Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(освітньо-кваліфікаційний рівень)

ГЮИК. 502610.009 ПЗ
(позначення документа)

«Дослідження та розробка гібридної рекомендаційної системи на прикладі музичного веб-сервісу»
(тема)

Виконав: студент 2 курсу, групи *СПРм-18-1*
Лі К.В.
(прізвище, ініціали)
Спеціальність 122 – *Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системне проектування*
(повна назва освітньої програми)

Керівник проф. *Ситніков Д.Е.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри СТ _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____ 2019 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Лі Кирилу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та розробка гібридної рекомендаційної системи на прикладі музичного веб-сервісу*
затверджена наказом по університету від « » 2019 р. №
2. Термін подання студентом роботи до екзаменаційної комісії *2019 р.*
3. Вихідні дані до роботи *Функція: Дослідження та розробка гібридної рекомендаційної системи на прикладі музичного веб-сервісу. Форма діалогу: веб-сервіс. Перелік використовуваних програмних засобів: ОС Ubuntu, інтегроване середовище розробки WebStorm, JavaScript, Python, Django framework, MySQL. Технічне забезпечення: комп'ютер з веб-браузером Google Chrome версії вище за 49, Safari версії вище за 10, Firefox версії вище за 55 чи Opera версії вище за 50 незалежно від операційної системи*
4. Перелік питань, що потрібно опрацювати в роботі *4.1 Вступ. 4.2 Огляд і аналіз сучасного стану проблеми 4.3 Потокові музичні сервіси 4.4 Музичні рекомендаційні системи 4.5 Загальний огляд існуючих сервісів 4.6 Постановка задачі 4.7 Дослідження методів та алгоритмів рішення задачі 4.8 Задача рекомендаційних систем 4.9 Основна проблема рекомендаційних систем 4.10 Проблема «холодного запуску» 4.11 Класифікація рекомендаційних систем 4.12 Аналіз методів машинного навчання для рекомендаційних систем 4.13 Оцінка точності роботи рекомендаційної системи 4.14 Розробка алгоритму гібридної рекомендаційної системи 4.15 Загальний алгоритм гібридної моделі 4.16 Проектування гібридного рекомендаційного веб-сервісу 4.17 UML-проектування 4.18 Проектування бази даних 4.19 Опис програмної реалізації рекомендаційного сервісу 4.20 Реалізація основного алгоритму на мові програмування Python 4.21 Організація роботи з даними веб-додатку. 4.22 Реалізація клієнтського SPA-додатку з використанням React.js 4.23 Перевірка отриманих результатів. 4.24 Висновки*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів). *5.1 Проблема «довгого хвоста» 5.2 «Сервіси, що використовують «рекомендації» 5.3 Схема роботи рекомендаційної системи 5.4 Розроблений алгоритм рекомендаційної системи 5.5 Діаграма прецедентів 5.6 Проектування бази даних 5.7 Програмна реалізація 5.8 Результати. Інтерфейс 5.9 Результати. Графік*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання « » 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломне проектування	01.09.2019	
2	Аналіз завдання, пошук літератури та аналогів з теми дипломної роботи	02.09 – 08.09.2019	
3	Опрацювання літератури та аналіз об'єкту дослідження	09.09 – 17.09.2019	
4	Формування постановки задачі	18.09 – 19.09.2019	
5	Дослідження методів та алгоритмів вирішення задачі	20.09 – 13.10.2019	
6	Розробка гібридного алгоритму системи	14.10 – 24.10.2019	
7	Проектування рекомендаційного сервісу	25.10 – 03.11.2019	
8	Розробка та тестування програмного засобу	04.11 – 11.11.2019	
9	Аналіз результатів, отриманих за допомогою програмного засобу	12.11 – 16.11.2019	
10	Оформлення пояснювальної записки	17.11 – 28.11.2019	
11	Оформлення графічної частини та презентаційних матеріалів комп'ютерного захисту	29.11 – 04.12.2019	
12	Представлення на рецензування	06.12.2019	
13	Представлення дипломної роботи в деканат	10.12.2019	

Студент _____
(підпис)

Керівник роботи _____ проф. *Ситніков Д.Е.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської атестаційної роботи містить: 62 сторінки, 19 рисунків, 7 таблиць, 4 додатка, 26 джерел. Графічна частина атестаційної роботи містить 9 плакатів.

Об’єктом дослідження є процес генерації персональних рекомендацій на основі вподобань користувачів

Предметом дослідження є методи та моделі формування особистих рекомендацій з використання гібридного підходу

Мета дослідження – проаналізувати існуючі методи та підходи до реалізації рекомендаційних систем, здійснити порівняльний аналіз обраних методів та алгоритмів їх реалізації, розробити ефективний гібридний алгоритм формування рекомендацій з метою підвищення точності роботи системи, розробити прототип рекомендаційної музичної системи, виконати аналіз отриманих результатів.

Методи дослідження – системний підхід, методи структурного аналізу і моделювання

Галузь застосування – реалізований гібридний алгоритм може бути використаний в будь-якому сучасному веб-сервісі, що потребує реалізацію ефективного рекомендаційного механізму із застосування гібридного підходу та алгоритмів машинного навчання

РЕКОМЕНДАЦІЙНІ СИСТЕМИ, АЛГОРИТМИ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КОНТЕНТНА ФІЛЬТРАЦІЯ, SVD, GSD, ВЕБ-СЕРВІС, PYTHON, NUMPY, MYSQL, REST API.

ABSTRACT

Thesis contains: 62 pages, 19 images, 7 tables, 4 applications, 26 sources. Graphic part of the thesis contains 9 posters.

The object of the research is the process of generating personal recommendations based on user preferences

The subject of the research is methods and models for generating personal recommendations with the usage of a hybrid approach

The purpose of the research is to analyze existing methods and approaches to the implementation of the recommendation systems, to perform a comparative analysis of the selected methods and algorithms for its implementation, to develop an effective hybrid algorithm of building recommendations to improve the accuracy of the system, to develop a prototype of the recommender music system, to analyze the results.

Research methods – systematic approach, methods of structural analysis and modeling

Field of application – the implemented hybrid algorithm can be used in any modern web service, which requires the implementation of an effective recommendation mechanism which is based on the hybrid approach and machine learning algorithms

RECOMMENDER SYSTEMS, ALGORITHMS, ITEM-BASED FILTERING, CONTENT-BASED FILTERING, WEB-SERVICE, SVD, GSD, PYTHON, NUMPY, MYSQL, REST API.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SVD – singular value decomposition
SGD – stochastic gradient descent
MF – matrix factorization
API – application programming interface;
CSS – Cascading Style Sheets;
ER – Entity-Relationship;
HTML – HyperText Markup Language;
JS – JavaScript;
JSON – JavaScript object notation;
SQL – Structured Query Language;
URL – Uniform Resource Locator;
REST - Representational State Transfer;
SPA – Single Page Application;
UX – User Experience;
UML - Unified Modeling Language;
БД – база даних;
СКБД – система керування базами даних.

ЗМІСТ

ВСТУП

Швидкий розвиток веб-технологій та поширення популярності веб-ресурсів призвели до того, що Інтернет став головним джерелом інформації, а обсяг доступної для користувача інформації став просто неосяжним.

Це призводить до того, що користувач опиняється перед проблемою вибору серед незліченної кількості товарів, фільмів або рекламних пропозицій. Величезна кількість доступних альтернатив ускладнює вибір дійсно актуальних варіантів серед безлічі інших.

В якості вирішення цієї проблеми інтеграція системи рекомендацій стала ефективною стратегією подолання такого перевантаження інформацією. Сьогодні майже кожен користувач Інтернету знайомий з такими системами. Наприклад, Amazon застосовує рекомендації, щоб запропонувати клієнтові інші товари, які також можуть бути йому цікаві, а популярний відео-сервіс Netflix рекомендує користувачам нові фільми, які можуть їх зацікавити.

Оскільки якісні рекомендації підтримують зацікавленість клієнтів та збільшують ймовірність конверсії, рекомендаційні системи є дуже актуальним інструментом для будь-якого бізнесу. Саме тому алгоритми та методи генерації рекомендацій перебувають у постійному розвитку та стають більш складними, прагнучи підвищити точність та актуальність рекомендацій. Попри можливу складність цих методів, основна ідея багатьох із них полягає в тому, щоб використовувати дані, отримані від інших людей та персоналізувати їх для індивідуального користувача.

Метою даної роботи є дослідження та розробка ефективного методу генерації рекомендацій, а також застосування рекомендаційної системи до музичного веб-сервісу.

Предметом дослідження виступає модель гібридної рекомендаційної системи, що застосовує методи машинного навчання та генерує рекомендації для користувачів музичного веб-сервісу на основі їх персональних вподобань.

1 ОГЛЯД І АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ

Протягом останніх десятиліть, із зростанням та розвитком технологій, зокрема Інтернету, наше оточення та методи взаємодії з сучасним світом стають все більш «цифровими». Використовуючи інтернет ми щодня здійснюємо покупки, спілкуємось, переглядаємо новини, слухаємо музику, сплачуємо рахунки тощо. Як наслідок, величезний потік інформації та даних є невід'ємною частиною нашого повсякденного життя, що призводить до проблеми перенасичення.

Сучасні користувачі стикаються з проблемою пошуку та вибору дійсно корисної для себе інформації. Іноді вони приймають рішення, не усвідомлюючи про існування інших, можливо кращих, альтернатив. Тому сьогодні досить велика кількість людей довіряє процес прийняття рішень третім сторонам, зокрема веб-сервісам.

Саме через відсутність обізнаності та постійне збільшення обсягу доступної інформації, що пропонують сучасні технології, такі як Інтернет, велику значимість набуває наявність методів фільтрації та відбору, які полегшують процес прийняття рішень.

Користувачі, які тепер мають доступ до величезного обсягу інформації, просто губляться серед безлічі альтернатив та пропозицій. Однак поява нових та практичних інструментів дозволяє підійти до вирішення цього питання більш ефективно. Ці інструменти спроможні задавати для користувачів певні напрямки, які допоможуть у пошуку найбільш необхідної та корисної інформації.

Отже, сукупність таких факторів, як доступ до великої кількості інформації, наявність безлічі альтернатив та недосвідченість користувачів у деяких предметних галузях досить ясно визначають необхідність у розвитку систем, які здатні допомагати користувачам у процесі прийняття рішень шляхом надання напрямів та пропозицій [1]. Рішенням саме цієї проблеми

можуть стати рекомендаційні системи, принципи побудування яких буде розглянуто далі в даній роботі. Все більше сучасних сервісів пропонують користувачам індивідуальні рекомендації, як спосіб полегшення взаємодії з сервісом та вирішення їх потреб.

Серед основних переваг впровадження рекомендаційних систем у сучасних онлайн-сервісах можна виділити наступні [2]:

- Збільшення конверсії. Для комерційних проектів, перш за все для інтернет-магазинів, найважливішою метою є можливість збільшити обсяг продажів за рахунок надання користувачам списку рекомендованих товарів. З великою ймовірністю отримані на основі попередніх транзакцій пропозиції відповідатимуть потребам користувача та як мінімум зацікавлять його [4]. Ті ж принципи можемо віднести і до некомерційних сервісів, які так само зацікавленні в збільшенні конверсії. Так, наприклад, інформаційний портал новин націлений на збільшення переглядів статей та передплатників.

- Збільшення задоволеності користувача. Якісно спроектована рекомендаційна система може покращити реалізацію потреб користувача та загальне враження від використання сервісу. Персоналізовані та корисні рекомендації в комбінації із зручним та зрозумілим інтерфейсом однозначно прийдуть до вподоби користувачеві. В свою чергу це призведе до збільшення довіри та частоти використання сервісу, а також ймовірності прийняття наступних рекомендацій

- Різноманітність пропозицій. Ще одною важливою функцією рекомендаційних систем є можливість для користувача отримати специфічні цікаві пропозиції, які важко знайти серед більш популярних просто переглядаючи усі доступні варіанти. Отже рекомендації є вирішенням так званої проблеми “довгого хвоста”, суть якої полягає в тому, що значна частина доступних, проте маловідомих товарів “губиться” серед відносно маленької частки більш відомих товарів.

- Краще розуміння потреб користувача. Дані про вподобання користувача, які можуть збиратись явно, неявно або прогнозуватись системою

також можуть бути використані в цілях покращення та оптимізації процесів виробництва, керування тощо.

Отже, визначивши основні переваги, які можуть бути отримані завдяки запровадженню рекомендаційних систем, не викликає здивування той факт, що все більше сучасних веб-сервісів в тій чи іншій степені використовують рекомендаційні механізми.

Сьогодні сфера їх використання є дуже різноманітною: електроні магазини, новинні портали, соціальні мережі, туристичні сервіси, медіа-сервіси (фільми, музика, книги). В даній роботі, в якості предметної області для дослідження рекомендаційних систем розглянемо музичні веб-сервіси.

1.1 Потокові музичні сервіси

Потокове передавання музики або точніше передавання аудіо - це спосіб передачі звуку, включаючи музику, не вимагаючи завантаження файлів з Інтернету. Музичні сервіси, такі як Spotify, Pandora та Youtube Music, використовують цей метод для надання пісень, якими можна насолоджуватись на всіх типах пристроїв.

Раніше, якщо людина хотіла послухати музику чи будь-який інший тип аудіо, вона завантажувала аудіофайл на свій пристрій у таких форматах, як MP3, WMA або FLAC. Однак при використанні потокових музичних сервісів, завантажувати файли не потрібно. Користувач може почати слухати улюблені композицій через будь-який сучасний пристрій майже відразу.

Потокова трансляція музики відрізняється від завантаження тим, що аудіозаписи не зберігаються фізично на жорсткому диску користувача. Користувач може отримувати доступ до аудіозаписів знов і знов, використовуючи лише свій електронний пристрій. Деякі, зазвичай платні, музичні сервіси потокової передачі дозволяють як прослуховувати онлайн, так і завантажувати музику для подальшого використання офлайн.

Принцип роботи потокової передачі полягає в тому, що аудіофайл доставляється невеликими пакетами, тому дані буферізуються на пристрої користувача та можуть бути відтворені в значній мірі відразу. Допоки користувач має надійне підключення до мережі, він матиме безперервний доступ до улюблених аудіозаписів.

1.2 Музичні рекомендаційні системи

На відміну від інших областей, в яких використовуються системи рекомендацій, такі як товари, фільми або готелі, рекомендації в музичній області мають певні специфічні характеристики, які слід враховувати при створенні музичної рекомендаційної системи. Деякі з цих особливостей у галузі музики впливають на застосування рекомендаційних систем та методів машинного навчання.

Серед цих особливостей можна виділити:

- тривалість музичної доріжки значно коротша, ніж тривалість фільму, відпустки або термін використання продукту;

- кількість предметів у комерційних музичних каталогах налічує десятки мільйонів композицій. Короткий час споживання та велика кількість доступних пісень означає, що рекомендації пісень, які не ідеально відповідають смаку користувача, як правило, не впливають на користувальницький досвід надто негативно. На відміну від рекомендацій щодо фільму, наприклад, коли користувачам потрібно набагато більше часу, щоб оцінити рекомендований фільм, і тому вони можуть бути більш засмучені через погані рекомендації.

- слухачам можуть подобатись повторювані рекомендації, на відміну від тих же фільмів, де користувачі зазвичай нехтують повторюваними рекомендаціями одних і тих же елементів.

- музика часто прослуховується в послідовності, наприклад, за альбомами або списками пісень. Тому рекомендації змістовних послідовностей пісень — важливе завдання для музичної сфери.

