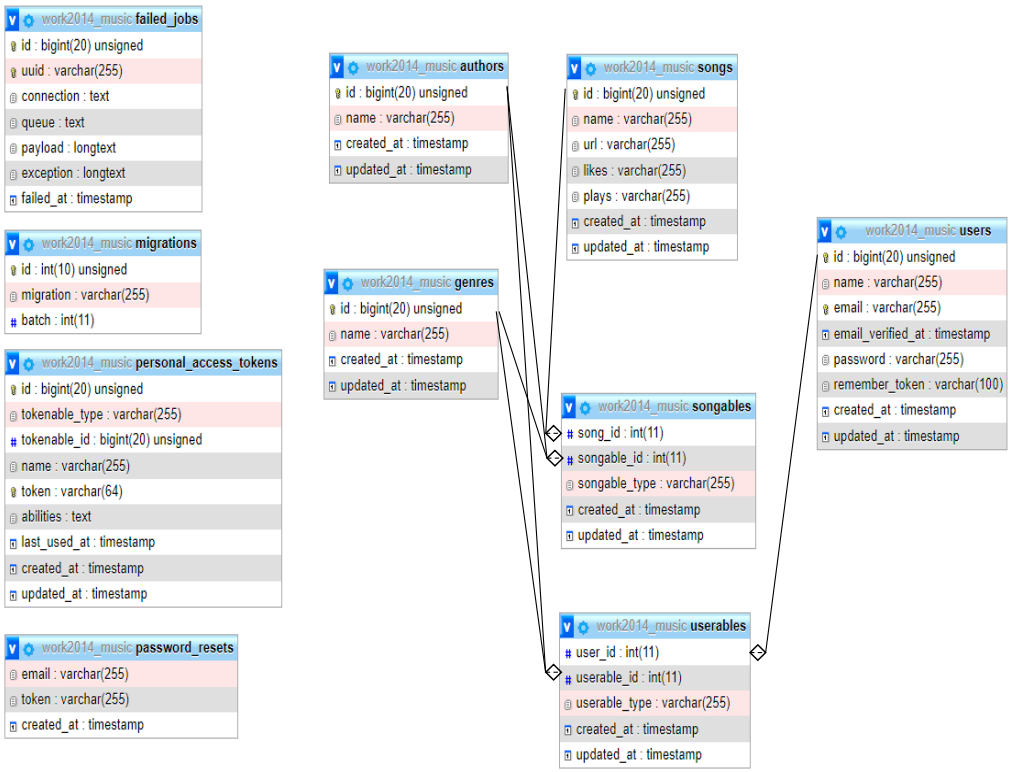

Рисунок 2.15 – Структура БД

Висновки за розділом

У розділі розглянуто основні алгоритми для побудови рекомендаційних систем, такі як Learning to Rank, Sequential, Variational Autoencoders, Reinforcement Learning. Крім того, були описані детально моделі оцінки вхідних змінних - Wide Component та Deep Component.

Були наведені приклади реалізації рекомендаційних алгоритмів, такі як Self-Attentive Sequential Recommendation (SASRec), Fusing Item Similarity Models with Self-Attention (FISSA), Collaborative Memory Networks (CMN), Collaborative Variational Autoencoder (CVAE).

Крім того, було проведено моделювання бізнес-процесів і архітектури модуля колаборативної фільтрації і спроєктовано структуру БД для підтримки роботи системи рекомендацій.

Розділ 3

Опис розробленого модуля фільтрації рекомендаційної музичної системи

3.1. Вибір програмних і технічних компонентів для розробки програмного модуля

Для розгортання модуля треба описати фізичну архітектуру системи та її компонентів, а також їх взаємодію під час роботи. Розгортання системи рекомендацій включає наступні компоненти (рис. 3.1):

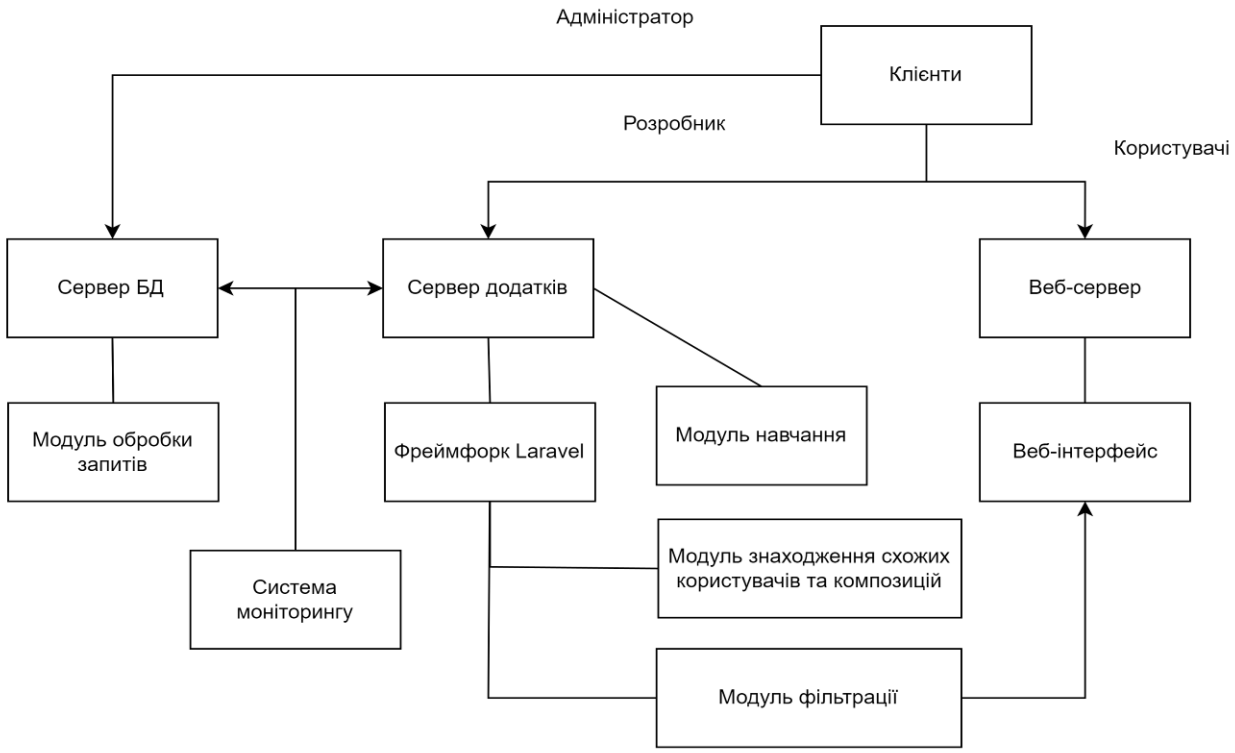

Рисунок 3.1 – Архітектура системи рекомендацій

1. Сервер бази даних, на якому зберігаються всі дані про користувачів та об'єкти, а також дані про їх взаємодію.

2. Сервер додатків, на якому працюють різні модулі системи рекомендацій, такі як знаходження схожих користувачів та композицій, генерація рекомендацій, модуль навчання та інші.

3. Веб-сервер, на якому працює веб-інтерфейс системи рекомендацій, що дозволяє користувачам взаємодіяти з системою та отримувати рекомендації.

4. Клієнти: різні типи клієнтів, які взаємодіють з системою рекомендацій (мобільні додатки, веб-браузери, десктопні додатки та інші).

5. Система моніторингу: програмний модуль, який дозволяє відслідковувати ефективність системи рекомендацій, моніторити її стан та виявляти можливі проблеми.

Ці компоненти контактують між собою через мережу, що забезпечує передачу даних між ними. Також можуть бути встановлені додаткові механізми для забезпечення безпеки даних, наприклад, файрволи, системи аутентифікації та авторизації, системи резервного копіювання даних та інші.

