МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування, назва факультету)

Кафедра програмування
(повна назва кафедри)

Магістерська робота

ЗАСТОСУВАННЯ МЕТОДІВ АНАЛІЗУ ДАНИХ ДЛЯ РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ

Виконала: студентка групи ПМІм-21
спеціальності
122 – «Комп'ютерні науки»
(шифр і назва спеціальності)

(підпис) Паславська Я. І. (прізвище та ініціали)

Керівник (підпис) Рикалюк Р. Є. (прізвище та ініціали)

Консультант (підпис) Костів В. Я. (прізвище та ініціали)

Рецензент (підпис) Сенько Б. С. (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет *прикладної математики та інформатики*

Кафедра *програмування*

Освітньо-кваліфікаційний рівень *магістр*

Спеціальність *122 – «Комп'ютерні науки»*

«ЗАТВЕРДЖУЮ»

Зав. кафедри *доц. Ярошко С. А.*

« 13 » *вересня* 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Паславська Ярина Ігорівна

(прізвище, ім'я, по батькові)

1. Тема роботи:
Застосування методів аналізу даних для реалізації рекомендаційної системи інтернет-магазину

керівник роботи: *доц. Рикалюк Р. Є.*
затверджена Вченою радою факультету від « *13* » *вересня* 20 *22* р., № *15*

2. Строк подання студентом роботи *12 грудня 2022 р.*

3. Вихідні дані до роботи
Література та інтернет-ресурси за тематикою роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд сучасного стану проблеми
2. Дослідження наявних підходів до побудови рекомендаційних систем
3. Підбір технологій для реалізації
4. Програмна реалізація рекомендаційних систем
5. Розробка клієнт-серверного додатку інтернет-магазину
6. Інтеграція рекомендаційних систем у додаток інтернет-магазину
7. Результати тестування
8. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схеми, діаграми, скріншоти
Презентація дипломної роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *1 вересня 2022 р.*

КАЛЕНДАРНИЙ ПЛАН

№	Найменування етапів дипломної (кваліфікаційної) роботи	Строк виконання етапів роботи	Примітка
1.	*Дослідження наявних підходів до побудови рекомендаційних систем*	*01.09.2022-11.09.2022*	
2.	*Підбір технологій*	*12.09.2022-14.09.2022*	
3.	*Реалізація рекомендаційної системи на основі вмісту*	*15.09.2022-30.09.2022*	
4.	*Розробка вебдодатку*	*01.10.2022-31.10.2022*	
5.	*Інтеграція рекомендаційних систем у вебдодаток*	*01.11.2022-05.11.2022*	
6.	*Тестування*	*05.11.2022-15.11.2022*	
7.	*Оформлення роботи*	*15.11.2022-30.11.2022*	

Студент _____ *Паславська Я. І.*
підпис

Керівник роботи _____ *доц. Рикалюк Р. Є.*
підпис

ЗМІСТ

ЗМІСТ 2

ВСТУП 3

1 ТЕОРЕТИЧНІ ВІДОМОСТІ 5

1.1 Загальні відомості 5

1.2 Колаборативна фільтрація 6

1.3 Рекомендаційна система на основі вмісту 12

2 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ 20

2.1 Загальний опис додатку 20

2.2 Деталі реалізації інтернет-магазину 22

2.3 Особливості бази даних 25

3 ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКОМЕНТАЦІЙНИХ СИСТЕМ 32

3.1 Реалізація алгоритму колаборативної фільтрації 32

3.2 Реалізація рекомендаційної системи на основі вмісту 36

3.3 Результати обчислень реалізованих рекомендаційних систем 43

ВИСНОВКИ 45

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 47

Додаток А. Рекомендаційна система на основі колаборативної фільтрації: код обчислення подібності товарів 49

Додаток Б. Рекомендаційна система на основі колаборативної фільтрації: код реалізації формування пропозицій 51