- споживання музики доволі часто є пасивним процесом, тобто слухач не приділяє цьому великої уваги, наприклад, слухаючи музику десь на фоні. Це може бути критично важливим при отриманні неявних зворотних даних, наприклад, пісня, яка відтворюється від початку до кінця, не обов'язково вказує на те, що слухач активно слухав цю пісню. Інтеграція додаткової контекстної інформації, такої як діяльність користувача під час прослуховування музики на пристроях є рішенням для усунення цієї проблеми.

1.3 Загальний огляд існуючих сервісів

Оскільки сьогодні музикальні потокові сервіси стають домінуючим та основним середовищем для прослуховування улюблених пісень, ці сервіси потокової передачі музики можуть збирати велику кількість даних про музичні звички своїх клієнтів.

Такі потокові сервіси, як Last.fm, Youtube-music або Pandora, використовують ці дані для надання рекомендацій своїм слухачам. Ці музичні системи рекомендацій є частиною більш широкого класу рекомендаційних систем, які фільтрують інформацію для прогнозування вподобань користувача.

Рекомендаційні системи поділяють на два основних класи, що використовують:

- колаборативну фільтрацію (аналізує вподобання користувача та історію його дій);

- контентну фільтрацію (аналізує елементи системи для того, щоб визначити ті, що найбільш співпадають із вподобаннями користувача);

1.3.1 Аналіз сервісу Last.fm

Популярний музичний сервіс Last.fm займається збиранням інформації о музичних композиціях, які прослуховують користувачі з подальшим її використанням у побудуванні рекомендацій та наданні відкритого API. Цей

сервіс створює музичну колекцію з рекомендованих пісень, спостерігаючи за тим, які музичні групи або окремі треки користувач слухає регулярно, і порівнюючи їх із музичною поведінкою інших користувачів. Last.fm рекомендуватиме композиції, які відсутні в бібліотеці користувача, але часто прослуховуються іншими користувачам зі схожими інтересами. Оскільки такий підхід використовує поведінку та вподобання користувачів, він є прикладом чистої колаборативної фільтрації.

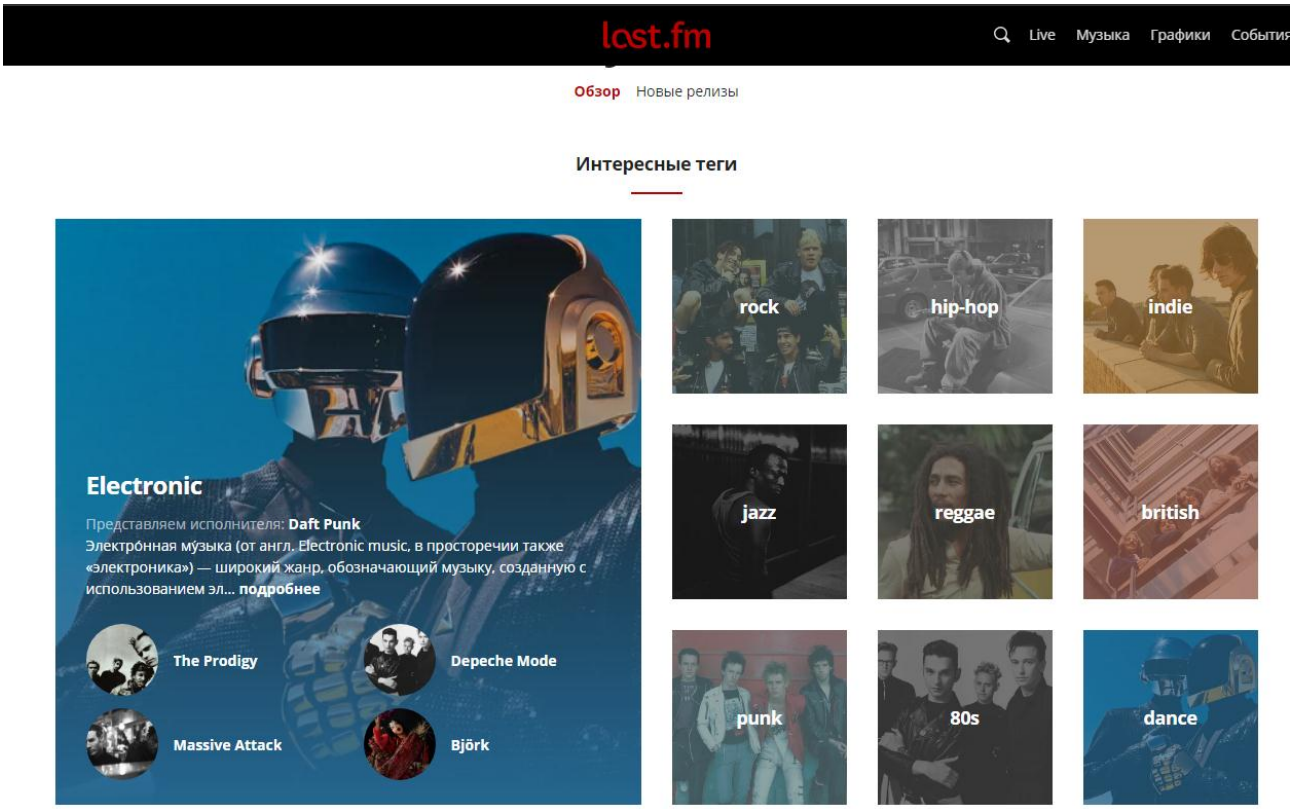

Рисунок 1.1 – «Музичний сервіс Last.fm»

1.3.2 Аналіз сервісу Pandora

Популярний музичний сервіс Pandora використовує властивості пісні або виконавця (колекція мета-атрибутів, що надаються проектом "Music Genome Project"), щоб створити станцію, яка складається з музики, що має подібні властивості. Відгуки користувачів використовуються для підвищення точності рекомендацій, знижуючи “важливості” певних атрибутів, якщо користувач не

оцінив певну пісню, та підкреслюючи інші атрибути, якщо користувачеві до вподоби та чи інша пісня. Це типовий приклад підходу контентної фільтрації.

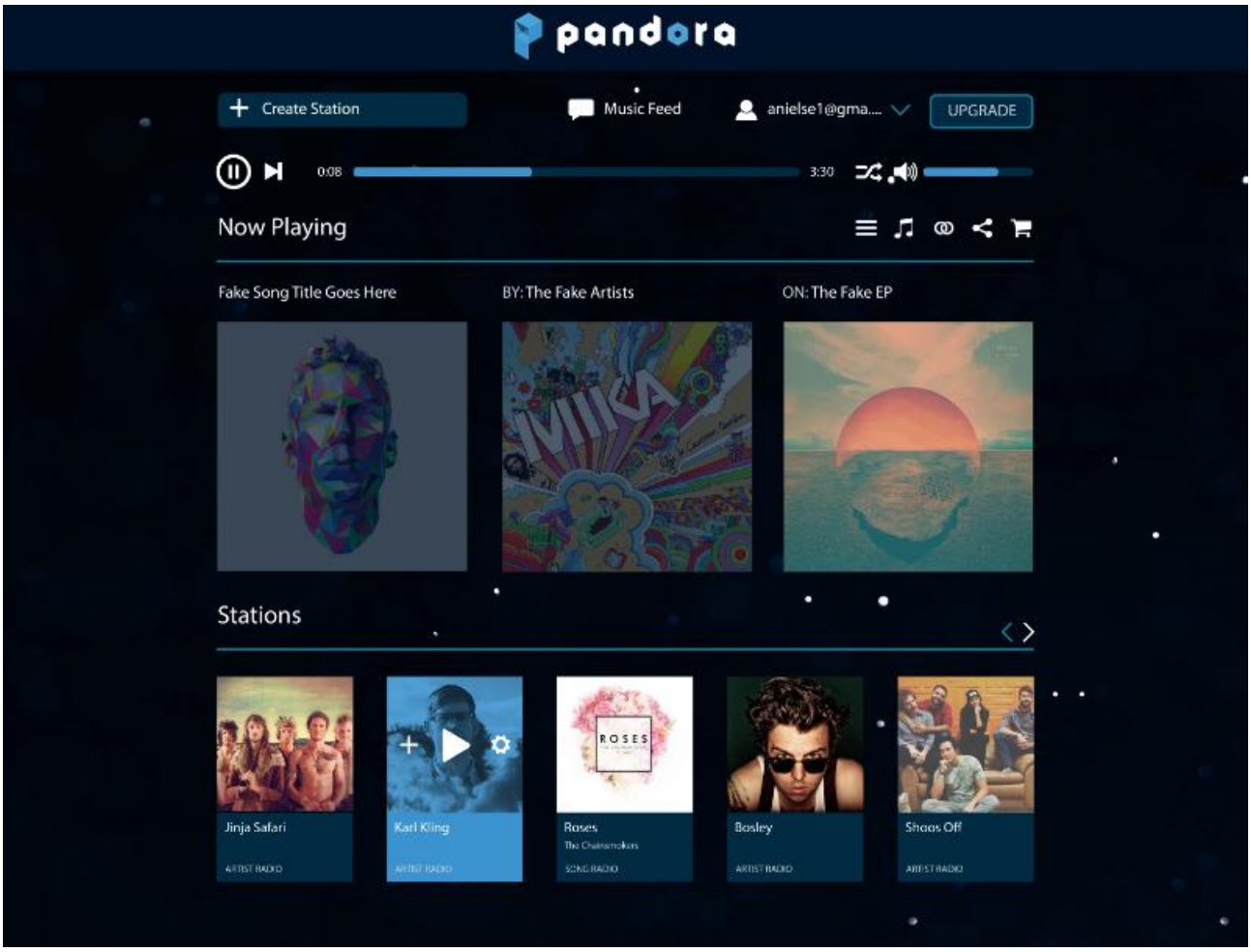

Рисунок 1.2 – «Музичний сервіс Pandora»

1.3.3 Аналіз сервісу Youtube music

Ще один сучасний музичний сервіс — Youtube music застосовує гібридний підхід, поєднуючи обидва способи побудування рекомендацій. Таким чином, сервіс підбирає списки пісень, опираючись на схожість користувача із іншими слухачами (колаборативна фільтрація), а також за спільними властивостями власне самих пісень (контентна фільтрація), дозволяючи налаштувати вподобання користувача за жанрами та артистами.

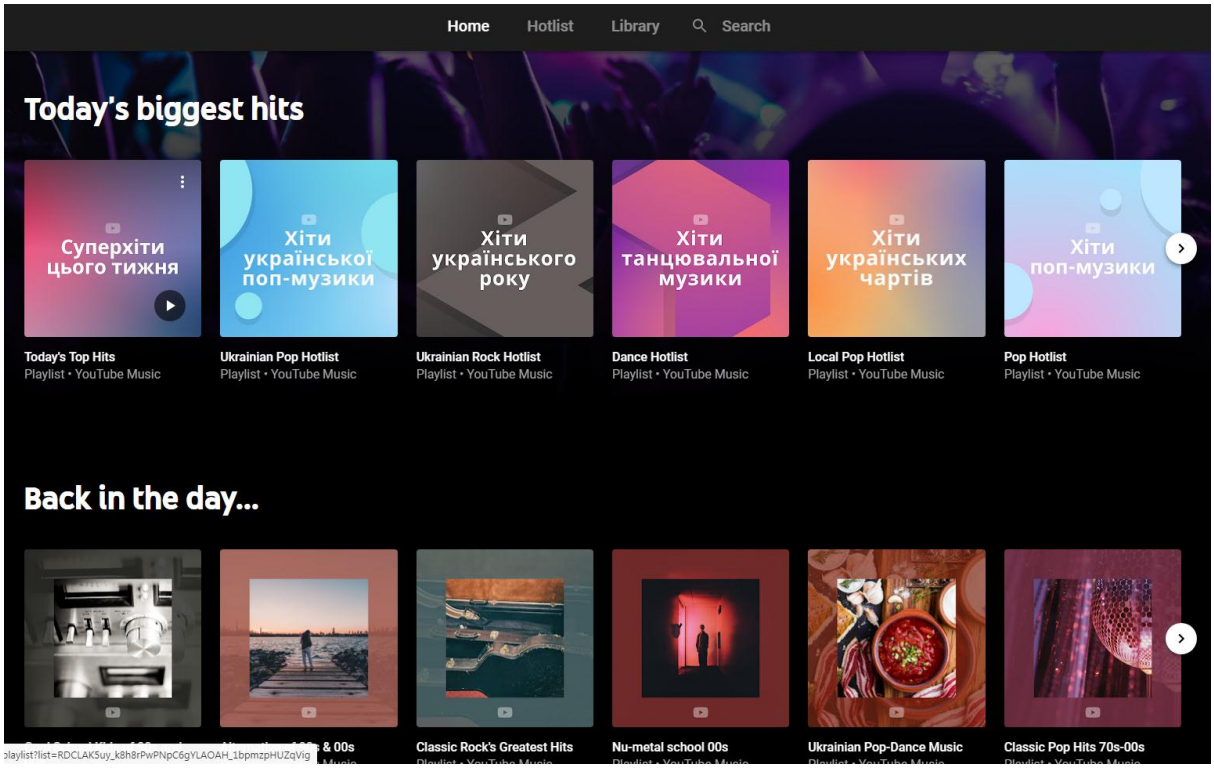

Рисунок 1.3 – «Музичний сервіс Youtube-music»

1.4 Постановка задачі

Враховуючи мету даної кваліфікаційної роботи, в процесі її написання необхідно:

- провести дослідження методів побудови та оцінки рекомендаційних систем;

- розробити універсальну модель гібридної системи;

- провести оцінювання ефективності роботи системи;

- програмно реалізувати прототип рекомендаційного музичного веб-сервісу, який використовуватиме розроблений гібридний алгоритм;

В процесі побудови математичної моделі системи необхідно вирішити наступну задачу:

Нехай C - сукупність усіх користувачів, S є набором усіх можливих елементів, які можуть бути рекомендовані (пісні, артисти), аu буде деякою функцією, що визначає корисність предмету s для користувача c.

$u: C \times S \to R$, де R - повністю упорядкована множина (наприклад, позитивні цілі числа в деякому діапазоні). Тоді для кожного користувача $c \in C$ ми намагаємось обрати такий елемент $s' \in S$, що максимізує функцію корисності для цього користувача [2].

Тобто:

$$\forall c \in C, s'_c = \underset{s \in S}{argmax} u(c, s), \tag{1.1}$$

Функція корисності u позначає наскільки сильно певний елемент може зацікавити користувача та може мати довільну форму, виходячи з алгоритму.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ТА АЛГОРИТМІВ РІШЕННЯ ЗАДАЧІ

Зародження рекомендаційних систем, як предмету досліджень можна простежити звернувши увагу до таких галузей як когнітивна наука, теорія наближень, інформаційний пошук, теорії прогнозування та моделювання вибору споживача в маркетингу. Проте рекомендаційні системи виникли, як незалежна область досліджень в середині 1990-х років, коли дослідники почали зосереджуватись на проблемі побудови рекомендацій, які явно залежали від системи оцінювання елементів [2].

Рекомендаційні системи, в загальному випадку, визначались як експертні системи, призначені для рекомендації продуктів або послуг користувачам.

Проблема, що повинна бути вирішена такими системами, зводиться до проблеми оцінювання рейтингу предметів, з якими користувач ще не знайомий. Зазвичай, ця оцінка базується на рейтингу, що користувач визначив для інших предметів та на деякій допоміжній інформації. Якщо ми визначимо рейтинги для елементів, які ще не були рецензовані, то ми можемо порекомендувати користувачу елементи з найвищим передбачуваним рейтингом [3].

На рисунку 2.1 зображено схему роботи традиційної системи рекомендацій.