Для розгортання системи було обрано фреймворку Laravel та СУБД mySQL.

Laravel - це популярний фреймворк PHP для розробки веб-додатків. Для розгортання рекомендаційної музичної системи на Laravel можна використовувати наступні особливості:

1. Використання міграцій для створення та оновлення БД. Міграції в Laravel дозволяють зберігати структуру БД та зміни в схемі в окремих файлах міграцій, які можна використовувати для автоматичного створення та оновлення БД при розгортанні системи.

2. Використання seed-ів для заповнення БД початковими даними. Seed-и дозволяють заповнити БД тестовими даними, що дозволяє провести випробування системи на тестових даних.

3. Використання Eloquent ORM для роботи з БД. Eloquent ORM є частиною Laravel та дозволяє просто та зручно взаємодіяти з БД використовуючи моделі та міграції.

4. Використання Blade для створення HTML-шаблонів. Blade є вбудованим шаблонізатором в Laravel, який дозволяє легко та швидко створювати HTML-сторінки за допомогою спеціальних директив та компонентів.

5. Використання Composer для управління залежностями. Composer дозволяє легко встановлювати та оновлювати залежності проекту, такі як бібліотеки та інші залежності.

6. Використання Artisan для створення та виконання міграцій та seed-ів. Artisan є консольним інтерфейсом для роботи з Laravel, який дозволяє створювати та виконувати міграції та seed-и в зручному консольному інтерфейсі.

7. Використання різноманітних бібліотек та пакетів, які дозволяють розширювати можливості Laravel та додавати нові функції до системи.

Laravel має стандартну структуру каталогів, яка досить добре організована та спрощує розробку веб-додатків. Основні каталоги, які містить фреймворк Laravel, описані нижче:

1. app: Каталог app містить основний код додатку. В цьому каталозі розміщуються моделі, контролери, сервіси та інші класи, які використовуються в додатку.

2. bootstrap: Каталог bootstrap містить файли, які використовуються при запуску додатку, такі як файли налаштування, класи для завантаження компонентів фреймворку та інші важливі файли.

3. config: Каталог config містить файли конфігурації для додатку, такі як файли для налаштування бази даних, автентифікації, маршрутизації та інші.

4. database: Каталог database містить файли для налаштування бази даних, міграції, сіди та інші файли, що пов'язані з базою даних.

5. public: Каталог public містить файли, які доступні для загального доступу, такі як статичні файли, зображення та інші ресурси.

6. resources: Каталог resources містить ресурси, які використовуються в додатку, такі як шаблони, переклади, стилі та інші.

7. routes: Каталог routes містить файли маршрутів, які використовуються для маршрутизації запитів у додатку.

8. storage: Каталог storage містить файли, що зберігаються в додатку, такі як логи, кешовані файли та інші.

9. tests: Каталог tests містить тести для додатку.

Крім цих основних каталогів, Laravel має ще декілька каталогів, таких як vendor (для зберігання сторонніх бібліотек), node_modules (для зберігання залежностей JavaScript), public_html (для зберігання статичних ресурсів у випадку використання Apache-серверу), і storage/app/public (для зберігання файлів, які можуть бути доступні для завантаження користувачами).

Структура каталогів Laravel є досить логічною та добре організованою, що дозволяє швидко зорієнтуватися в проекті та швидко знайти потрібні файли. Більшість файлів та каталогів мають зрозумілі назви, що також полегшує розуміння структури проекту. Також Laravel має детально описану документацію, яка містить багато прикладів та порад з розробки додатків на цьому фреймворку.

Для базового коду колаборативної фільтрації використано пакет бібліотек tigoCaval, який використовує алгоритми euclidean distance та slope one.

3.2. Налаштування бази даних системи

Принцип роботи з даною базою даних в my_sql для розробленої системи на php Laravel полягає в зберіганні даних, що використовуються для функціонування системи, таких як інформація про користувачів, авторів, жанри музики, пісні, що пропонуються користувачам, і т.д.

На рис. 3.2 предствлено вигляд PHPMyadmin з базою.

Запити щодо колаборативної фільтрації можуть використовувати дані з таблиці songables та userables, які містять інформацію про зв'язок між користувачами та піснями, що їх вони вже слухали.

Основні запити для колаборативної фільтрації можуть включати наступні:

1. Запит для вибору користувачів, які слухали ту ж саму пісню, що і певний користувач:

SELECT DISTINCT(users.id), users.name FROM users JOIN userables ON users.id = userables.user_id WHERE userables.song_id IN (SELECT song_id FROM userables WHERE user_id = <user_id>) AND users.id != <user_id>;

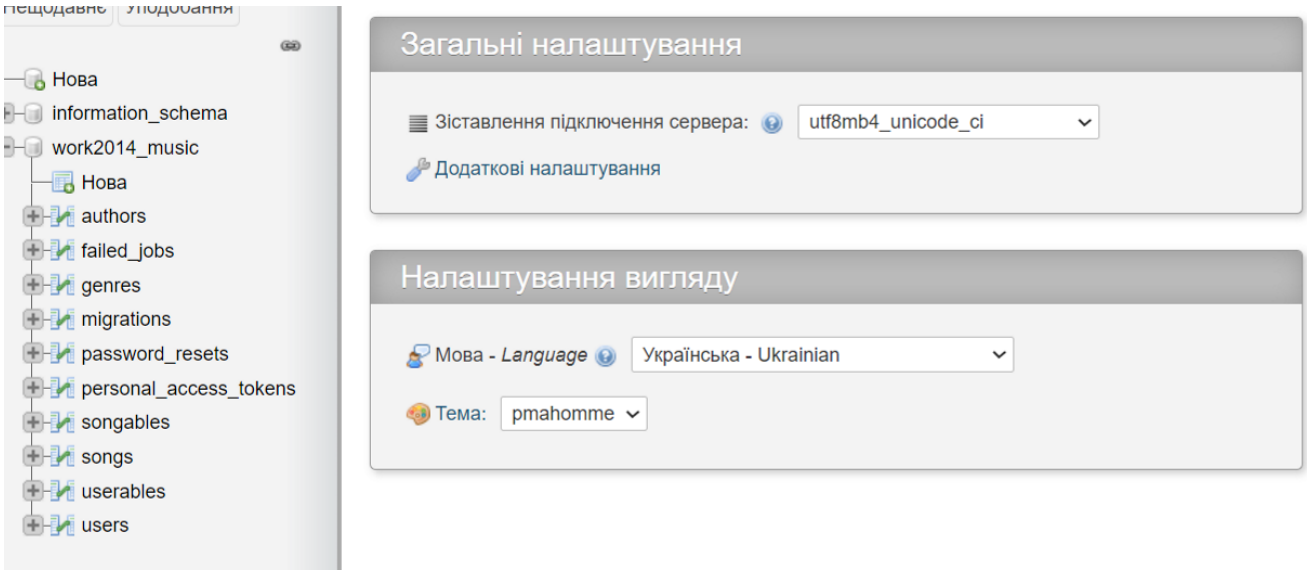

Рисунок 3.2 – Вікно PHPMyadmin

2. Запит для вибору всіх пісень, які були вже прослухані користувачем:

SELECT DISTINCT(song_id) FROM userables WHERE user_id = <user_id>;

3. Запит для вибору користувачів, які слухали ті ж пісні, що й певний користувач, і вибір серед них тих, хто має найбільшу кількість спільних пісень:

SELECT users.id, users.name, COUNT(*) as common_songs FROM users JOIN userables ON users.id = userables.user_id WHERE userables.song_id IN (SELECT song_id FROM userables WHERE user_id = <user_id>) AND users.id != <user_id> GROUP BY users.id ORDER BY common_songs DESC LIMIT <n>;