Додаток В. Рекомендаційна система на основі вмісту: код реалізації попередньої обробки даних 53

Додаток Г. Рекомендаційна система на основі вмісту: код реалізації формування пропозицій 54

ВСТУП

За останні декілька років дедалі більше аспектів життя людей пов’язані з всесвітньою мережею Інтернет, а пандемія коронавірусу значно пришвидшила цей процес. Стало дуже зручно спілкуватися, оплачувати рахунки та податки, перераховувати гроші та робити замовлення без потреби виходити з дому та стояти в чергах у банках або поштових відділеннях. Купівля товарів не стала винятком, тому інтернет-магазини почали розвиватися шаленими темпами. Водночас інтернет-магазини, як і звичайні магазини, мають на меті заохотити клієнтів купувати більше, ніж ті спочатку планували.

Досвід останнього сторіччя показав, що реклама є одним із найкращих способів заохотити клієнтів споживати більше товарів, а фактично, витрачати більше грошей. Але навіть її можна розвивати. Одним із напрямів такого розвитку є персоналізована реклама, що була недоступною ще декілька років тому, а тепер є звичною. Цей напрям реклами був недоступним, тому що звичайні магазини не мали можливості збирати ні загальну інформацію про клієнтів (ім’я, вік, стать і т.д.), ні про те, що кожен із них купує, а тим більше що кожному з них подобається, щоб потім здійснювати аналіз.

Натомість з розвитком інтернет-магазинів були реалізовані інструменти, що дають можливість збирати всю можливу інформацію про клієнтів, тобто особисту інформацію про особу, список її покупок, залишені відгуки, а також перелік товарів, які вона переглядала, та тривалість цього перегляду.

Ця інформація відкриває для маркетингу цілком нові можливості. Наприклад, відгук користувача про товар не лише дає інформацію для клієнтів, які думають про купівлю цього товару, але й допомагає системі зрозуміти, які речі подобаються цьому конкретному користувачу, а які ні. Ці дані дають можливість формувати персоналізовану рекламу.

Персоналізовану рекламу також можна назвати рекомендаційною системою, бо інтернет-магазин намагається підібрати рекомендації для конкретного клієнта на основі тієї інформації, що система вже про нього зібрала.

Зрештою, від цього виграє не лише магазин, але й покупець. Інколи користувач прагне купувати схожі до уже придбаних товари, однак їхній пошук серед великої кількості товарів у каталозі може бути проблемою.

Інтернет-магазини, а також вебсервіси, почали розробляти різні алгоритми обчислення рекомендацій, щоб зацікавити та/або «прив’язати» до себе клієнтів.

Завданнями цієї магістерської роботи є:

а) дослідити наявні підходи до побудови рекомендаційних систем, з’ясувати їхні переваги та недоліки;

б) освоїти основні алгоритми, що лежать в основі обчислень рекомендаційних систем;

в) підібрати найоптимальніші технології програмування для реалізації рекомендаційних систем;

г) реалізувати декілька рекомендаційних систем, в основі яких лежать різні алгоритми обчислення;

г) розробити клієнт-серверний додаток книжкового інтернет-магазину;

д) інтегрувати рекомендаційні системи в реалізований додаток для візуалізації результатів, а також для можливого використання у майбутньому.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Загальні відомості

Рекомендаційна система – це система, що займається побудовою персоналізованих пропозицій для конкретного користувача на основі інформації про його покупки та вподобання, а також даних про товари, тобто їхні характеристики та/або опис.

Алгоритми рекомендаційних систем можуть використовувати різні доступні системі дані для ефективного обчислення пропозицій, наприклад:

а) інформація про покупки користувача;

б) текстові відгуки користувача;

в) вподобання користувача: оцінка товару за визначеною шкалою, наприклад 5-бальною, або «сподобався» / «не сподобався»;

г) характеристики товару: категорія, автор / виробник /видавництво, опис.