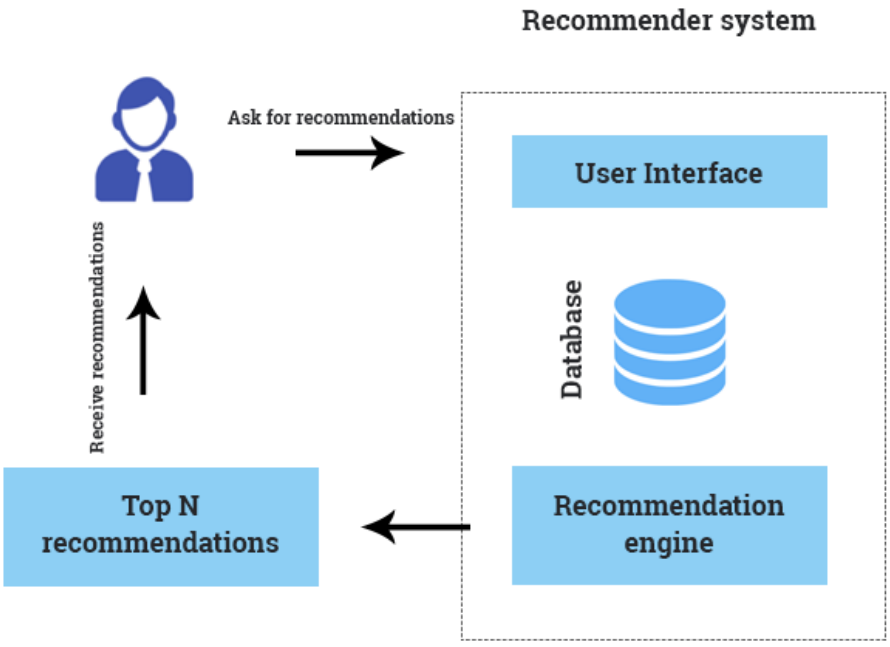

Рисунок 2.1 - «Схема роботи традиційної системи рекомендацій»

На вибір користувача щодо запропонованого ряду продуктів або послуг впливають численні фактори. Використовуючи інформацію, надану спільнотою (блоги, соціальні мережі), історію географічного розташування, фотографії з гео-тегами, система намагається допомогти користувачу, генеруючи персоналізовані рекомендації, які будуть корисними у процесі прийняття рішень та вибору.

2.1 Задача рекомендаційних систем

Нехай C - сукупність всіх користувачів і нехай S є набором усіх можливих елементів, які можуть бути рекомендовані, наприклад, фільми, пісні або міста.

Кількість елементів у множині S може бути дуже великою, досягати обсягу у тисячі або навіть мільйони елементів. Так само потужність множини C може бути дуже великою.

Нехай u буде деякою функцією, що визначає корисність предмету s для користувача c.

$u: C \times S \to R$, де R - повністю упорядкована множина (наприклад, позитивні цілі числа в деякому діапазоні). Тоді для кожного користувача $c \in C$ ми намагаємось обрати такий елемент $s' \in S$, що максимізує функцію корисності для цього користувача [1]. Тобто:

$$\forall c \in C, s'_c = \underset{s \in S}{argmax} u(c, s), \tag{2.1}$$

В рекомендаційних системах корисність елементу зазвичай визначається деякою оцінкою, яка позначає наскільки сильно певному користувачу подобається певний елемент. Проте функція корисності u може бути й довільною. Таким чином, u може або визначатися користувачем, або розраховуватися додатком [4].

Кожен елемент множини C може являти собою профіль користувача що включає різні характеристики, такі як вік, стать, прибуток, сімейний стан тощо. У найпростішому випадку профіль може містити лише один (унікальний) елемент, зазвичай це ID користувача.

Аналогічним чином, кожен елемент простору S може бути представлений як набором характеристик (наприклад, для колекції пісень це можуть бути: назва, жанр, виконавець тощо), так і лише своїм ідентифікатором.

2.2 Основна проблема рекомендаційних систем

Основна проблема рекомендаційних систем полягає в тому, що функція u, зазвичай, визначена не на всьому просторі $C \times S$, а лише на деякій його підмножині.

У зв'язку з цим виникає необхідність у екстраполяції функції u на весь простір $C \times S$ [4].

Від самого початку функція u визначена лише на елементах, яким користувач дав оцінку заздалегідь. Приклад матриці «користувач-елемент» для простого додатку, що рекомендує музичні групи, приведено у таблиці 2.1 (оцінки визначаються діапазоном від 1 до 10, null – позначає відсутність оцінки).

Таблиця 2.1 - «Матриця рейтингу для рекомендаційної системи»

	BMTH	Sam Smith	Justin Bieber	Evanescence
Користувач А	null	9	8	5
Користувач Б	10	1	3	4
Користувач В	2	10	null	6
Користувач Г	9	null	3	8

Таким чином, рекомендаційна система повинна бути спроможна оцінити (передбачити) оцінки для нерецензованих комбінацій «користувач-елемент» та підібрати рекомендації на основі цих прогнозів.

Екстраполяція від відомих до невідомих оцінок зазвичай здійснюється за рахунок:

а) евристичного визначення значень функції корисності та підтвердження його продуктивності експериментальним шляхом;

б) визначення функції корисності, яка оптимізує певні критерій продуктивності, як, наприклад, середня квадратична похибка;

Коли усі невідомі елементи отримують рейтинг, рекомендації для користувача визначаються шляхом вибору найвищого рейтингу серед усіх оцінок для цього користувача, відповідно до формули 2.1. Також ми можемо порекомендувати N кращих елементів для користувача або множині користувачів до відповідного елемента.

Слід зазначити, що процес генерації рекомендацій залежить від багатьох факторів, серед яких:

а) доступ до інформації про користувача (інтереси, рейтинги, місце розташування тощо);

б) механізм (алгоритм) фільтрування (колаборативна фільтрація, контентний, гібридний тощо);

в) використання методів для покращення результатів (Баєсові мережі, нечіткі моделі тощо);

г) структура та масштабованість бази даних;

д) продуктивність роботи системи;

е) аналіз якості результату за допомогою показників (точність, новизна тощо);

Нові рейтинги елементів, які ще не отримали оцінку, можуть бути отримані різними способами, використовуючи методи машинного навчання, теорії наближень або за допомогою евристичних методів [5].

2.3 Проблема «холодного запуску»

Під проблемою «холодного запуску» розуміють можливість генерації ненадійних рекомендацій через відсутність первинних рейтингів [4].

Розрізняють три типи цієї проблеми:

а) проблема нового користувача;

б) проблема нового матеріалу;

в) проблема нової спільноти;

Проблема нового користувача є досить суттєвою складністю при генерації індивідуальних рекомендацій. Оскільки ми не маємо рецензованих елементів для нового користувача, метод фільтрації вмісту на основі пам'яті (memory-based content filtering) стає неефективним у цьому випадку. Вирішення

цієї проблеми полягає в додаванні додаткової інформації щодо нового користувача (наприклад, вподобання).

Так само, проблема нового матеріалу виникає через додавання нових елементів у рекомендаційну систему. Оскільки нові елементи не мають початкових рейтингів, то вони залишаються непоміченими більшістю користувачів. Наявність групи мотивованих користувачів для оцінювання нових елементів системи допоможе вирішити цю ситуацію.

Проблема нової спільноти виникає при «запуску» системи через малу кількість або взагалі відсутність оцінок. Рекомендації на основі колаборативної фільтрації і заохочення користувачів при оцінюванні елементів системи допоможуть вирішити проблему нової спільноти.

2.4 Класифікація рекомендаційних систем

В залежності від механізму отримання рекомендацій системи розподіляють на ті, які застосовують [4]:

а) методи на основі змісту;

б) методи на основі колаборативної фільтрації;

в) гібридні методи;

2.4.1 Методи на основі змісту

В методах на основі вмісту корисність $u(c, s)$ елементу s для користувача c оцінюється на основі інших оцінок $u(c, s_i)$, що були визначені користувачем c для елементів $s_i \in S$, які «подібні» до елемента s.

Наприклад, у музичному додатку, щоб порекомендувати нові групи для прослуховування користувачу c, така система намагатиметься знайти спільні особливості серед груп, які користувач c високо оцінив у минулому (жанр, країна, мова тощо). Після цього лише групи з високим рівнем «схожості» до інтересів користувача потраплять до рекомендованих.

Підхід на основі змісту походить з досліджень в таких галузях, як пошук інформації та фільтрування інформації. Саме тому існує багато систем на основі змісту, які зосереджені на рекомендаціях елементів, що містять текстову інформацію (документи, новини, URL-адреси). Значним покращенням в цьому

підході стало використання так званих «профілей», які містять інформацію про смаки користувачів, вподобання та їх потреби. Інформація необхідна для формування профілю може бути отримана від користувачів явним шляхом, наприклад, через опитування, або неявно – через відстеження історії транзакцій в системі.

Отже, нехай *Content(s)* буде профілем елементу (фільму, пісні, туристичного місця тощо), що складається з, наприклад, набору атрибутів, які характеризують елемент s. Зазвичай, він складається шляхом визначення певних характерних особливостей елементу *s* (його змісту) та використовується для визначення ступеню відповідності предмета для подальших рекомендацій.

Оскільки, як згадувалося раніше, орієнтовані на зміст системи використовувались для рекомендації текстових елементів, саме зміст зазвичай описувався за допомогою ключових слів. Так, наприклад, такі системи можуть відображати для користувачів список веб-сторінок, які мають у своєму змісту 100 найбільш важливих або інформативних слів.

«Важливість» або «інформативність» слова k_j в документі d_j визначається завдяки певним ваговим показникам w_{ij}, які можуть задаватися різноманітними способами.

Одним із найбільш відомих показників для визначення ваги ключових слів є так званий «term frequency/inverse document frequency)» (TF-IDF) показник [6]. Він визначається наступним чином:

Припустимо, що N - загальна кількість документів, які можуть бути рекомендовані користувачам і що ключове слово k_j з'являється в n_i з них. Крім того, припустимо, що $f_{i,j}$ – це кількість появ ключового слова k_i в документі d_j. Тоді, $TF_{i,j}$, показник частоти (або нормалізованої частоти) ключового слова k_i в документі d_j, визначаємо так:

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}}, \tag{2.2}$$

Максимум обчислюється серед значень частот $f_{z,j}$ усіх ключових слів k_z, що з'являються в документі d_j. Однак, ключові слова, що містяться в багатьох документах не є корисними в процесі визначення релевантного та

нерелевантного документів. Виходячи з цього, на практиці дуже часто використовують показник зворотної частоти документа (IDF_i) в комбінації з простим показником частоти терміну ($TF_{i,j}$).

Зворотній показник частоти документа для ключового слова k_i в загальному вигляді визначається за формулою (2.3). В свою чергу визначення повної ваги ключового слова k_i для документа d_j за показником TF-IDF знаходиться як позначено у формулі (2.4).

$$IDF_i = \log \frac{N}{n_i}, \tag{2.3}$$

$$w_{i,j} = TF_{i,j} \times IDF_i, \tag{2.4}$$

У цьому випадку зміст документа d_j визначається таким чином [7]:

$$Content(d_j) = (w_{1j}, \dots, w_{kj}), \tag{2.5}$$

Як зазначалося вище, системи, що працюють з методами на основі змісту рекомендують предмети подібні до тих, які сподобались користувачу раніше. Зокрема, відбувається порівняння різних елементів системи, які попередньо отримали оцінку від користувача та рекомендуються найбільш відповідні до них.

Нехай *UserProfile(c)* буде позначати профіль користувача *c* та відображатиме його смаки та вподобання. Інформація щодо профіля може бути отримана шляхом аналізу змісту елементів, які були раніше переглянуті або оцінені користувачем та зазвичай будується з використанням методу аналізу ключових слів, що був описаний вище.

Наприклад, *UserProfile(c)* може бути заданий в якості вектору вагових коефіцієнтів $(w_{c1}, \dots, w_{ck})$, де кожен коефіцієнт w_{ci} визначає важливість ключового слова k_i для користувача *c* та може бути розрахований з вагових векторів елементів, які були оцінені індивідуально з використанням різних технік.

У цьому випадку функція корисності $u(c, s)$ визначається як [7]:

$$u(c, s) = f(UserProfile(c), Content(s), \tag{2.6}$$

Використовуючи вищенаведений принцип з області інформаційного пошуку, $UserProfile(c)$ для користувача c та $Content(s)$ для документа s можуть бути представлені як вектори TF-IDF $\overrightarrow{w_c}$ та $\overrightarrow{w_s}$ відповідно, що містять вагові коефіцієнти ключових слів. Більш того, функція корисності в цьому випадку представляється деякою евристичною оцінкою, яка визначається в термінах векторів $\overrightarrow{w_c}$ та $\overrightarrow{w_s}$, а саме косинусним коефіцієнтом або коефіцієнтом Отіаі (формула 2.7), де K - це загальна кількість ключових слів у системі.

$$u(c,s) = \cos(\overrightarrow{w_c}, \overrightarrow{w_s}) = \frac{\overrightarrow{w_c} * \overrightarrow{w_s}}{\|\overrightarrow{w_c}\|_2 * \|\overrightarrow{w_s}\|_2} = \frac{\sum_{i=1}^{K} w_{i,c} * w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2} * \sqrt{\sum_{i=1}^{K} w_{i,s}^2}}, \quad (2.7)$$

Наприклад, якщо користувач c читає багато статей за тематикою біоінформатики, то рекомендаційна система, яка працює на основі змісту елементів зможе порекомендувати іншу статтю присвячену біоінформатиці. Саме такий результат буде отримано тому, що ці статті будуть містити більше термінів з галузі біоінформатики ніж статті з других категорій, а $UserProfile(c)$, визначений вектором $\overrightarrow{w_c}$ буде містити ці терміни k_i з більш великими вагами w_{ic}. Відповідно, рекомендаційна система, яка використовує коефіцієнт Отіаі або подібний показник схожості визначить значення функції корисності $u(c,s)$ для статей s з високими вагами $\overrightarrow{w_s}$ за цими термінами як більш високе, ніж у усіх інших статей.

Слід зазначити, що робота методів заснованих на змісті обмежується особливим характеристиками об’єктів системи. Тому для достатнього набору таких характеристик, вміст об’єктів має бути або у формі, яка може легко та автоматично аналізуватися системою, або характеристики мають задаватися вручну. На практиці це може стати значною проблемою, особливо для специфічних галузей (наприклад, мультимедійні дані).

Ще однією проблемою може стати те, що два різні елементи з однаковим набором ключових слів вважаються ідентичними для такої системи.

До того ж, якщо система може рекомендувати лише ті елементи, які максимально узгоджуються з профілем користувача, то власне рекомендації, що він отримуватиме можуть містити лише елементи схожі з тими, які користувач оцінив високо. Наприклад, людина, яка не мала досвід у

прослуховуванні етнічної музики ніколи не отримає у рекомендаціях таку групу, навіть якщо вона є надзвичайно популярною у всьому світі і може прийти користувачу до вподоби. Тому гарно спроектована система має враховувати такі випадки та боротися із однорідністю рекомендованого контенту [6].

2.4.2 Методи на основі колаборативної фільтрації

На відміну від методів рекомендацій на основі вмісту, методи колаборативної фільтрації намагаються спрогнозувати цінність елементів для конкретного користувача на основі тих, що були попередньо оцінені іншими користувачами.

Тобто, оцінка $u(c, s)$ елементу s для користувача c оцінюється на основі інших оцінок $u(c_j, s)$, що були призначені елементу s такими користувачами $c_j \in C$, які за вподобаннями «подібні» до користувача c [8].