4. Запит для вибору пісень, які ще не були прослухані певним користувачем, але були прослухані користувачами, які мають найбільшу кількість спільних пісень з ним:

SELECT DISTINCT(songables.song_id) FROM songables JOIN userables ON songables.author_id = userables.author_id OR songables.genre_id = userables.genre_id WHERE userables.user_id IN (SELECT users.id FROM users JOIN userables ON users.id = userables.user_id WHERE userables.song_id IN (SELECT song_id FROM userables WHERE user_id = <user_id>) AND users.id != <user_id> GROUP BY users.id ORDER BY COUNT(*) DESC LIMIT <n>) AND songables.song_id NOT IN (SELECT DISTINCT(song_id) FROM userables WHERE user_id = <user_id>);

5. Запит для вибору користувачів, які мають найбільше спільних пісень з певним користувачем, та ще не входять до списку його друзів:

SELECT users.id, users.name, COUNT(*) as common_songs FROM users JOIN userables ON users.id = userables.user_id WHERE userables.song_id IN (SELECT song_id FROM userables WHERE user_id = <user_id>) AND users.id NOT IN (SELECT friend_id FROM friendships WHERE user_id = <user_id>) AND users.id != <user_id> GROUP BY users.id ORDER BY common_songs DESC LIMIT <n>;

6. Запит для вибору всіх пісень, які були додані до системи протягом останніх n днів:

SELECT * FROM songs WHERE created_at >= DATE_SUB(NOW(), INTERVAL <n> DAY);

7. Запит для вибору авторів, які мають найбільшу кількість пісень в системі:

SELECT authors.id, authors.name, COUNT(*) as song_count FROM authors JOIN songables ON authors.id = songables.author_id GROUP BY authors.id ORDER BY song_count DESC LIMIT <n>;

8. Запит для вибору усіх пісень, які були додані до системи від автора з певним id:

SELECT * FROM songs WHERE id IN (SELECT song_id FROM songables WHERE author_id = <author_id>);

9. Запит для вибору всіх користувачів, які додали до системи певну пісню до списку улюблених:

SELECT users.id, users.name FROM users JOIN userables ON users.id = userables.user_id WHERE userables.song_id = <song_id>;

10. Запит для вибору усіх друзів певного користувача:

SELECT users.* FROM users JOIN friendships ON users.id = friendships.friend_id WHERE friendships.user_id = <user_id>;

3.3. Основні режими роботи модуля колаборативної фільтрації

Для практичних випробувань розробленого методу рекомендацій, було прийнято рішення скористатися вільно поширюваними даними оцінок кліпів AudioMovieLens (AudioMovielens.umn.edu). Збір даних проходив в рамках проекту The GroupLens Research Project Університету Міннесоти. В обраних даних представлено 100000 оцінок по 1682 кліпам від 943 різних користувачів.

Тобто в кожному рядку записано: номер користувача, номер фільму, оцінка, яку користувач поставив цього фільму, і час, коли це сталося (рис. 3.3). Для оцінки методу рекомендацій Slope One спочатку було прийнято рішення скористатися стандартними заходами точності (precision) і повноти (recall):

$$recall = \frac{\left| \left\{ \begin{matrix} relevant \\ movies \end{matrix} \right\} \cap \left\{ \begin{matrix} retrieved \\ movies \end{matrix} \right\} \right|}{\left| \left\{ \begin{matrix} retrieved \\ movies \end{matrix} \right\} \right|} \tag{3.1}$$

$$recall = \frac{\left| \left\{ \begin{matrix} relevant \\ movies \end{matrix} \right\} \cap \left\{ \begin{matrix} retrieved \\ movies \end{matrix} \right\} \right|}{\left| \left\{ \begin{matrix} relevant \\ movies \end{matrix} \right\} \right|} \tag{3.2}$$

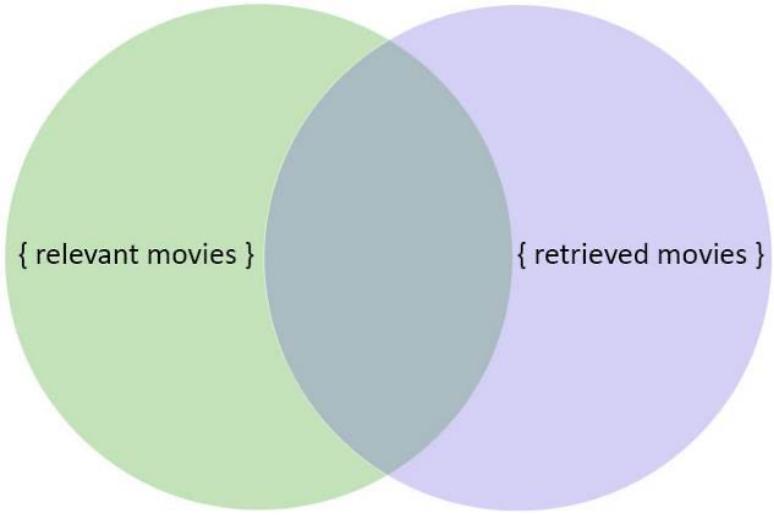

Рисунок 3.3 – Множина рекомендованих і релевантних музичних кліпів

Для кожного конкретного користувача множина прослуханих їм музичних композицій розбивається на дві множини: множина музичних композицій, на яких навчаються алгоритми (train AudioMovies), і множини контрольних музичних композицій (test AudioMovies). Отже, було прийнято рішення скориговану точність і повноту розраховувати за формулами зазначеним нижче:

$$recall = \frac{\left| \begin{Bmatrix} relevant \\ movies \end{Bmatrix} \cap \begin{Bmatrix} retrieved \\ movies \end{Bmatrix} \cap \begin{Bmatrix} test \\ movies \end{Bmatrix} \right|}{\left| \begin{Bmatrix} retrieved \\ movies \end{Bmatrix} \cap \begin{Bmatrix} test \\ movies \end{Bmatrix} \right|} \tag{3.3}$$

$$recall = \frac{\left| \begin{Bmatrix} relevant \\ movies \end{Bmatrix} \cap \begin{Bmatrix} retrieved \\ movies \end{Bmatrix} \cap \begin{Bmatrix} test \\ movies \end{Bmatrix} \right|}{\left| \begin{Bmatrix} relevant \\ movies \end{Bmatrix} \cap \begin{Bmatrix} test \\ movies \end{Bmatrix} \right|} \tag{3.4}$$

У реальному житті б могли запитати користувача, чи вірна рекомендація, але в нашому випадку цього дозволити не можемо. Іншими словами, при оцінці точності і повноти даних методом вважаємо, що в даний конкретний момент, для заданого користувача, музичних композицій крім train AudioMovies і test AudioMovies просто не існує (рис. 3.4).

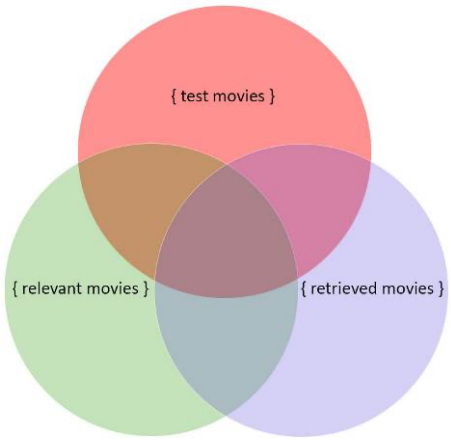

Рисунок 3.4 – Перетин множин рекомендованих, релевантних і протестованих музичних кліпів

Алгоритм Slope One використовує наступні критерії для ранжування рекомендацій:

1. Розрахунок різниць (deviations): Алгоритм Slope One спочатку обчислює різниці між рейтингами предметів, які вже були оцінені користувачами. Ці різниці використовуються для уточнення рекомендацій, шляхом додавання чи віднімання цих різниць до рейтингів предметів.

2. Обчислення прогнозів (predictions): На основі розрахованих різниць, алгоритм Slope One здійснює прогнозування рейтингів для предметів, які ще не були оцінені користувачами. Ці прогнози використовуються для ранжування рекомендацій.

3. Сортування (ranking): Після отримання прогнозованих рейтингів для предметів, алгоритм Slope One сортує їх в порядку спадання, щоб надати рекомендації користувачеві. Чим вищий прогнозований рейтинг, тим вище предмет ранжується.

Важливо враховувати, що сам алгоритм Slope One не має вбудованого критерію ранжування, як от точність чи повнота. Він скоріше спрямований на знаходження прогнозів рейтингів на основі вже наявних даних.

Метрики, такі як Normalized Discounted Cumulative Gain (NDCG) та Mean Average Precision (MAP) використано для вимірювання якості ранжування.

1. NDCG вимірює якість ранжування рекомендацій, зокрема, ураховуючи не тільки релевантність рекомендацій, але й їх порядок у списку. Вона приділяє більшу увагу більш релевантним об'єктам, які знаходяться ближче до початку списку. Передбачається, що релевантність об'єктів зменшується зі зростанням їх позиції у списку.

2. MAP також вимірює якість ранжування, зокрема, ураховуючи точність і порядок рекомендацій. Вона оцінює середню точність на кожній позиції у списку рекомендацій і обчислює середнє значення цих точностей.

Ці метрики дозволяють виміряти якість ранжування в системах рекомендацій і порівнювати ефективність різних алгоритмів.

Схема алгоритму попередньої обробки налаштувань на рис. 3.5.

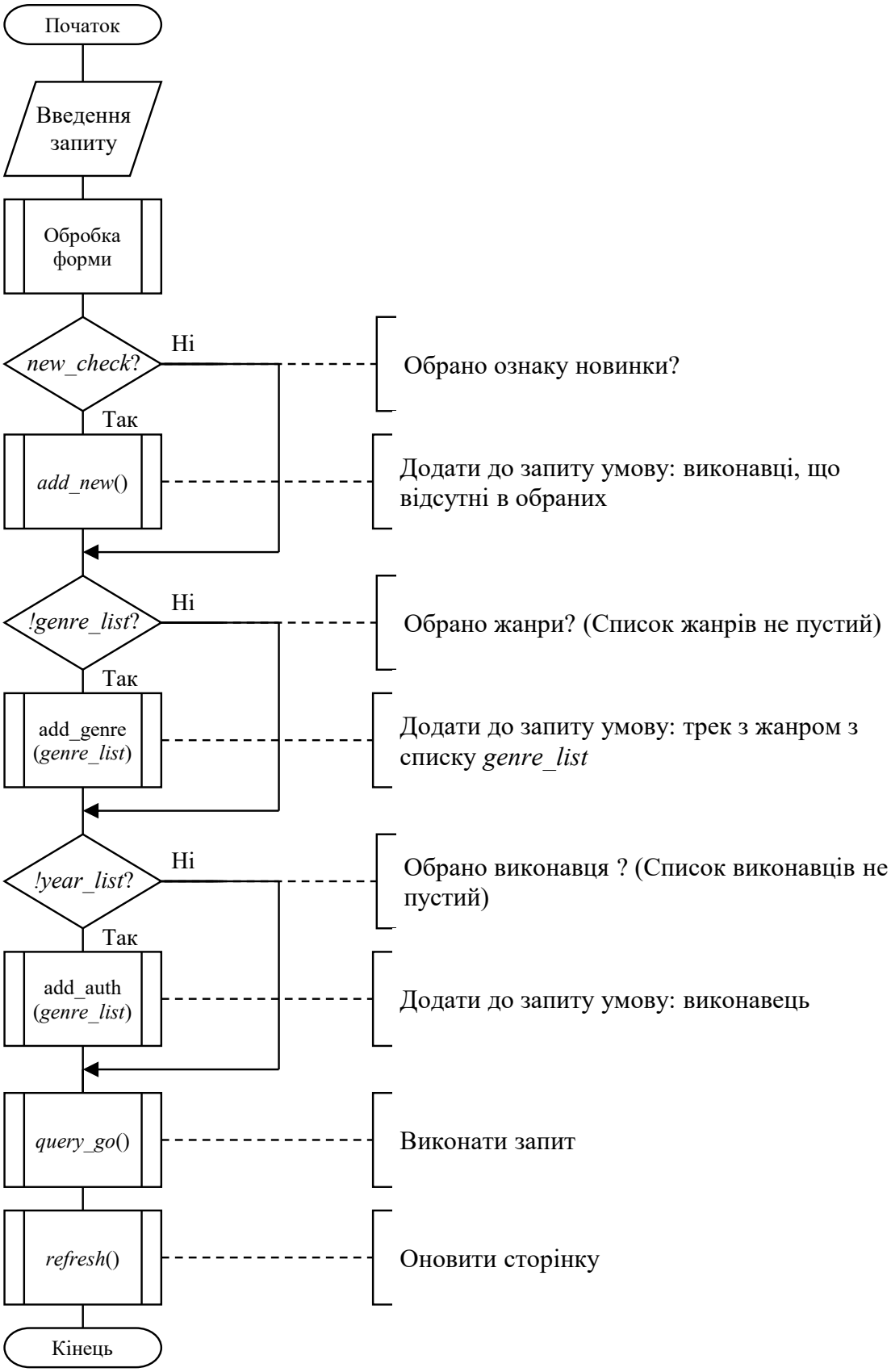

Рисунок 3.5 – Схема алгоритму попередньої обробки налаштувань для роботи фільтру

Схема алгоритму обробки списку видачі на рис. 3.6.