У тому ж музичному додатку (попередній приклад), щоб порекомендувати нові групи користувачу c, система на основі методів колаборативної фільтрації намагатиметься знайти «однодумців» користувача c (зі спільними інтересами та подібними оцінками). Після цього лише групи, які найбільш до вподоби «однодумцям» потраплять до рекомендованих.

Залежно від алгоритмів, які використовуються для колаборативної фільтрації, такі системи поділяються на два загальних класи:

а) Колаборативна фільтрація на основі сусідства (memory-based, neighborhood-based);

б) Колаборативна фільтрація на основі моделі (model-based);

Колаборативна фільтрація на основі сусідства - це перший метод, що використовується при колаборативній фільтрації. Також цей метод прийнято називати анамнестичним (memory-based), тому що він формує рекомендації на основі аналізу наявних у системі оцінок, що були зроблені певним користувачем та іншими користувачами, які мають аналогічні показники (вподобання) [8].

В свою чергу цей метод поділяється на два підтипи:

а) на базі схожості користувачів (user-based);

б) на базі схожості елементів (item-based);

В user-based методі значення невідомої оцінки $r_{c,s}$ для користувача c та елемента s зазвичай розраховується як сумарне значення оцінок інших (як правило, N найбільш схожих) користувачів для того ж елемента s [8]:

$$c' \in \hat{C},\ r_{c,s} = aggr(r_{c',s}), \tag{2.8}$$

де $\hat{C}$ – позначає множину N користувачів, які є найбільш схожими з користувачем c та мають оцінки елемента s (N може бути будь-яким числом в діапазоні від 1 до кількості усіх користувачів);

Функція агрегації може бути довільною. Деякі приклади функції приведені у нижче [8]:

$$r_{c,s} = \frac{1}{N}\sum_{c' \in \hat{C}} r_{c',s}, \tag{2.9}$$

$$r_{c,s} = k\sum_{c' \in \hat{C}} sim(c,c') \times r_{c',s}, \tag{2.10}$$

$$r_{c,s} = \overline{r_c} + k\sum_{c' \in \hat{C}} sim(c,c') \times (r_{c',s} - \overline{r_c}), \tag{2.11}$$

де k – спеціальний множник, який нормалізує значення та розраховується за нижчезазначеною формулою (2.12);

$\overline{r_c}$ – середній рейтинг користувача c, що знаходимо за формулою (2.13);

$$k = \frac{1}{\sum_{c' \in \hat{C}} |sim(c,c')|}, \tag{2.12}$$

$$\overline{r_c} = \left(\frac{1}{|S_c|}\right)\sum_{s \in S_c} r_{c,s}, \tag{2.13}$$

де S_c – множина елементів, які мають оцінку від користувача c;

В найпростішому випадку, функція агрегації може бути представлена середнім арифметичним оцінок інших користувачів (див. форм. 2.9). Однак, зазвичай використовують зі зваженою сумою, відповідно до 2.10. Значення схожості між користувачами c та c′ $sim(c,c')$ є, по суті, значенням відстані та

використовується у якості ваги. Чим більше схожі користувачі, тим більшу вагу матиме рейтинг $r_{c',s}$ при передбаченні рейтингу $r_{c,s}$. Слід зазначити, що $sim(x, y)$ є евристичним показником, який вводиться для того, щоб мати змогу диференціювати рівні схожості користувачів та в водночас спростити процедуру оцінки рейтингу. Рекомендаційні системи можуть використовувати власні показники схожості, якщо розрахунки нормалізуються коефіцієнтом k, згідно з формулою 2.12.

Одна з проблем, яка виникає при використанні формули 2.10 полягає в тому, що вона не враховує той факт, що різні користувачі можуть розцінювати шкалу рейтингу по-різному.

Тому, для вирішення цього обмеження використовують скориговану вагову суму за формулою (2.11).

У цьому підході, замість абсолютних значень оцінок використовують зважену суму, яка враховує відхилення від середнього рейтингу відповідного користувача.

Обчислення ступеню схожості між користувачами, як правило, базується на оцінках об’єктів, які обидва користувача обрали.

Отже, введемо множину S_{xy}, що представляє набір елементів, які були оцінені обома користувачами x та y, таким чином:

$$S_{xy} = \{ s \in S \mid r_{x,s} \neq \emptyset \,\&\, r_{y,s} \neq \emptyset \}, \tag{2.14}$$

Зазвичай, на практиці, використовують наступні метрики [8]:

а) евклідова відстань;

б) коефіцієнт кореляції Пірсона;

в) коефіцієнт Жаккара;

Евклідова відстань – це геометрична відстань між двома точками в багатомірному просторі, яка обчислюється наступним чином:

$$sim(x, y) = \sqrt{\sum_{s \in S_{xy}} (r_{x,s} - r_{y,s})^2}, \tag{2.15}$$

Також може використовуватися квадрат евклідової відстані для того, щоб отримати більш широкий діапазон ваг схожості.

Коефіцієнт кореляції Пірсона – є більш точним методом визначення лінійної залежності між двома величинами. Обчислення схожості за коефіцієнтом Пірсона [13]:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}}(r_{x,s} - \overline{r_x})(r_{y,s} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}}(r_{x,s} - \overline{r_x})^2 \sum_{s \in S_{xy}}(r_{y,s} - \overline{r_y})^2}}, \quad (2.16)$$

Для скінченних множин, кожен елемент яких є невід' ємний Полем Жаккаром в 1901 році була запропонована наступна міра схожості:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} min(r_{x,s,}\, r_{y,s})}{(\sum_{s \in S_{xy}}(r_{x,s}) + \sum_{s \in S_{xy}}(r_{y,s}) - \sum_{s \in S_{xy}} min(r_{x,s,}\, r_{y,s}))}, \quad (2.17)$$

Хоча для обчислення ступеню схожості можуть використовуватися різні підходи, існує одна загальна стратегія при побудові таких систем. Вона полягає у тому, що бажано заздалегідь проводити обчислення показників схожості усіх користувачів $sim(x,y)$ (включно із множиною S_{xy}) та перераховувати їх лише за потребою [13]. Тоді при запиті користувача рекомендованих елементів невідомі рейтинги можуть бути обчислені більш ефективно, спираючись на обчислені попередньо показники схожості.

Item-based підхід, в свою чергу, полягає у порівняння схожості елементів системи. Алгоритм цього методу є аналогічним з user-based, за виключенням того, що розглядаються не користувачі та їх схожості, а самі об’єкти. На практиці було доказано, що item-based алгоритм не тільки забезпечує більшу ефективність при обчисленнях, але й дає таку ж або навіть кращу якість та точність у роботі [8].

Ідея колаборативної фільтрації на основі моделі полягає в тому, що можливо побудувати певну модель за сукупністю наявних у системі оцінок. На основі цієї моделі й будуть обчислюватися передбачення та рекомендації у системі

Модель може бути побудована за допомогою наступних методів [8]:

а) модель Байєса;

б) методи кластерного аналізу;

в) методи латентного семантичного аналізу (LSA);

г) сингулярне розкладення (SVD);

2.4.3 Гібридні методи

Деякі системи рекомендацій використовують гібридний підхід, поєднуючи методи на основі змісту та методи колаборативної фільтрації.

Такий підхід допомагає уникнути певних обмежень, які властиві цим методам, якщо вони використовуються окремо.

Різні способи поєднання методів на основі змісту та методів колаборативної фільтрації у гібридну систему рекомендацій класифікуються наступним чином:

а) реалізація методів на основі змісту та методів колаборативної фільтрації окремо і поєднання їх передбачень;

б) включення деяких можливостей методів на основі змісту до методів колаборативної фільтрації;

в) включення деяких можливостей методів колаборативної фільтрації до методів на основі змісту;

г) побудова загальної об'єднуючої моделі, на базі обох методів;

Розглянемо використання об’єднучої моделі. Оскільки усі вищезазначені методи прогнозують значення рейтингів, поєднання різних джерел інформації може значно покращити результати та виправити індивідуальні недоліки кожного з методів. Для об’єднання результатів прогнозування, зазвичай, використовують просту зважену суму усіх обчислених рейтингів [5]:

$$r_{hybrid} = w_{cf} * r_{cf} + w_{cbf} * r_{cbf}, \tag{2.18}$$

$$w_{cf} + w_{cbf} = 1, \tag{2.19}$$

де cf – індекс для колаборативної фільтрації;

cbf – індекс для фільтрації на основі змісту;

Наведені вагові коефіцієнти не є статичними, їх значення залежить від доступності даних у системі. Якщо отриманої інформації достатньо для роботи обох підходів, то їх внесок в гібридні рейтинги буде рівним ($w_{cf} + w_{cbf} = 0{,}5$).

Якщо ж, наприклад, для колаборативної фільтрації було знайдено менш ніж 5 «сусідів», то w_{cf} може бути меншим за 0,5 або навіть дорівнювати нулю.

2.5 Аналіз методів машинного навчання для рекомендаційних систем

До терміну Deep Learning (DL), як правило, відносять підмножину дисципліни машинного навчання. Характерна суть глибокого навчання полягає в тому, що вона вивчає різні рівні відображень та абстрактних форм даних.

З практичних причин будь-яку нейронну диференційовану архітектуру розглядають як «глибоке навчання», оскільки вона оптимізує диференційовану цільову функцію, використовуючи варіант стохастичного градієнтного спуску (SGD) [9]. Серед найбільш популярних та ефективних архітектурних парадигм виділяють:

- багатошаровий перцептрон (MLP) — це нейронна мережа, що складається з декількох прихованих шарів між вхідним та вихідним шаром. MLP можна інтерпретувати як масив шарів, що виконують нелінійні перетворення, вивчаючи ієрархічні особливості відображень [12]

- автокодувальник (AE) — це модель нейронних мереж, яка використовує навчання без вчителя та реконструює вхідні дані у вихідному шарі. Метою автокодувальника є навчитися представленню (кодуванню) набору даних, зазвичай задля зниження розмірності [15, 16].

- згорткові нейронні мережі (CNN) — клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовується до аналізу візуальних зображень [11, 13]. CNN використовують відносно мало попередньої обробки, що робить їх більш ефективними в порівнянні з алгоритмами класифікації зображень.

- рекурентні нейронні мережі (RNN) — клас штучних нейронних мереж, у якого поєднання між вузлами утворюють орієнтований граф у часі. Це створює внутрішній стан мережі, що дозволяє їй проявляти динамічну поведінку в часі [14]. На відміну від нейронних мереж прямого поширення,

РНМ можуть використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів.

Галузь рекомендаційних стрімко розвивається в останні роки, тому застосування складних та різноманітних архітектурних рішень, що використовують нейронні мережі стає звичним підходом. Серед переваг рекомендаційних моделей, побудованих з використанням технік DL, можемо виділити наступні:

- нелінійні трансформації. На відміну від лінійних моделей, нейронні мережі здатні моделювати нелінійність даних. Ця властивість дозволяє виявити складні та неявні особливості взаємодій користувача [10]. Звичайні методи, такі як матрична факторизація, машина факторизації, розріджена лінійна модель, по суті є лінійними моделями;

- навчання представлень. Нейронні мережі є надзвичай ефективними при виявленні неявних та корисних представлень у вхідних даних. Використовуючи цей підхід до великої кількості інформації, щодо користувачів та товарів, яку зберігають сучасні ресурси, можна досягти досить точних результатів в рекомендаціях. Це дозволяє покращувати рекомендаційні моделі завдяки інформації, отриманій з тексту, зображень, аудіо та відео.

- моделювання послідовності. Нейронні мережі демонструють вражаючі результати у задачах обробки послідовних даних, таких як машинний переклад, розпізнавання мови. Моделювання послідовних сигналів є важливою темою для розрахування тимчасової динаміки поведінки користувачів та еволюції елементів [10]. Прикладом застосування цього підходу є прогнозування наступного товару у електронному кошику.

- гнучкість. Методики Deep Learning є дуже універсальними та гнучкими, особливо з появою спеціалізованих фреймворків, таких як TensorFlow, Keras, PyTorch тощо. Завдяки модульності цих інструментів, ми можемо, наприклад, комбінувати різні підходи, створюючи гібридні та більш ефективні рекомендаційні моделі.

2.5.1 Метод SVD

Методи, що використовують колаборативну фільтрацію на основі моделі зазвичай засновані на матричній факторизації (MF). Методи MF використовуються як спосіб декомпозиції змінних и зменшення розмірності матриць [16]. Такі методи спроможні вирішувати проблеми масштабованості та розрідженості краще, аніж ті, що засновані на пам'яті.

Мета MF полягає в тому, щоби дізнатися приховані вподобання користувача та атрибути (властивості) елементів системи із відомих рейтингів. А потім використати отримані значення для прогнозування невідомих оцінок через добуток розкладених матриць.

Одним із типів розкладення матриць є сингулярне розкладення (SVD). Сингулярне розкладання є досить зручним методом при роботі з матрицями. SVD дозволяє виділити геометричну структуру матриці та допомагає наочно представити наявні дані. Сингулярне розкладання використовується при вирішенні багатьох типів задач, таких як наближення методом найменших квадратів та рішення систем рівнянь до стиснення й розпізнавання зображень.

В загальному вигляді модель SVD описується так:

Нехай задана матриця X розмірності $(m * n)$ така, що

$$X = USV^T, \tag{2.20}$$

де U – ортогональна матриця $(m * r)$,

S – діагональна матриця $(r * r)$,

V^T – ортогональна матриця $(r * n)$.

В даному прикладі матриця U представляє вектор особливостей користувачів, V представляє вектор особливостей елементів, діагональна матриця S містить упорядковані елементи:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k, \tag{2.21}$$

Отже виходить, що ми представляємо кожного користувача вектором з r-факторів u_i, а кожен елемент вектором факторів - v_j. Далі для прогнозування рейтингу користувача i елемента j беремо їх скалярний добуток. Таким чином вектор факторів u_i визначає наскільки користувачу до вподоби, а вектор факторів v_j – наскільки цей фактор присутній у кожному з елементів

2.5.2 Метод стохастичного градієнтного спуску

Більшість базових реалізацій рекомендаційних систем засновані на застосуванні memory-based підходів. Хоча вони й дають гарні результати, проте погано масштабуються та схильні до проблеми «холодного» старту. З іншого ж боку застосування алгоритму SVD може бути дуже ресурсномістким, особливо для дуже розріджених матриць [19].

Найбільш актуальним підходом до рішення цієї проблеми є застосування мінімізації середньоквадратичної похибки відповідно для матриці користувача U, так і елементів – V:

$$\min_{Q^*,P^*} \sum_{(u,i)\in K}(r_{ui} - P_u^T Q_i)^2 + \lambda(\|Q_i\|^2 + \|P_u\|^2), \quad (2.22)$$

де K – множина пар (u, i),

r_{ui}– рейтинг елемента i для користувача u,

λ – нормалізуючий фактор.

Навчання нашої моделі полягає у мінімізації середньоквадратичної похибки. Для цього ми можемо застосувати метод стохастичного градієнтного спуску (SGD).

Стохастичний градієнтний спуск – це оптимізаційний алгоритм, який відрізняється від звичайного градієнтного спуску тим, що градієнт функції, що оптимізується вираховується на кожному кроці не як сума градієнтів від кожного елемента вибірки, а як градієнт від одного, випадково обраного елемента [18].

Алгоритм SGD базується на оптимізаційному методі градієнтного спуску [17]. Основна ідея якого полягає в тому, щоб здійснити оптимізацію в напрямку найшвидшого спуску, тобто у напрямку антиградієнта $-\nabla f$:

$$x^{k+1} = x^k - \lambda^k \nabla f(x^k), \quad (2.23)$$

де λ^k- є кроком спуску та вибирається або константою, або є дрібним (ділиться на певну величину).