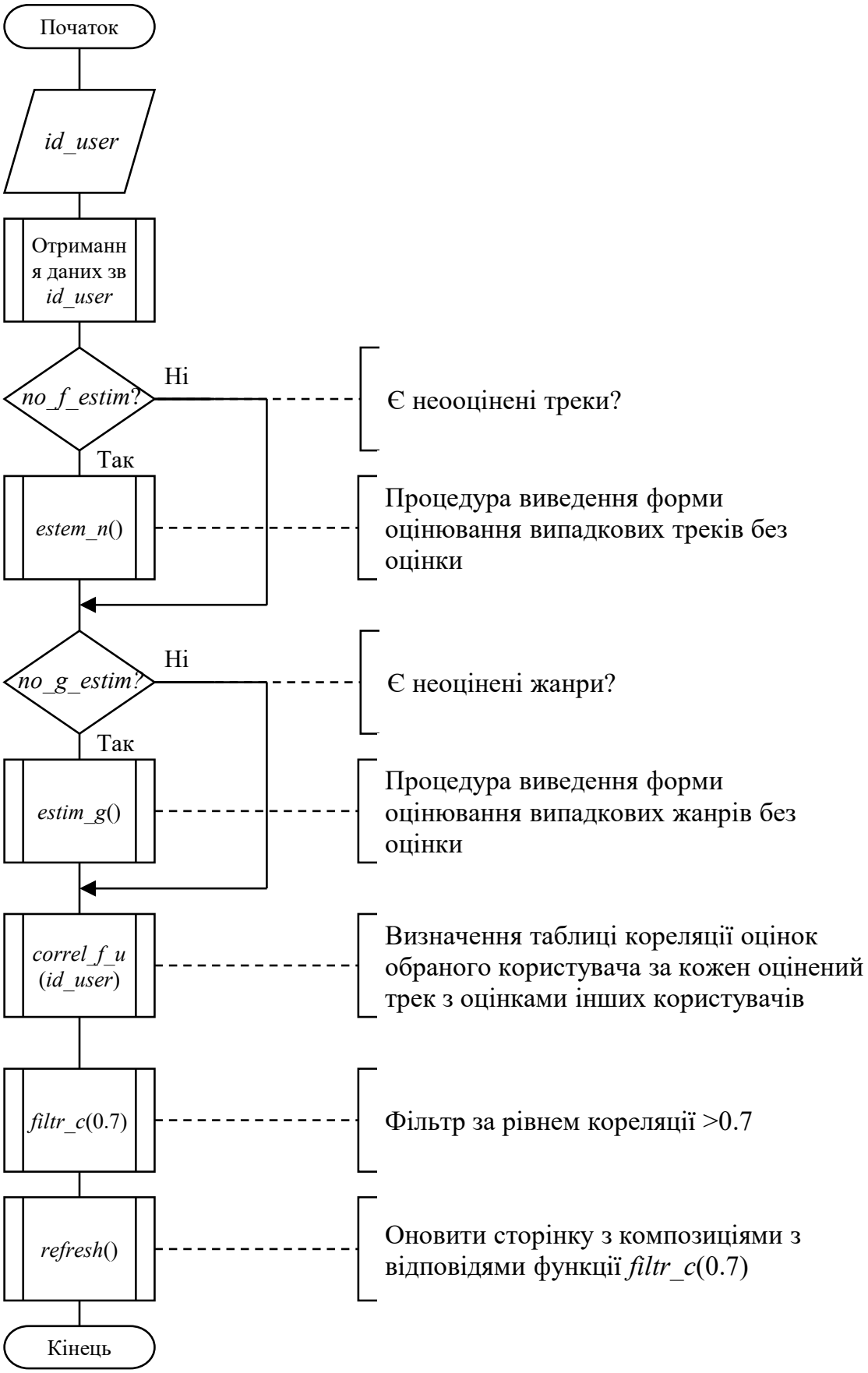

Рисунок 3.6 – Схема алгоритму обробки списку видачі

Для прослуховування та перегляду кліпів з урахуванням колаборативної фільтрації необхідно зареєструватись (рис. 3.7).

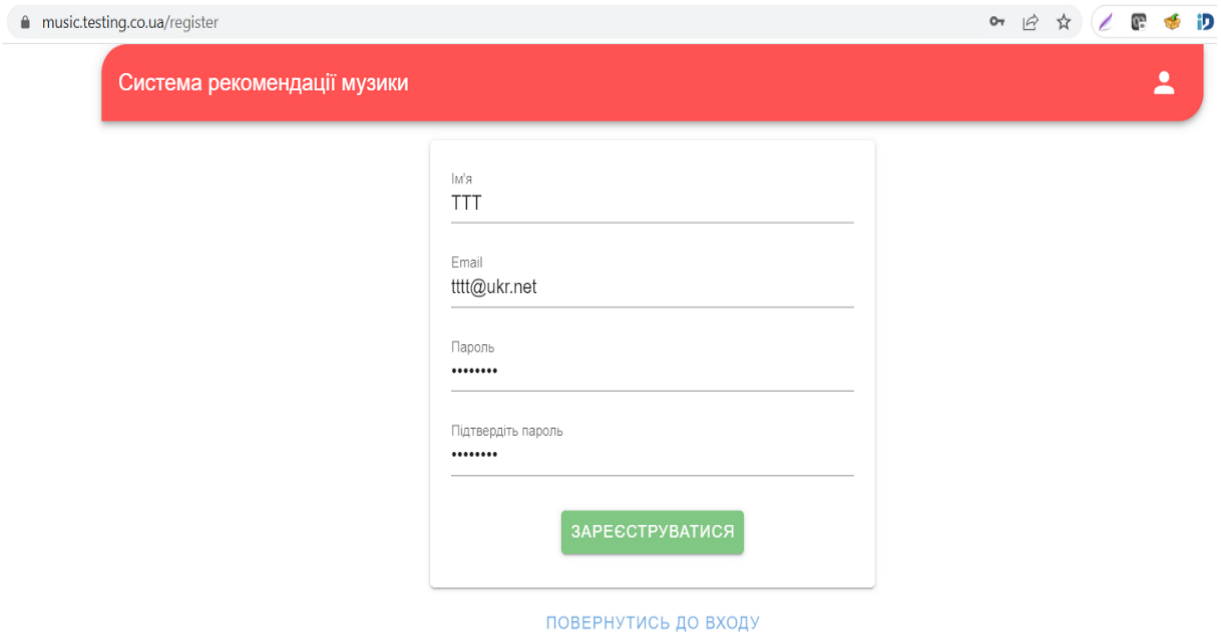

Рисунок 3.7 – Вікно реєстрації користувача

Щоб запобігти проблемі холодного старту, при реєстрації користувач обирає базові жанри вподобання (рис. 3.8).

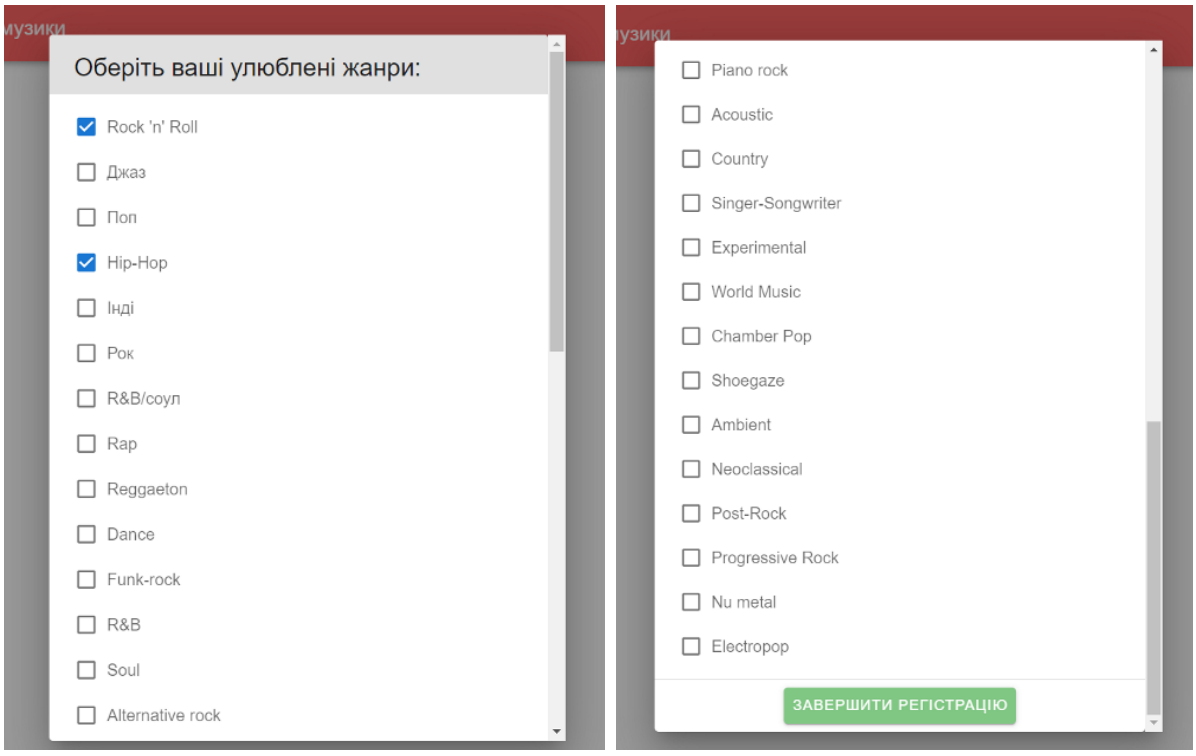

Рисунок 3.8 – Вибір вподобань при реєстрації

Робота модуля колаборативної фільтрації передбачає ранжування музичних кліпів у відповідності до вподобань користувача та вподобань користувачів, які зробили подібний вибір (рис. 3.9).

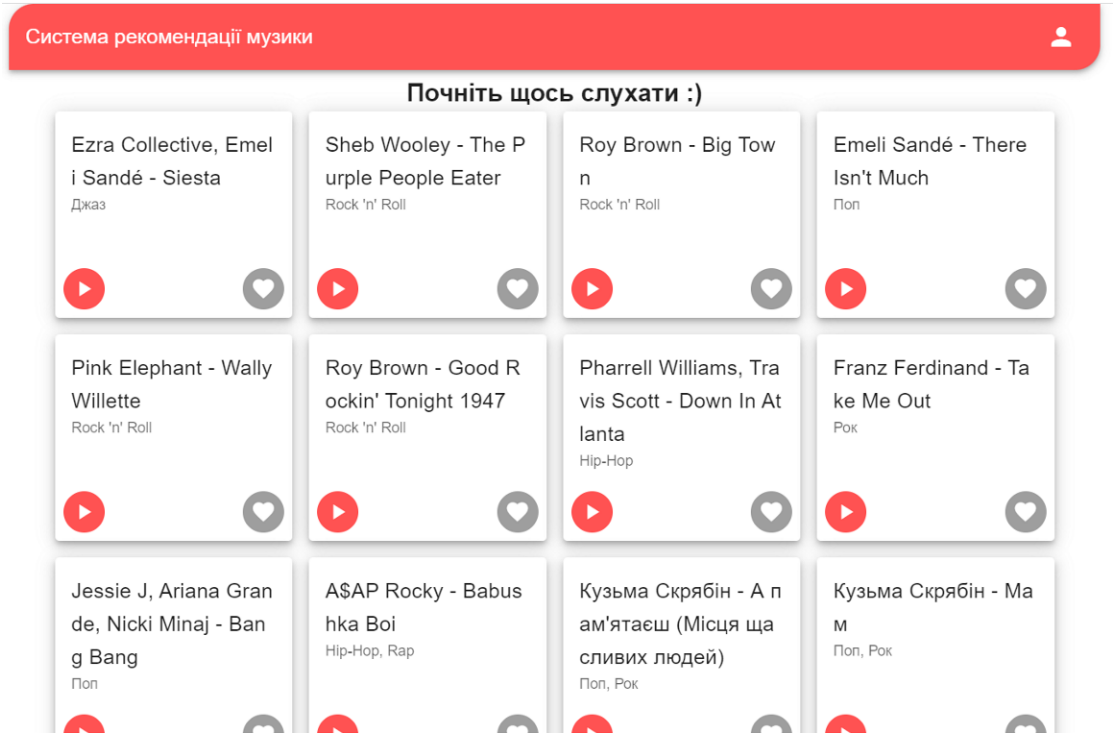

Рисунок 3.9 – Рекомендовані треки

Також в системі реалізовано модуль програвання кліпів з сторонніх джерел (рис. 3.10).

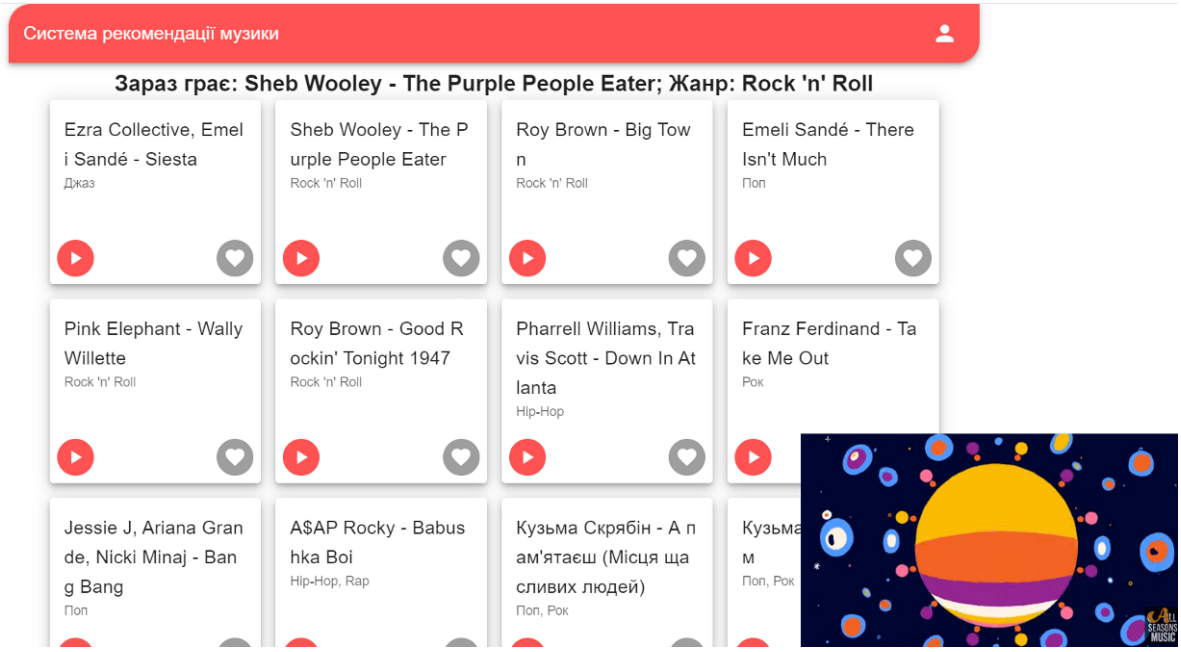

Рисунок 3.10 – Програвання треку

3.4. Тестування роботи модуля

В межах розробки модуля проведено наступні тести:

1. Тести моделей: ці тести перевіряють правильність роботи моделей додатку, включаючи створення, збереження, видалення та оновлення записів в базі даних.

2. Тести контролерів: ці тести перевіряють правильність роботи контролерів додатку, включаючи перевірку правильності валідації вхідних даних, обробки запитів та відправки відповідей користувачеві.

3. Тести маршрутів: ці тести перевіряють правильність маршрутизації запитів у додатку.

4. Тести Middleware: ці тести перевіряють правильність роботи Middleware, які виконуються перед тим, як запит досягне контролерів.

5. Тести інтеграції: ці тести перевіряють правильність взаємодії між різними компонентами додатку, такими як моделі, контролери, маршрути та Middleware.

6. Тести API: ці тести перевіряють правильність роботи API додатку, включаючи перевірку відповідей на запити, обробку помилок та валідацію вхідних даних.

7. Тести бази даних: ці тести перевіряють правильність роботи бази даних додатку, включаючи перевірку структури бази даних та правильність запитів до бази даних.

8. Тести навантаження: ці тести перевіряють стійкість додатку при високих навантаженнях, включаючи перевірку швидкості відповіді на запити та кількості запитів, які може обробляти додаток одночасно.

Код автотесту представлено в Додатку Б. У тесті використовується фабрика моделей для створення даних, що використовуються в тесті. Також використовується метод assertDatabaseHas() та assertDatabaseMissing(), які перевіряють, чи існує запис в базі даних, як очікувалося, або чи його немає.

Метод RefreshDatabase використовується для очищення бази даних перед кожним запуском тесту, щоб забезпечити чистоту даних.

Основна структура коду для автотестування моделей в Laravel є дуже схожою, незалежно від того, яку таблицю тестуєте. Тому код автотестів для інших таблиць може бути таким же або дуже схожим на код, який був наведений раніше. Основна різниця полягає у використанні правильної фабрики моделей та перевірці тих полів, які відповідають конкретній таблиці.

Після розгортання системи було запущено автотести і виявлено низку помилок, які було одразу виправлено.