Зупинки ітераційного алгоритму буде визначатись досягненням певної заданої точності. Проблемою цього алгоритму є необхідність обчислювати градієнт від кожного елемента вибірки, що може значно сповільнити алгоритм. Ідея SGD полягає в використання лише одного елемента для перерахування нового наближення ваг (значень). Тобто тепер нове наближення будемо знаходити як:

$$x^{k+1} = x^k - \lambda^k \nabla f_i(x^k), \quad (2.24)$$

де i – випадково обраний індекс.

2.6 Оцінка точності роботи рекомендаційної системи

Важливим пунктом при розробці рекомендаційних систем є оцінювання якості роботи таких систем, зокрема точності рекомендацій. В цьому розділі наведемо деякі загальноприйняті метрики оцінювання.

Слід зазначити, що деякі з метрик краще працюють із системами, які прогнозують рейтинг, а деякі – з тими, що дають рекомендації типу «Top N».

Для оцінювання точності використовують наступні показники [8]:

а) mean absolute error (MAE):

$$MAE = \frac{\sum_{c \in C} \sum_{s \in S_{test}} |rec(c,s) - r_{c,s}|}{\sum_{c \in C} |S_{test}|}, \quad (2.25)$$

де C – множина користувачів;

S_{test} – тестова вибірка елементів;

$rec(c,s)$ – рейтинг, який був визначений системою;

$r_{c,s}$ – рейтинг, який визначив користувач c для елемента s;

б) root mean square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{c \in C} \sum_{s \in S_{test}} (rec(c,s) - r_{c,s})^2}{\sum_{c \in C} |S_{test}|}}, \qquad (2.26)$$

в) нормалізовані MAE, RMSE. Значення MAE та RMSE нормалізуються поділом цих показників на різницю між мінімальним та максимальним рейтингом у виборці;

г) усереднені MAE, RMSE. Якщо вибірка нерівномірна за елементами або користувачами, то можна розрахувати MAE/RMSE для кожного елемента (користувача) окремо, а потів взяти середнє від отриманих значень [8];

2.7 Розробка алгоритму гібридної рекомендаційної системи

В ході дослідження предметної області та підходів до побудови рекомендаційних систем були проаналізовані та розглянуті методи колаборативної фільтрації, контентної фільтрації та гібридні методи.

Отже, перевагами методів на основі змісту є:

а) відсутність необхідності у відгуках інших користувачів;

б) орієнтованість на персональні вподобання користувача;

в) спроможність рекомендувати нові або непопулярні елементи;

г) можливість пояснити вибір користувачу;

Серед недоліків цих методів виділяємо наступні:

а) складність у знаходженні доречних «ключових особливостей», особливо для предметної області, що пов’язана із туризмом;

б) орієнтованість на персональні вподобання призводить до того, що об’єкти, які не входять до колу інтересів користувача майже неможливо отримати у якості рекомендації, тому виникає проблема одноманітності.

В свою чергу методи колаборативної фільтрації дають можливість рекомендувати об’єкти, про які користувач може не здогадуватися та які мають досить високу ймовірність йому сподобатись. Також реалізація колаборативної фільтрації простіша та не дуже сильно залежить від особливостей предметної області, до якої вона застосовується.

Колаборативна фільтрація, на відміну від content-based підходу, напряму залежить від наявності у системі достатньої кількості інформації. Адже розрідженість матриці рейтингів або відсутність схожих користувачів

(елементів) призводять до значного спаду ефективності такої системи (проблема «холодного старту»).

Враховуючи усі вказані особливості, для реалізації рекомендаційної системи нашого сервісу було обрано гібридний підхід. Для побудови гібридної моделі оцінювання була обрана модель з ваговими коефіцієнтами (за формулою 2.18). При цьому підході результати обох методів фільтрації (контентна та колаборативна) поєднуються за допомоги зваженої суми.

Якщо даних для роботи методів достатньо, то вагові коефіцієнти є рівними та дорівнюють 0,5. Проте коефіцієнти можуть змінюватись динамічно, залежно від, наприклад, розрідженості даних.

2.7.1 Застосування алгоритму фільтрації на основі змісту

Для реалізації контентної фільтрації необхідно визначити спеціальні теги, що характеризують найбільш популярні музичні жанри. Для більш наглядної та реалістичної роботи системи було вирішено скористатися публічними вибірками даних, які надає сервіс last.fm. Серед існуючих колекцій даних є вибірка – tags.dat, яка містить список найбільш популярних музичних жанрів та напрямів за якими ми і будемо калібрувати профілі користувачів.

Повний список містить досить велику кількість унікальних музичних жанрів, приклад деяких з них наведений у таблиці 2.2. Для полегшення роботи системи кількість жанрів було скорочено до фіксованого значення. Ці жанрові теги визначають профілі як артистів, так і користувачів системи.

В системі профілі задаються у вигляді багатомірних векторів, кожен компонент яких визначає ступінь належності до певної «особливості».

Таблиця 2.2 – Список визначених жанрів

Metal	Alt. metal	Death metal
Pop	Lounge	Jazz
Dance	Post-rock	Industrial
Folk	Hip-hop	Rap
Rock	Grunge	Hard-rock
Classic	Techno	Rnb
Indie	Electro	Reggae

Вектор профілю місць задається у бінарному форматі та характеризує належність або відсутність належності цього артиста до певного жанру.

Вектор профілю користувача в свою чергу представлено у вигляді нормалізованих значень, які формуються на базі персональних вподобань, що вказує сам користувач та середнього значення векторів профілей артистів, яким користувач давав оцінку:

$$w_{cj} = 0.5 * (\frac{\sum_{s \in s_{sj}} (r_{cs} - \overline{r_c})}{|s_{sj}|} + (pref_{cj} - \overline{r_c})), \ldots \quad (2.27)$$

де, w_{cj} – значення компонента j у векторі профілю користувача;

$\overline{r_c}$ – середній рейтинг користувача c (формула 2.13);

$pref_{cj}$ – значення компонента j у векторі вподобань користувача;

S_{sj} – множина елементів, які оцінив користувач с та які містять у профілі компонент j;

Сформувавши профіль користувача ми можемо виконувати прогнозування рейтингів для артистів. Для цього скористаємось коефіцієнтом Отіаі (за формулою 2.7).

Для отримання прогнозованої оцінки скористаємось формулою:

$$r_{c,s} = (R_{\max} / 2) + (u(c,s) * R_{\max} / 2), \quad (2.28)$$

де $R_{\max}$ – максимальне можливе значення оцінки;

$u(c,s)$ – коефіцієнтом Отіаі для елемента s;

Наведемо приклад розрахунків. Нехай для формування профілей артистів було визначено 3 жанри: rock / pop / hip-hop, а середній рейтинг для користувача – 3 (табл. 2.3).

В такому випадку розрахунки профілю користувача матимуть наступний вигляд (табл. 2.4).

Таблиця 2.3 – Нормалізація оцінок

	Артист А	Артист Б	Артист В	Артист Г
Профілі	1 0 1	0 1 1	0 1 0	1 1 0
Оцінка	4	3	1	4

Норм. оцінка	(4 – 3) = 1	(3 – 3) = 0	(1 – 3) = -2	(4 – 3) = 1

Таблиця 2.4 – Розрахунок профілю

	Rock	Pop	Hip-hop
За оцінками	(1 + 1) / 2 = 1	(0 – 2 + 1) / 3 = -0.333	(1 + 0) / 2 = 0.5
Норм. вподобання	(4.6 – 3) = 1.6	(1 – 3) = -2	(3 – 3) = 0
Загалом	(1 + 1.6) / 2 = 1.3	(-2 - 0.333) / 2 = -1.16665	(0.5 + 0) / 2 = 0.25

Згідно зі сформованим профілем спрогнозуємо оцінку для «Артисту Д», що має вектор «010».

$$u(c,s) = \frac{(1.3*0+1*-1.16665+0*0.25)}{1.76*1} = -0.6628.$$

$$r_{c,s} = 2.5 + (2.5 * -0.6628) = 0.84.$$

Тобто, «Артист Д», який належить виключно до жанру pop, отримає прогнозовану оцінку 0.84. Результат добре відображає сформований профіль, адже наш користувач надає перевагу жанрам rock, hip-hop, а до pop-жанру відноситься доволі погано.

2.7.2 Застосування алгоритму колаборативної фільтрації

Для реалізації колаборативного фільтрації було обрано model-based підхід, який передбачає побудування динамічної моделі за допомогою алгоритмів ML.

Проаналізувавши існуючі методи машинного навчання вибір було зроблено на застосування алгоритмів сингулярного розкладання та стохастичного градієнтного спуску. Обрані алгоритми є досить простими в реалізації, проте дають дуже точні результати.

Отже рекомендаційна модель буде реалізовано за допомогою алгоритму SVD за допомогою мови програмування Python із застосуванням алгоритму GSD для тренування моделі.

Для більш наглядних результатів роботи будемо використовувати тестові вибірки з початковими даними щодо виконавців, користувачів та жанрів, які надає сервіс Last.fm.

2.8.Загальний алгоритм гібридної моделі

Для об’єднання результатів обох методів фільтрування, використовуємо формулу 2.18.

Коефіцієнти w_{cf}, w_{cbf} будемо вважати рівними, згідно з формулою 2.19.

На основі проаналізованих алгоритмів та способів їх застосування було розроблено блок-схему алгоритму (рис. 2.2), який описує процес генерації рекомендацій в загальному вигляді.

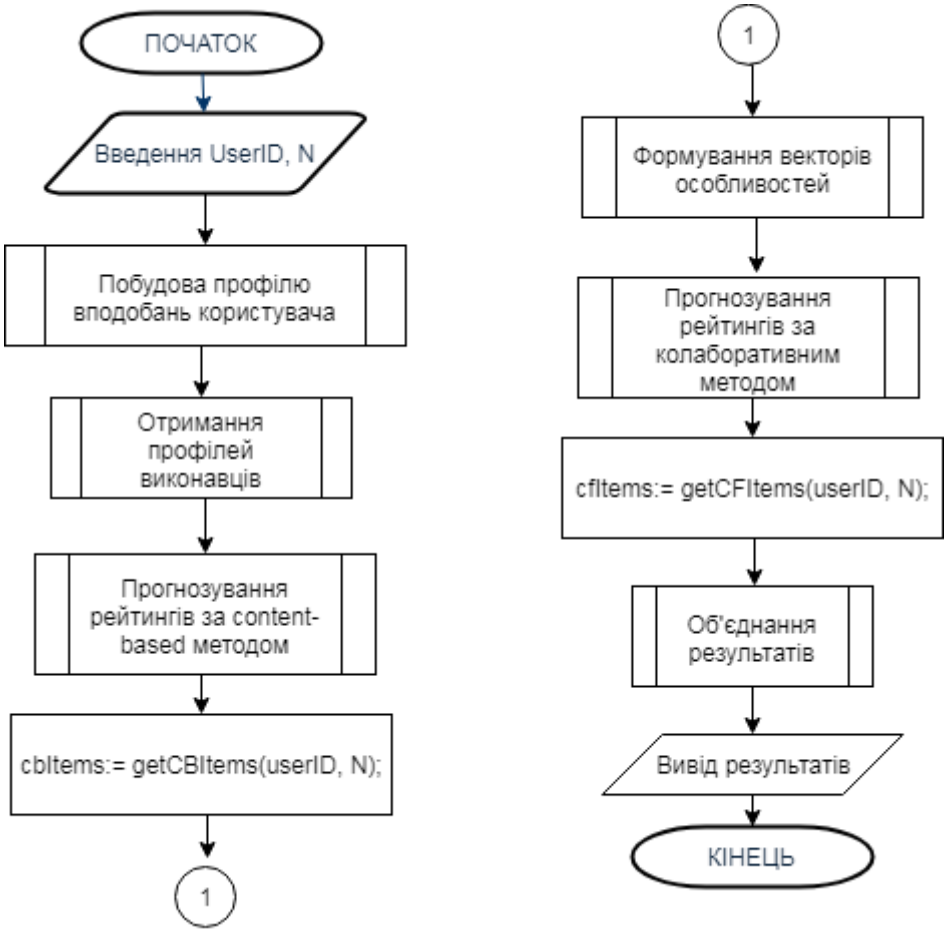

Рисунок 2.2 – Блок-схема гібридного алгоритму системи

3 ПРОЕКТУВАННЯ ГІБРИДНОГО РЕКОМЕНДАЦІЙНОГО ВЕБ-СЕРВІСУ

3.1 UML-проектування

UML (англ. Unified Modeling Language - уніфікована мова моделювання) – це мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

UML є мовою широкої профілю, та являє собою відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яку також називають UML-моделлю.

UML був створений для більш зручного способу визначення, візуалізації, проектування та документування, в основному, програмних систем. UML не є мовою програмування, але на підставі UML-моделей можливо здійснити генерацію коду [20].

3.1.1 Діаграма варіантів використання

Діаграма прецедентів (діаграма варіантів використання) в UML стандарті – це діаграма, що відображає відносини між акторами та прецедентами і є складовою частиною моделі прецедентів, яка в свою чергу дозволяє описувати систему на концептуальному рівні [21].

Під акторами у діаграмі прецедентів розуміють певний набір ролей користувача (розуміється в широкому сенсі: людина, зовнішня сутність, клас, інша система), що взаємодіє з деякою сутністю (системою, підсистемою, класом) [20]. Актори не можуть бути пов'язані один з одним, крім випадків позначення відносин узагальнення / успадкування. На діаграмах актори відображаються стилізованими чоловічками.

Прецедент позначає дії, які можуть виконуватися у системі та призводять до спостережуваних акторами результатів, позначається у вигляді еліпса з написом.

Основним призначенням діаграми прецедентів є опис функціональності та поведінки, що дозволяє замовнику, кінцевому користувачеві та розробнику спільно обговорювати систему, яка вже існує або потребує проектування [23].

Отже, для більш наглядного розуміння принципів роботи веб-сервісу в цілому, а також його основних функціональних можливостей була спроектована діаграма варіантів використання (рисунок 3.1).

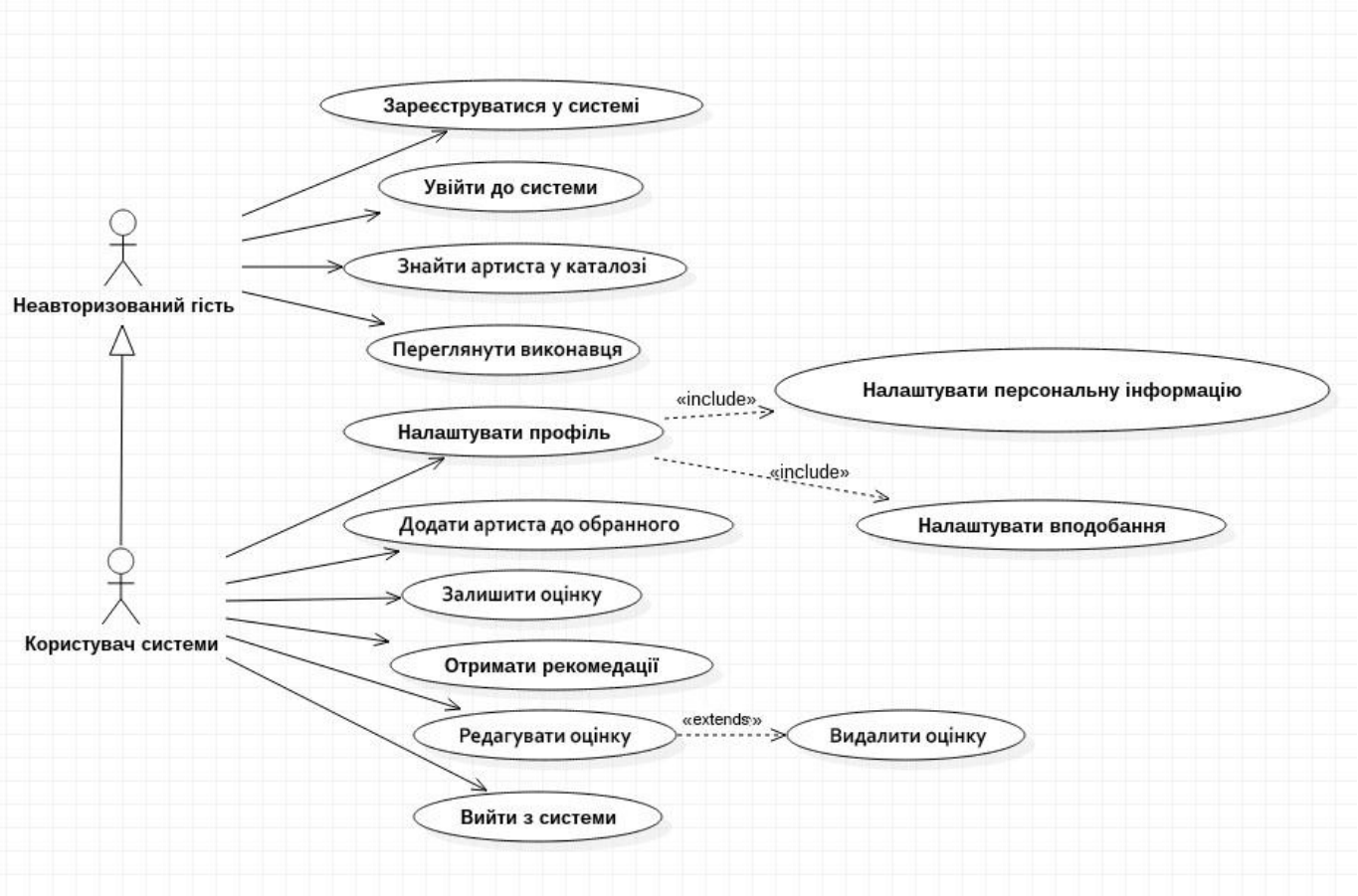

Рисунок 3.1 – Діаграма прецедентів

Основні актори зазначені на діаграмі:

а) Неавторизований гість;

б) Користувач системи (успадкований від «неавторизованого гостя»);

Для усіх користувачів системи було виділено такі прецеденти:

а) Зареєструватися у системи;

б) Увійти до системи;

в) Знайти артиста у каталозі;

г) Переглянути виконавця;

У свою чергу для авторизованих користувачів серед функціональних можливостей виділяємо:

а) Налаштувати профіль;

б) Налаштувати персональну інформацію (включається до «Налаштувати профіль»);

в) Налаштувати вподобання (включається до «Налаштувати профіль»);

г) Додати артиста до обраного;

д) Залишити оцінку;

е) Отримати рекомендації;

є) Редагувати оцінку;

ж) Видалити оцінку (включається до «Редагувати відгук»);

з) Вийти з системи;

3.1.2 Діаграми послідовності та комунікації

Діаграма послідовності – діаграма, що для певного набору об’єктів системи на єдиній часової осі відображає життєвий цикл якогось іншого об’єкта (створення, діяльність, знищення) та взаємодію акторів в межах певного прецеденту [20].

За допомогою діаграми прецедентів, як вже було зазначено вище, виявляють основі групи користувачів системи та задачі, які ця система має вирішувати.

В свою чергу, за допомогою діаграми послідовності ми надаємо опис послідовності дій, які необхідні для досягнення (виконання) поставленої мети конкретного прецеденту.

Основними елементами діаграми послідовності є:

а) позначення самих об’єктів (прямокутники з назвами об’єктів);

б) вертикальні «лінії життя», що відображають плин часу в процесі виконання прецеденту;

в) прямокутники (на «лінії життя»), які відображають діяльність об’єкта або виконання ним певних функцій;

г) стрілки, що позначають обмін сигналами або повідомленнями між об’єктами;

Діаграма комунікації – діаграма, на якій зображуються взаємодії та обмін повідомленнями між частинами системи, об’єктами або ролями [20].

На відміну від діаграми послідовності, на діаграмі комунікацій явно вказуються відносини між об’єктами, а час як окремий вимір не використовується.

Це дозволяє дещо спростити опис прецеденту та змістити фокус уваги на взаємовідносини об’єктів. Для підтримки порядку повідомлень їх хронологічно нумерують.

В процесі проектування сервісу було реалізовано діаграми послідовності та комунікацій для головних прецедентів системи:

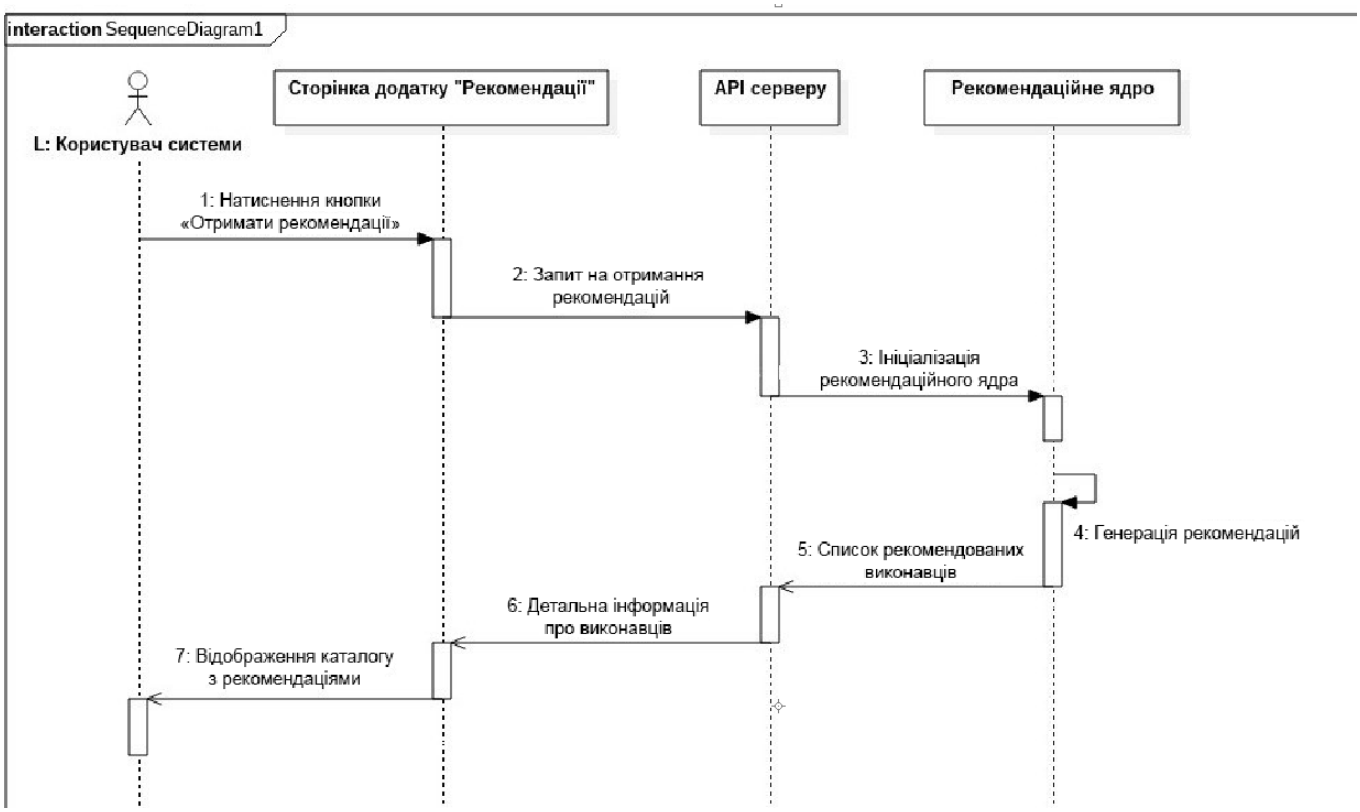

Рисунок 3.2 – Діаграма послідовності для прецеденту «Отримати рекомендації»

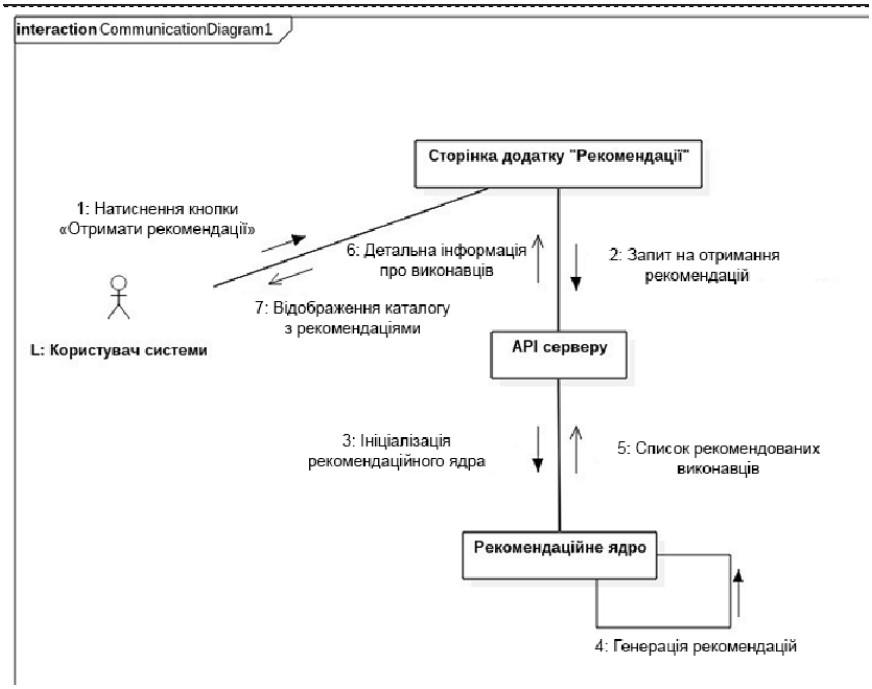

Рисунок 3.3 – Діаграма комунікацій для прецеденту «Отримати рекомендації»

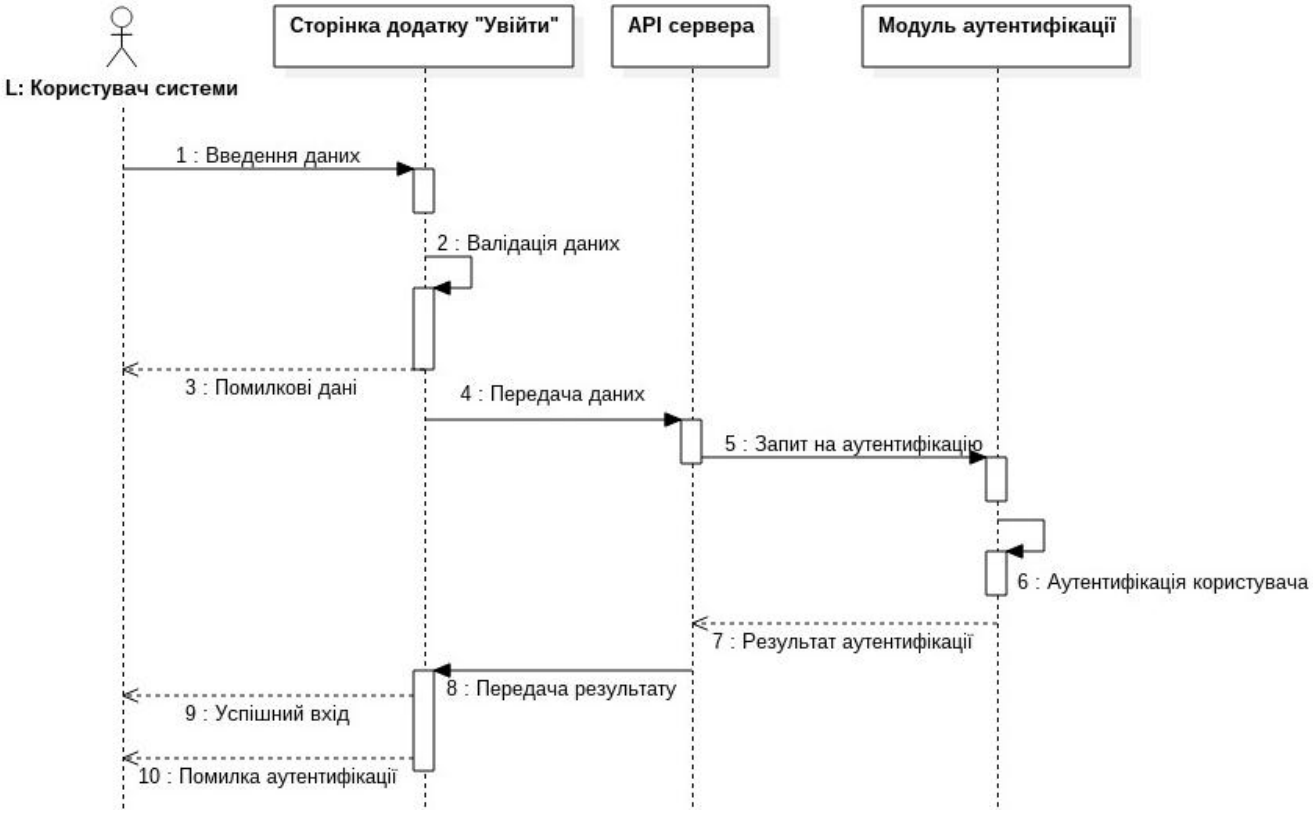

Рисунок 3.4 – Діаграма послідовності для прецеденту «Увійти до системи»

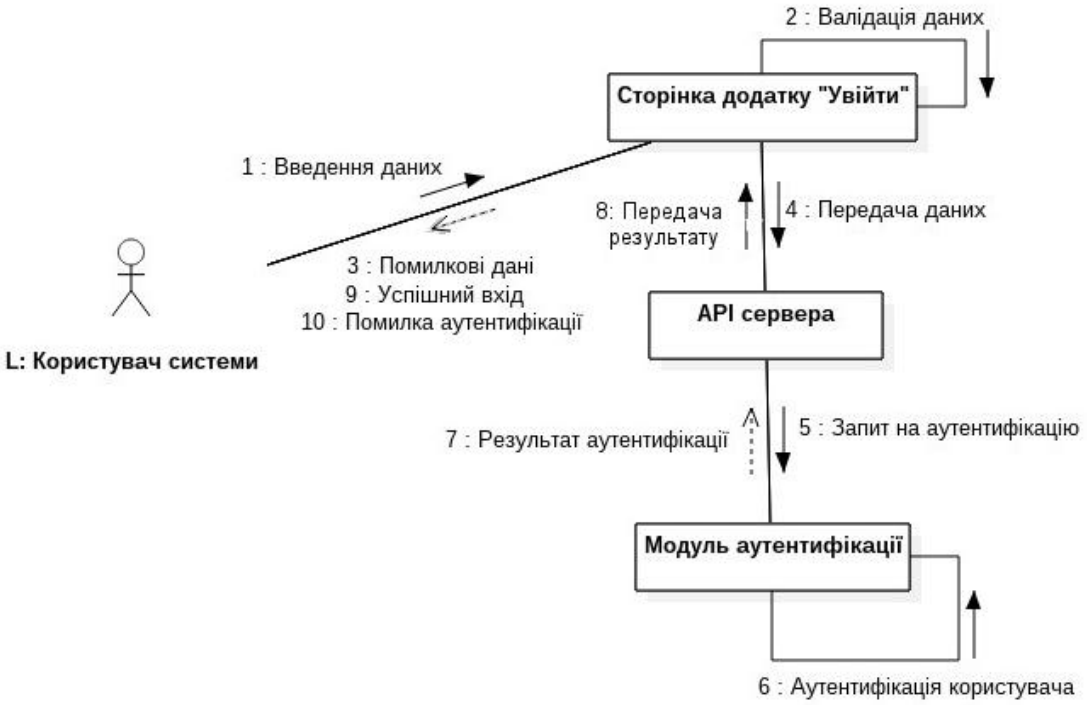

Рисунок 3.5 – Діаграма комунікацій для прецеденту «Увійти до системи»

Рисунок 3.6 – Діаграма послідовності для прецеденту «Знайти виконавця у каталозі»

Рисунок 3.7 – Діаграма комунікацій для прецеденту «Знайти виконавця у каталозі»

3.2 Проектування бази даних

Під проектування бази даних розуміють процес створення схеми бази даних і визначення необхідних обмежень цілісності.