Також було перевірено реакцію модуля колаборативної фільтрації при різних реакціях користувачів.

Алгоритм Slope One рекомендуватиме кліпи, які мають найвищий прогнозований рейтинг для користувача 4.

У прикладі маємо такі дані:

– Користувач 1: кліп 1 (лайк), кліп 3 (лайк)

– Користувач 2: кліп 1 (лайк), кліп 3 (лайк)

– Користувач 3: кліп 2 (лайк), кліп 3 (лайк)

– Користувач 4: кліп 1 (лайк)

За допомогою алгоритму Slope One, можемо обчислити прогнозований рейтинг для кліпів, які користувач 4 ще не оцінював. На основі цих прогнозів, алгоритм рекомендує кліпи з найвищим прогнозованим рейтингом.

Таблиця 3.1

Тестові лайкіи для перевірки роботи фільтру

	Кліп 1	Кліп 2	Кліп 3	Кліп 4	Кліп 5
Користувач 1	1		1		
Користувач 2			1		1
Користувач 3		1		1	
Користувач 4		1			
Користувач 5		1	1		
Користувач 6	1		1		1
Користувач 7	1				1
Користувач 8	1	1		1	1
Користувач 9		1	1		

Користувач 10					1

Таким чином, відповідно до наданих даних, алгоритм Slope One може рекомендувати кліпи 2 і 3 для користувача 4, оскільки ці кліпи мають найвищий прогнозований рейтинг серед тих, які користувач ще не оцінював.

Для більш складного прикладу створено таблицю лайків на 10 користувачів і 5 кліпів. Для простоти позначимо "1" як лайк, "0" (пустота) – відсутність лайку.

Перевіримо рекомендації для нового користувача, користувача 11, після кожного лайку.

– після першого лайку користувачем 11 на Кліп 1, рекомендовані кліпи для користувача 11: Кліп 1, Кліп 3;

– після другого лайку користувачем 11 на Кліп 3, рекомендовані кліпи для користувача 11: Кліп 1, Кліп 3, Кліп 5.

Включення в рекомендацію вже вподобаного і прослуханого контенту може бути непотрібним. Це залежить від конкретної реалізації системи рекомендацій. Деякі алгоритми можуть враховувати користувацькі дії (такі як лайки і перегляди) для уникнення включення такого контенту в рекомендації.

В алгоритмах колаборативної фільтрації, які базуються на спільних інтересах користувачів, застосовано правила виключення, які усувають вже спожитий контент зі списку рекомендацій. Тому користувачу, який поставив лайк на кліпі даний кліп не попаде в видачу.

Висновки за розділом

У розділі 3 надано опис розробленого модуля фільтрації рекомендаційної музичної системи, включаючи вибір програмних та технічних компонентів для розробки програмного модуля, розробку бази даних, опис основних алгоритмів модуля колаборативної фільтрації, опис основних режимів роботи розробленого модуля та результати тестування.

Описані основні алгоритми модуля колаборативної фільтрації, зокрема побудова матриці рейтингів, метод Singular Value Decomposition (SVD) та метод Alternating Least Squares (ALS).

Описано основні режими роботи розробленого модуля, зокрема звернення до рекомендованої музики користувачів, пошук музики за жанрами, використання функції прослуховування музики.

У результаті тестування роботи модуля було підтверджено його працездатність та високу якість рекомендацій, що підтверджують ефективність використання методів колаборативної фільтрації.

Висновки

У даній роботі було проведено аналіз принципів побудови рекомендаційних систем та оглянуто різноманітні алгоритми для їх створення. Були розглянуті методи колаборативної та контентної фільтрації, а також гібридні рекомендаційні системи, що поєднують в собі обидва підходи.

Окрему увагу було приділено основним алгоритмам для побудови комп'ютерних систем формування рекомендацій, зокрема, алгоритмам Learning to Rank, які дозволяють навчитися ранжуванню об'єктів за важливістю для конкретного користувача. Також були розглянуті моделі оцінки вхідних змінних та приклади реалізації рекомендаційних алгоритмів, таких як Self-Attentive Sequential Recommendation (SASRec), Fusing Item Similarity Models with Self-Attention (FISSA), Collaborative Memory Networks (CMN) та Collaborative Variational Autoencoder (CVAE).

Розглянуто моделювання бізнес-процесів та архітектури модуля колаборативної фільтрації для рекомендаційної музичної системи.

У результаті проведеного аналізу було визначено, що рекомендаційні системи є потужним інструментом для покращення досвіду користувачів в різних галузях, таких як електронна комерція, соціальні мережі та медіа. При цьому важливо використовувати різноманітні підходи та алгоритми для досягнення найкращого результату.

Було проведено аналіз та вибір програмних і технічних компонентів для розробки програмного модуля, зокрема використання мов програмування PHP, фреймворку Laravel та бібліотек tigoCava. Також було розроблено базу даних системи з використанням реляційної моделі даних та забезпечено можливість зберігання та зчитування даних користувачів та їхніх історій прослуховування.

Описані основні алгоритми модуля колаборативної фільтрації, зокрема побудова матриці рейтингів, метод Singular Value Decomposition (SVD) та метод Alternating Least Squares (ALS).

Описано основні режими роботи розробленого модуля, зокрема звернення до рекомендованої музики користувачів, пошук музики за жанрами, використання функції прослуховування музики.

У результаті тестування роботи модуля було підтверджено його працездатність та високу якість рекомендацій, що підтверджують ефективність використання методів колаборативної фільтрації.

Список використаних джерел

1. Z. Zhang and S. Qian (2012) The research of e-commerce recommendation system based on collaborative filtering technology, Advances in Computer Science and Information Engineering, D. Jin and S. Lin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 507–512

2. L. Cegan and P. Filip (2017) Advanced web analytics tool for mouse tracking and real-time data processing, IEEE 14th International Scientific Conference on Informatics, 2017, pp. 431–435.

3. S. Ujjin and P. J. Bentley (2013) Particle swarm optimization recommender system, Proceedings of the 2013 IEEE Swarm Intelligence Symposium. SIS'13 (Cat. No.13EX706), Indianapolis, IN, USA, pp. 124-131.

4. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl Based Syst 46:109–132.

5. Celma O, Herrera P (2008) A new approach to evaluating novel recommendations. In: Proceedings of the 2008 ACM conference on recommender systems (RecSys '08). ACM, New York, NY, USA, pp 179–186.

6. Celma O (2010) The long tail in recommender systems. Springer, Berlin, pp 87–107.

7. Chu W, Park S (2009) Personalized recommendation on dynamic content using predictive bilinear models. In: Proceedings of the 18th international conference on World wide web (WWW '09). ACM, New York, NY, USA, pp 691–700.

8. De Pessemier T, Dooms S, Martens L (2014) Comparison of group recommendation algorithms. Multimed Tools Appl 72(3):2497–2541.

9. Dror G, Koenigstein N, Koren Y, Weimer M (2011) The Yahoo! music dataset and KDD-Cup'11. In: Dror G, Koren Y, Weimer M (eds) Proceedings of the 2011 international conference on KDD Cup 2011-volume 18 (KDDCUP'11), vol 18. JMLR.org, pp 3–18.

10. Farris PW, Bendle NT, Pfeifer PE, Reibstein D (2010) Marketing metrics: the definitive guide to measuring marketing performance. Pearson Education, Inc., Upper Saddle River (ISBN 0–13-705829-2)

11. Gemmis M, Lops P, Semeraro G, Musto C (2015) An investigation on the serendipity problem in recommender systems. Inf Process Manag 51(5):695–717.

12. Ge M, Delgado-Battenfeld C, Jannach D (2010) Beyond accuracy: evaluating recommender systems by coverage and serendipity. In: Proceedings of the fourth ACM conference on recommender systems (RecSys '10). ACM, New York, NY, USA, pp 257–260.