Серед основних цілей проектування можна виділити наступні:

– забезпечення зберігання в БД всієї необхідної інформації;

– забезпечення можливості отримання даних по всім необхідним запитам;

– скорочення надмірності і дублювання даних;

– забезпечення цілісності бази даних;

Зазвичай проектування бази даних здійснюється в три етапи:

а) концептуальне проектування;

б) даталогічне проектування;

в) фізичне проектування;

Концептуальним проектування прийнято називати побудову семантичної моделі предметної області, тобто інформаційної моделі, яка представляю найбільш високий рівень абстракції.

Через це, така модель створюється без орієнтації на якусь конкретну СУБД або модель даних, її призначення – описати модель даних, опираючись на бачення користувачем предметної області.

Конкретний вид та зміст концептуальної моделі бази даних визначається обраним для цього формальним апаратом.

Зазвичай використовуються відповідні графічні нотації. Одним із найпоширеніших засобів опису концептуальних схем предметної області є модель даних "сутність-зв'язок» (ER-модель).

ER-модель використовується при проектуванні з достатньо високим рівнем абстракції та дозволяє виділити ключові сутності системи та позначити зв’язки, які можуть бути встановлені між цими сутностями [22].

Як правило концептуальна модель бази даних містить опис інформаційних об’єктів або понять предметної області, а також може включати опис вимоги до припустимих значень даних та зв’язків між ними.

В ході проектування веб-сервісу було розроблено ER-діаграму (рис 3.8), що відображає основні сутності, які приймають участь у роботі системи.

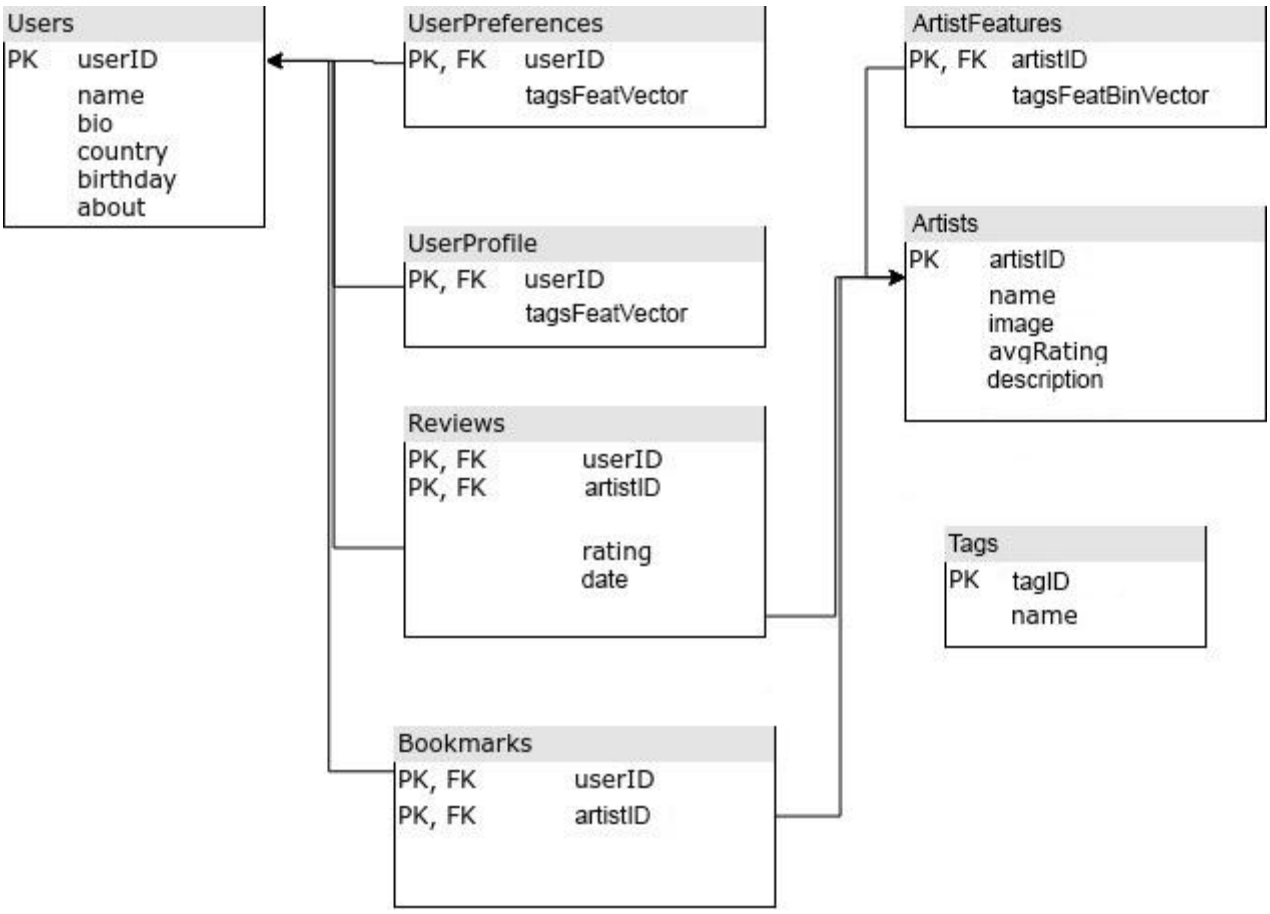

Рисунок 3.8 – ER-діаграма бази даних веб-сервісу

Спроектована діаграма містить наступні сутності системи:

а) «Users» – містить інформацію про зареєстрованих користувачів додатку. Сутність ідентифікується первинним ключем «userID»;

б) «UserPreferences» – містить інформацію щодо вподобань користувача, яку він може скорегувати через інтерфейс. Сутність пов’язана з таблицею «Users» через зовнішній ключ «userID», який також служить і первинним ключем;

в) «UserProfile» – містить дані профіля користувача у вигляді вектору вподобань, який формується на основі вподобань користувача та оцінок артистів. Сутність пов’язана з таблицею «Users» через зовнішній ключ «userID», який також служить і первинним ключем;

г) «Artists» – містить інформацію про музичних виконавців. Первинний ключ заданий полем «artistID» ;

д) «Tags» – містить список спеціальних «тегів», на базі яких формуються профілі користувача та артистів. Ідентифікується первинним ключом – «tagID»;

е) «ArtistFeatures» – ця таблиця містить профілі артистів, які представлені у вигляді бінарного вектору. Первинний ключ «artistID» також виступає у якості зовнішнього, для зв’язку з таблицею «Artists»;

є) «Bookmarks» – є проміжною таблицею для забезпечення зв’язку типу «багато до багатьох» між таблицями «Users» та «Artists». Ідентифікується полями «userID» та «artistID», які формують первинний ключ та забезпечують зв’язок між таблицями;

ж) «Reviews» – зберігає оцінки, які користувачі залишають про артистів. На основі отриманих оцінок формуються подальші рекомендації. Ідентифікується полями «userID» та «artistID», які формують первинний ключ та забезпечують зв’язок з відповідними таблицями;

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНОГО СЕРВІСУ

В якості ключових технологій для програмної реалізації клієнтської та серверної частини сервісу було обрано наступні:

- мова програмування Python (версія 3.8);
- бібліотеки numpy, pandas;
- Django REST фреймворк [23];
- СКБД MySQL та бібліотека PyMySQL;

4.1 Реалізація основного алгоритму на мові програмування Python

Python – сучасна мова програмування високого рівня. Python підтримує структурний, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване програмування.

Основні архітектурні риси Python [24]:

- динамічна типізація;
- автоматичне керування пам'яттю;
- повна інтроспекція;
- механізм обробки виключень;
- підтримка багатопоточних обчислень;
- високорівневі структури даних;
- підтримка модулів та пакетів.

Сьогодні Python є однією з найпопулярніших мов програмування, що активно розвивається та знаходить застосування в різноманітних сферах.

Серед основних переваг цієї мови програмування можна виділити наступні [25]:

- висока ефективність для вирішення складних задач, в тому числі в області ML;
- висока швидкодія;
- популярність та активна підтримки спільноти;
- простий синтаксис;
- наявність готових бібліотек для багатьох задач;

4.1.1 Використання сторонніх бібліотек для обробки даних

NumPy – це базовий пакет для виконання наукових обчислень та роботи з алгоритмами машинного навчання з Python. Серед інших можливостей NumPy надає [26]:

- потужний об’єкт для роботи з N-розмірними даними;
- складні функції приведення;
- інструменти для інтеграції C/C++ коду;
- корисні обчислення з лінійної алгебри, чисельних перетворень та випадкових величин;

Крім очевидних наукових застосувань, NumPy також може бути використаний як ефективний багатовимірний контейнер для загальних даних. Це дозволяє NumPy легко та швидко інтегруватись із найрізноманітнішими базами даних.

Pandas – це ще одна корисна та фундаментальна бібліотека для роботи з даними у Python.

Зокрема Pandas спрощує роботу зі складними даними та структурами, забезпечуючи ефективну роботу з обробкою, трансформацією та візуалізацією будь-яких даних.

Отже, мова програмування Python в комбінації з потужними бібліотеками для роботи з даними добре підходить для реалізації нашого гібридного рекомендаційного сервісу із застосуванням алгоритмів машинного навчання. Приклади та деталі програмної реалізації наведено у додатку Б.

4.2 Організація роботи з даними веб-додатку

MySQL – вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам та інструмент для підвищення швидкодії обробки великих баз даних. Зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку серед більшості мов програмування. MySQL надає багатий набір функціональних можливостей, які підтримують безпечне середовище для зберігання, обслуговування і отримання

даних. MySQL характеризується перш за все великою швидкістю, стійкістю і простотою використання, була розроблена

Надійна реляційна база даних MySQL задовольняє усім вимогам нашого сервісу. Для організації взаємодії між базою даних та рекомендаційним ядром будемо використовувати сторонній пакет PyMySQL, що являє собою реалізацію mysql-клієнта на чистому Python.

Для забезпечення надійного та зручного інтерфейсу для взаємодії з клієнтським додатком звернемось до REST-архітектури.

REST (Representational State Transfer) – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

Головними обмеженнями при використанні REST архітектури є:

– клієнт-серверна модель;

– взаємодія без збереження стану;

– однорідний інтерфейс;

Центральною абстракцією в REST є ресурс.

Ресурс – це представлення віртуального об'єкту (наприклад, зображення), реального об'єкту чи колекції об'єктів. Взагалі, ресурс це абстракція, яка може бути представлена чим завгодно, саме розробник API вирішує, що в нього буде ресурсом.

В клієнт-серверній моделі сервер надає якийсь сервіс чи ресурси, які отримують клієнти, виконуючи запити. При чому клієнт може бути чим завгодно: мобільним додатком, браузером або банкоматом, в даному контексті це не має значення.

В архітектурі REST сервер не повинен зберігати ніякої інформації про стан операції. Сесії повинен зберігати клієнт. Це означає, що якщо сервер отримав два різних запити від одного клієнта, вони не повинні впливати один на одного. Через це, всю інформацію, потрібну для здійснення дії, клієнт повинен відправляти в запиті до сервера. Такий підхід дозволяє економити купу часу та ресурсів і полегшує масштабування сервера.

Всі компоненти в архітектурі REST підтримують однорідний інтерфейс. Це зменшує зв'язність між компонентами і сервісами які вони надають і дозволяє нескладно змінювати компоненти при потребі.

Зазвичай REST архітектура будується на основі HTTP-протоколу та підтримує GET, POST, PUT, PATCH, DELETE повідомлення:

Більшість запитів до API нашого сервісу будуть на виконання базових CRUD-операцій (створення-читання-оновлення-видалення).

Таблиця 4.1 ілюструє відповідність цих операцій в SQL та HTTP.

Таблиця 4.1 – Відповідність операцій REST

Операція	SQL	HTTP
CREATE	INSERT	POST
READ	SELECT	GET
UPDATE	UPDATE	PUT/PATCH
DELETE	DELETE	DELETE

Наприклад:

– Отримати користувача з ID = 1 можна виконавши GET-запит: /user/1;

– Створити нового користувача можна виконавши POST-запит: /user;

– Повністю замінити користувача можна виконавши PUT-запит: /user/1;

– Частково оновити інформацію про користувача можна виконавши PATCH-запит: /user/1;

– Видалити користувача з ID =1 можна виконавши DELETE-запит: /user/1;

Одним з найбільш потужних та гнучких інструментів для побудування REST-архітектури на мові Python є фреймворк Django.

Отже реалізацію нашого API будемо розробляти саме на Django, приклади програмного коду наведені у додатку Б.

4.3 Реалізація клієнтського SPA-додатку з використанням React.js

Сучасні браузерні технології, які дозволяють реалізовувати складні та масштабні проекти, а також зручність використання та доступність з майже будь-якого пристрою роблять веб-платформу дуже привабливою для реалізації проектів.

Саме тому, однією з головних вимог до реалізації сервісу стало проектування та розробка надійного, швидкого та сучасного клієнтського веб-додатку зі зручним та інтуїтивно зрозумілим інтерфейсом.

Для реалізації клієнтського додатку було обрано популярну JS-бібліотеку для побудови інтерфейсу користувача – ReactJS.

4.3.1 Особливості розробки додатків з React.js

React – відкрита Javascript-бібліотека для створення інтерфейсів користувача, яка розвивається та підтримується «Facebook inc».

React дозволяє створювати односторінкові веб-додатки (Single Page Application), тобто такі, які працюють без перезавантаження сторінки. Основна ідея полягає в тому, що вхідну точка додатку – це одна веб-сторінка, яка завантажує весь необхідний код та сторонні модулі, що необхідні для роботи.

Цей підхід приніс багато переваг та інноваційних рішень, які значно покращили зручність розробки, підтримки та використання веб-додатків.

Серед плюсів SPA можна виділити:

а) зручний підхід до створювання насичених та складних інтерфейсів користувача, керування анімацією та станом додатку в цілому;

б) більш висока швидкість загрузки, ініціалізації та оновлення даних додатку;

в) відсутність перезавантаження сторінок;

Однією з характерних особливостей саме ReactJS є можливість використовувати JSX – мову розмітки з близьким до HTML синтаксисом, що компілюється в JavaScript.

Також розробники можуть досягати більшої продуктивності додатків за допомогою техніки, що використовує React для здійснення операцій с Document Object Model (DOM), яка називається Virtual DOM.

Document Object Model, або DOM, - це спосіб представлення та взаємодії з об'єктами в HTML, XHTML і XML документах. Відповідно до цієї моделі, кожен такий документ являє собою ієрархічне дерево елементів, яке називають DOM-деревом. Використовуючи спеціальні методи, ми можемо отримати доступ до певних елементів документа і змінювати їх так, як ми хочемо.

Створюючи динамічні інтерактивні веб-сторінки, розробники прагнуть, щоб DOM оновлювався так швидко, як це можливо, одразу після зміни стану певного елемента. Для вирішення даної задачі деякі фреймворки

використовують прийом, який називається «dirty checking» і полягає в регулярному опитуванні стану документа і перевірці змін в структурі даних.

React пропонує абсолютно інший підхід в управлінні DOM-деревом. Virtual DOM, в свою чергу, зберігається в пам'яті. Саме тому замість багаторазового рендерингу нового дерева, React спочатку відображає зміни на Virtual DOM, а потім перемальовує ті вузли, що змінили свій стан, єдиним «патчем». Схема роботи Virtual DOM представлена на рисунку 4.1.

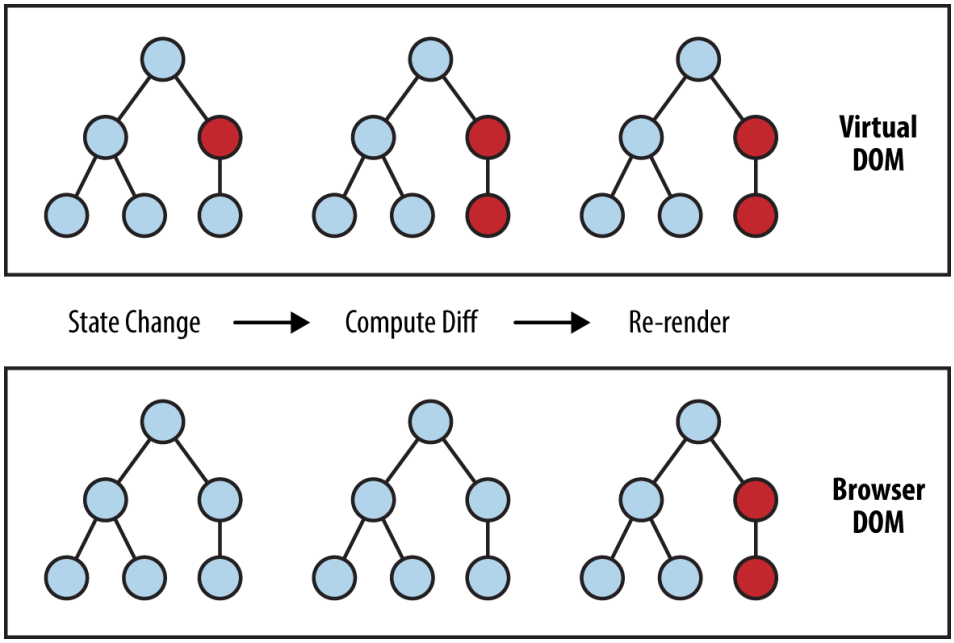

Рисунок 4.1 – Схема роботи Virtual DOM

До переваг ReactJS слід також додати:

– можливість створювати ізоморфні додатки (з використанням серверного рендерингу);

– компонентно-орієнтований підхід до розробки інтерфейсу;

– прозорість роботи зі станом (даними) додатку;

– можливість досить легко портувати веб-додаток на мобільні платформи;

– розвинена екосистема;

З часом вимоги до односторінкового додатку обов’язково зростають та обсяг даних, якими розробник мусить керувати за допомогою JavaScript стає все більшим. Стан додатку зазвичай включає в себе відповіді сервера, локальні дані, стани елементів інтерфейсу тощо.

Управляти великими даними, що постійно змінюються складно, дуже легко втратити контроль над потоками інформації у додатку.

Тому для спрощення управління глобальним станом було вирішено використати React-сумісну бібліотеку для контролю даними – Redux.

Redux базується на принципах єдиного джерела інформації, односпрямованого потоку даних, іммутабельності та деяких інших принципах функціонального програмування. Спрощена схема роботи Redux зображена на рисунку 4.2.

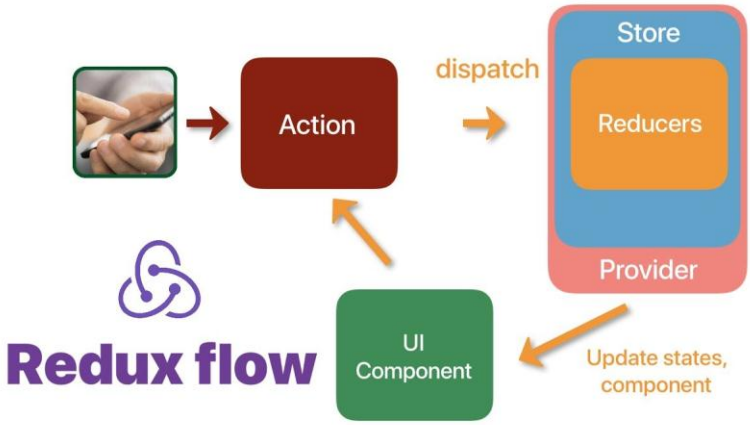

Рисунок 4.2 – Схема роботи Redux

4.3.2 Демонстрація розробленого клієнтського інтерфейсу

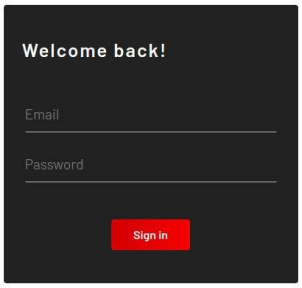

Рисунок 4.3 – Сторінка входу у систему

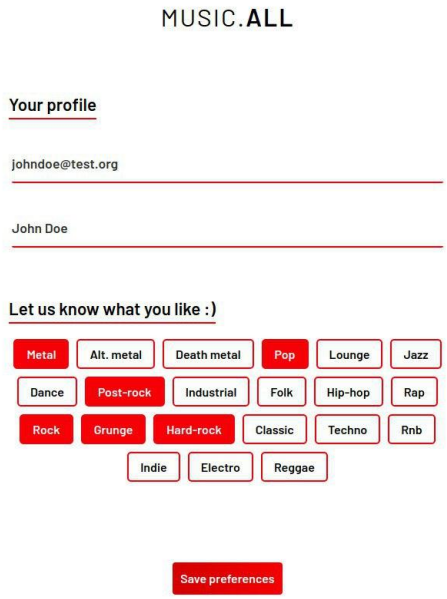

Рисунок 4.4 – Сторінка налаштування вподобань користувача

Рисунок 4.5 – Сторінка музичного каталогу з рекомендаціями

4.4 Перевірка отриманих результатів

Для проведення тестових експериментів було використано набір даних з відкритим доступом, який надає сервіс Last.fm. Цей набор включає в себе такі колекції даних:

- user_artists.dat – userID, artistsID, rating;
- artists.dat – id, name, url, pictureURL;
- tags.dat – tagID, tagValue;

Даний набір містить інформацію про 9000 виконавців та 50000 оцінок (0-5).

Виконавши за допомогою Pandas data-frames об’єднання початкових даних, отримуємо підмножину даних наступного виду (таблиця 4.1):

Таблиця 4.1 – Структура тестових даних

name	userID	artistID	rating
Depeche mode	1642	72	5
Linkin park	1033	377	2
Evanescence	1069	378	4.6
Miley Cyrus	954	461	4

Для проведення експерименту необхідно виділити тестову підмножину. Поділимо тестові дані на дві підмножини:

- дані для розрахунків (80%);
- дані для контролю (20%);

Отже, необхідно спрогнозувати рейтинги для другої групи даних (20%) на основі відомих даних (80%). Таке співвідношення має надати нам інформацію про ефективність нашого алгоритму. Для оцінювання ефективності будемо використовувати середньоквадратичне відхилення (RMSE). Результати експериментів, що були отримані при різній кількості ітерацій тренування моделі за алгоритмом GSD, відображені у таблиці 4.2 та на рисунку 4.3.

Таблиця 4.2 – Результати проведених експериментів

150 ітерацій		300 ітерацій	
Рекомендаційний метод	RMSE	Рекомендаційний метод	RMSE
Контентний	1.05	Контентний	1.05
Колаборативний	0.88	Колаборативний	0.79
Гібридний	0.85	Гібридний	0.76

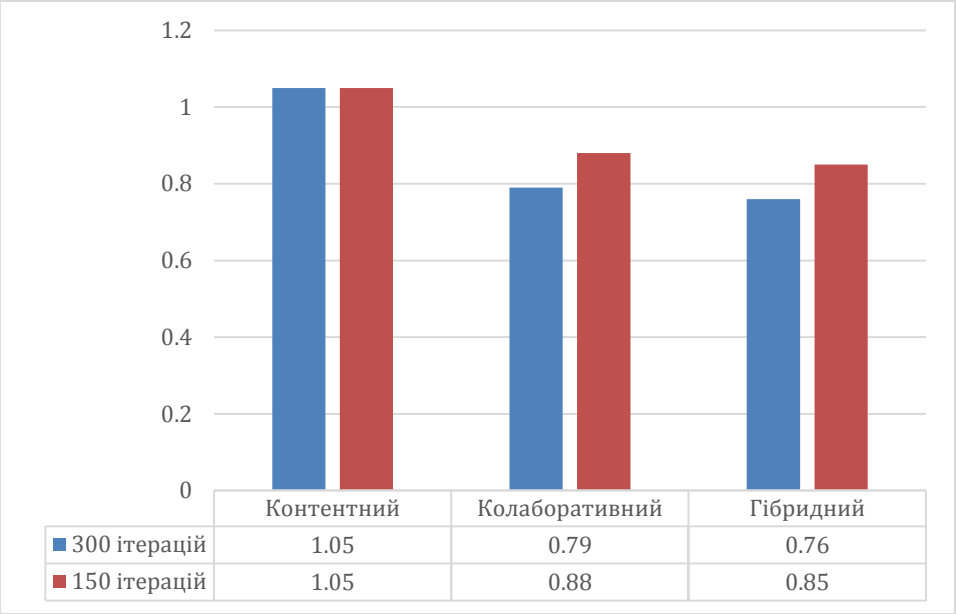

Рисунок 4.3 – Графічне зображення результатів

Отже, як відображають результати проведених експериментів – розроблений гібридний алгоритм дає найбільш точні за RMSE показником результати (0,76), що є досить точним результатом.

В результаті можемо зробити висновок, що колаборативний підхід робить вагомий вклад в кінцевий результат, а також залежить від кількості ітерацій проведених під час калібрування моделі. Тобто застосування алгоритму SVD у комбінації з GSD для налаштування моделі дають досить гарні результати. Проте слід зазначити, що саме застосування гібридного підходу дає найкращий результат, бо використовує переваги обох традиційних підходів та компенсує недоліки кожного з них окремо. Таким чином розроблений метод є гнучким та досить точним за RMSE метрикою, хоча звісно ж, варто враховувати, що задоволеність потреб кінцевого користувача в плані якості рекомендацій завжди носить дещо суб’єктивний характер

ВИСНОВКИ

У результаті виконання магістерської атестаційної роботи було виконано огляд існуючої проблеми та її актуальності, а також запропоновано рекомендаційні системи, як спосіб її вирішення.

На підготовчому етапі було проведено аналіз предметної галузі та її актуальності. Після порівняння вже існуючих, схожих за тематикою сервісів була сформована постановка задачі.

Далі було проведено дослідження методів побудови та оцінювання рекомендаційних систем.

В процесі дослідження було розглянуто типи рекомендаційних систем та математичні моделі на яких вони будуються. Також було проаналізовано основні проблеми, що є специфічними для даного типу систем, способи їх вирішення, методи та алгоритми, які можуть бути для цього застосовані.

В результаті було прийняте рішення про необхідність розробки гібридного механізму на основі колаборативної фільтрації із застосуванням алгоритмів SVD та SGD та розроблено загальний алгоритм роботи системи.

Етап проектування було завершено UML-проектуванням, в ході якого були розроблені діаграми прецедентів, послідовності та комунікацій, також була спроектована база даних сервісу, основні сутності якої були відображені за допомогою ER-діаграми.

На наступному етапі розроблений алгоритм було реалізовано за допомогою серверної мови програмування – Python, а також прототип рекомендаційного веб-додатку з використанням сучасних веб-технологій.

Завершальним етапом стало проведення тестових експериментів на реальній виборці даних, в результаті яких було отримано досить точні результати. Порівняння отриманих результатів підтвердило, що застосування саме гібридного підходу дає найбільш точний за RMSE оцінкою результат.

Загалом, поставлена задача була успішно вирішена в процесі виконання роботи. Однак в подальшому можливе проведення оптимізації та модифікації рекомендаційних алгоритмів, з метою підвищення ефективності роботи системи, розширення функціоналу веб-сервісу та якості бази оцінок та рекомендацій..

ПЕРЕЛІК ПОСИЛАНЬ

1) Kaggle Official Homepage. https://www.kaggle.com

2) Panagiotis Adamopoulos and Alexander Tuzhilin. 2015. On unexpectedness in recommender systems: Or how to better expect the unexpected. ACM Transactions on Intelligent Systems and Technology (TIST) 5, 4 (2015), 54.

3) Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. 2011. Context-aware recommender systems. AI Magazine 32 (2011).

4) Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 19–28.

5) Charu C Aggarwal. 2016. Content-based recommender systems. In Recommender Systems. Springer, 139–166.

6) Charu C Aggarwal. 2016. Ensemble-based and hybrid recommender systems. In Recommender Systems. Springer, 199–224.

7) Charu C Aggarwal. 2016. Evaluating Recommender Systems. In Recommender Systems. Springer, 225–254.

8) Fabio Aiolli. 2013. Efficient top-n recommendation for very large scale binary rated datasets. In Proceedings of the 7th ACM conference on Recommender systems. ACM, 273–280.

9) Masoud Alghoniemy and Ahmed Tewfik. 2001. A Network Flow Model for Playlist Generation. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME). Tokyo, Japan.

10) Masoud Alghoniemy and Ahmed H Tewfik. 2000. User-defined music sequence retrieval. In Proceedings of the eighth ACM international conference on Multimedia. ACM, 356–358.

11) Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 2011. Modern Information Retrieval – The Concepts and Technology Behind Search (2nd ed.). Addison-Wesley, Pearson, Harlow, England.

12) Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion.

13) Basiliyos Tilahun Betru, Charles Awono Onana, and Bernabe Batchakui. 2017. Deep Learning Methods on Recommender System: A Survey of State-of-the-art. International Journal of Computer Applications 162, 10.

14) Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction 12, 4 (2002), 331–370.

15) Xiaoyan Cai, Junwei Han, and Libin Yang. 2018. Generative Adversarial Network Based Heterogeneous Bibliographic Network Representation for Personalized Citation Recommendation. In AAAI.

16) S. Cao, N. Yang, and Z. Liu. 2017. Online news recommender based on stacked auto-encoder. In ICIS. 721–726.

17) Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In Recsys. 288–296.

18) Cheng Chen, Xiangwu Meng, Zhenghua Xu, and Thomas Lukasiewicz. 2017. Location-Aware Personalized News Recommendation With Deep Semantic Analysis. IEEE Access 5 (2017), 1624–1638.

19) Cen Chen, Peilin Zhao, Longfei Li, Jun Zhou, Xiaolong Li, and Minghui Qiu. 2017. Locally Connected Deep Learning Framework for Industrial-scale Recommender Systems.

20) Фаулер М., Скотт К UML. Основы – СПб.: Символ, 2006, 184 с

21) Леоненков А.В Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006, 319 с

22) Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. – СПб.: БХВ-Петербург, 2009. – 528 с.

23) Django rest framework official webpage. www.django-rest-framework.org

24) Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.

25) Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.

26) Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.