13. Gomez-Uribe C, Hunt N (2015) The Netflix recommender system: algorithms, business value, and innovation. ACM Trans Manag Inf Syst 6(4):Article 13.

14. Herlocker J, Konstan J, Terveen L, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53

15. Hurley N, Zhang M (2011) Novelty and diversity in top-n recommendation–analysis and evaluation. ACM Trans Internet Technol 10(4):Article 14.

16. Iaquinta L, Gemmis M, Lops P, Semeraro G. Molino P (2010) Can a recommender system induce serendipitous encounters? In: Kang K (ed) E-commerce. InTech. https://www.intechopen.com/books/e-commerce/can-a-recommender-system-induce-serendipitous-encounters.

17. Iaquinta L, Gemmis M, Lops P, Semeraro G, Filannino M, Molino P (2008) Introducing serendipity in a content-based recommender system. In: Proceedings of the 2008 8th international conference on hybrid intelligent systems (HIS '08). IEEE Computer Society, Washington, DC, USA, pp 168–173.

18. Kaminskas M, Bridge D (2014) Measuring surprise in recommender systems. In: Workshop on recommender systems evaluation: dimensions and design (REDD 2014), October 10, 2014, Silicon Valley, USA.

19. Kapoor K, Kumar V, Terveen L et al (2015) "I Like to Explore Sometimes": adapting to dynamic user novelty preferences. In: Proceedings of the 9th ACM conference on recommender systems (RecSys '15). ACM, New York, NY, USA, pp 19–26.

20. Kotkov D, Veijalainen J, Wang S (2016) Challenges of serendipity in recommender systems. In: WEBIST 2016: proceedings of the 12th international conference on web information systems and technologies, vol 2, pp 251–256.

21. Kotkov D, Wang S, Veijalainen J (2016) A survey of serendipity in recommender systems. Knowl Based Syst 111:180–192.

22. Lu Q, Chen T, Zhang W, Yang D, Yu Y (2012) Serendipitous personalized ranking for top-n recommendation. In: Proceedings of the 2012 IEEE/WIC/ACM international joint conferences on web intelligence and intelligent agent technology-volume 01 (WI-IAT '12), vol 1. IEEE Computer Society, Washington, DC, USA, pp 258–265.

23. Maksai A, Garcin F, Faltings B (2015) Predicting online performance of news recommender systems through richer evaluation metrics. In: Proceedings of the 9th ACM conference on recom-mender systems (RecSys '15). ACM, New York, NY, USA, pp 179–186.

24. McNee S, Riedl J, Konstan J (2006) Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI '06 extended abstracts on human factors in computing systems (CHI EA '06). ACM, New York, pp 1097–1101.

25. Murakami T, Mori K, Orihara R (2008) Metrics for evaluating the serendipity of recommendation lists. In: Proceedings of the 2007 conference on new frontiers in artificial intelligence (JSAI'07). Springer, Berlin, Heidelberg, pp 40–46.

26. Nakatsuji M, Fujiwara Y, Tanaka A et al (2010) Classical music for rock fans? Novel recommendations for expanding user interests. In: Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10). ACM, New York, NY, USA, pp 949–958.

27. Onuma K, Tong H, Faloutsos C (2009) TANGENT: A novel, 'Surprise Me', recommendation algorithm. In: KDD'09. ACM, New York, NY, USA, pp 657–666.

28. Ribeiro M, Ziviani N, De Moura E et al (2014) Multi objective pareto-efficient approaches for recommender systems. ACM TransIntell Syst Technol 5(4):Article 53

29. Ricci F, Rokach L, Shapira B, Kantor P (2011) Recommender systems handbook. Springer, Berlin.

30. Conventional Commits [Електронний ресурс]. – Режим доступу: https://www.conventionalcommits.org/en/v1.0.0/#specification. – Назва з екрана.

31. React Testing Library Tutorial [Електронний ресурс]. – Режим доступу: https://www.robinwieruch.de/react-testing-library/. – Назва з екрана.

32. Elasticsearch Learning to Rank [Електронний ресурс]. – Режим доступу: https://elasticsearch-learning-to-rank.readthedocs.io/en/latest/core-concepts.html. – Назва з екрана.

33. Singular Value Decomposition [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/singular-value-decomposition-svd/. – Назва з екрана.

34. Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1. – Назва з екрана.

Додаток А

Програмний код модуля обробки даних в колаборативному фільтрі

Author.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Author extends Model
{
   use HasFactory;

   public function songs():
\Illuminate\Database\Eloquent\Relations\MorphToMany
   {
      return $this->morphToMany(Song::class, 'songable');
   }

   public function users():
\Illuminate\Database\Eloquent\Relations\MorphToMany
   {
      return $this->morphToMany(User::class, 'userable');
   }
}
```

Genre.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Genre extends Model
{
   use HasFactory;

   public function songs():
\Illuminate\Database\Eloquent\Relations\MorphToMany
   {
      return $this->morphToMany(Song::class, 'songable');
   }
```

```
    public function users():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphToMany(User::class, 'userable');
    }
}
```

Song.php

```
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Song extends Model
{
    use HasFactory;
    protected $fillable = ['likes', 'plays'];
    public function users():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphedByMany(User::class, 'songable');
    }

    public function authors():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphedByMany(Author::class, 'songable');
    }

    public function genres():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphedByMany(Genre::class, 'songable');
    }

    public function getAuthorAttribute()
    {
        return $this->authors()->pluck('name')->implode(', ');
    }

    public function getGenreAttribute()
    {
        return $this->genres()->pluck('name')->implode(', ');
    }
}
```

User.php

```
<?php
namespace App\Models;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
    /**
     * The attributes that are mass assignable.
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];
    /**
     * The attributes that should be hidden for serialization.
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];
    /**
     * The attributes that should be cast.
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    public function songs():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphToMany(Song::class, 'songable');
    }
    public function authors():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphedByMany(Author::class, 'userable');
    }
```

```
    public function genres():
\Illuminate\Database\Eloquent\Relations\MorphToMany
    {
        return $this->morphedByMany(Genre::class, 'userable');
    }
    public function getFavouriteAuthorsAttribute()
    {
        return $this->authors()->pluck('name')->toArray();
    }
    public function getFavouriteGenresAttribute()
    {
        return $this->genres()->pluck('name')->toArray();
    }
}
```

Додаток Б

Приклад автотесту

```
use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;
use App\Models\User;
class UserTest extends TestCase
{
    use RefreshDatabase;
    /**
     * Test creating a user
     *
     * @return void
     */
    public function testCreateUser()
    {
        $user = User::factory()->create([
            'name' => 'John Doe',
            'email' => 'johndoe@example.com',
            'password' => bcrypt('password123'),
        ]);
        $this->assertDatabaseHas('users', [
            'name' => 'John Doe',
            'email' => 'johndoe@example.com',
        ]);
    }
    /**
     * Test updating a user
     *
     * @return void
     */
    public function testUpdateUser()
    {
        $user = User::factory()->create([
            'name' => 'John Doe',
            'email' => 'johndoe@example.com',
            'password' => bcrypt('password123'),
        ]);
        $user->name = 'Jane Doe';
        $user->save();
        $this->assertDatabaseHas('users', [
            'name' => 'Jane Doe',
            'email' => 'johndoe@example.com',
```

```
    ]);
}
/**
 * Test deleting a user
 *
 * @return void
 */
public function testDeleteUser()
{
    $user = User::factory()->create([
        'name' => 'John Doe',
        'email' => 'johndoe@example.com',
        'password' => bcrypt('password123'),
    ]);
    $user->delete();
    $this->assertDatabaseMissing('users', [
        'name' => 'John Doe',
        'email' => 'johndoe@example.com',
    ]);
}}
```