Рекомендаційні системи поділяються на декілька видів у залежності від того, як і з якими даними вони працюють, наприклад:

а) рекомендаційні системи на основі алгоритму колаборативної фільтрації – використовують інформацію про здійснені покупки або оцінки користувача;

б) рекомендаційні системи на основі вмісту (content-based recommendation systems) – використовують властивості товару та оцінки та/або відгуки користувача;

в) рекомендаційні системи на основі знань (knowledge-based recommendation systems) – використовують властивості товару;

г) гібридні рекомендаційні системи – передбачають одночасне використання кількох вищезгаданих рекомендаційних систем і потребують тих даних, які необхідні для використаних рекомендаційних систем.

Детальніше всі підходи описані в [4].

У магістерській роботі було розглянуто та реалізовано 2 підходи: рекомендаційної системи на основі алгоритму колаборативної фільтрації та рекомендаційної системи на основі вмісту.

1.2 Колаборативна фільтрація

Колаборативна фільтрація – це метод формування рекомендацій для користувача на основі вже зібраних даних про його покупки та вподобання у порівнянні з покупками та вподобаннями інших користувачів. Основна ідея цього методу – це припущення, що користувачі, що мали схожі вподобання у минулому, матимуть їх і в майбутньому [15].

Існує два види колаборативної фільтрації [3]:

а) вид, що базується на пам’яті. Його завданням є збір даних про оцінки товарів користувачами та за допомогою метрики подібності (як правило косинусну міру подібності або коефіцієнт кореляції Пірсона) пошук схожих користувачів / схожих товарів та обчислення прогнозу відсутніх оцінок;

б) вид, що базується на моделі. Його завданням є збір даних про оцінки товарів користувачами та побудова на основі цих даних моделі для алгоритмів машинного навчання (нейронні мережі, факторизація матриць та ін.) та передбачення відсутніх оцінкок.

У магістерській роботі було досліджено перший вид: підхід, що базується на пам’яті. Своєю чергою він також поділяється за способами обробки даних на два види [9]:

а) колаборативна фільтрація на основі порівняння користувачів (user-based). Основна ідея: на основі зібраних даних (покупки, відгуки) знайти користувачів, які мають найбільше спільного у вподобаннях та рекомендувати покупцю ті товари, які він ще не оцінив / не купив, але які позитивно оцінив користувач зі схожими вподобаннями;

б) колаборативна фільтрація на основі порівняння товарів (item-based). Основна ідея: на основі зібраних даних (покупки, відгуки) знайти товари, що мають найбільші значення коефіцієнтів подібності, і при виборі певного

товару рекомендувати декілька інших товарів. Можна розширити цей функціонал індивідуальними пропозиціями для користувачів: знайти товари, що найбільше схожі на ті, що сподобалися користувачу, обчислити прогнозовану оцінку і рекомендувати товари з найбільшими передбаченими оцінками.

Перевагами першого способу є: нескладна реалізація, набір пропозицій є різноманітнішим та незалежним від контексту.

Недоліками ж цього способу є розрідженість матриці, бо користувачі оцінюють порівняно мало товарів, а також проблема «холодного старту», тобто неможливість підібрати рекомендації для нових користувачів, адже система ще не має інформації про їхні вподобання. Також варто врахувати, що при збільшенні кількості користувачів: обчислення стають складнішими і потребують більше ресурсів; необхідно більше місця для збереження отриманих результатів. Складною є також підтримка системи, бо при додавання нових користувачів необхідно частіше оновлювати таблицю подібності, що зумовлює додаткове навантаження на сервери.

Багато з цих недоліків можна вирішити, вибравши інший спосіб формування рекомендацій, а саме колаборативну фільтрацію на основі порівняння товарів. Основні переваги: велика точність передбачених оцінок, інтуїтивно зрозуміле формування пропозицій, відсутність потреби оновлювати таблицю подібності часто, оскільки ці значення зазвичай є доволі стабільними. Завдяки цим перевагам великі платформи використовують саме цей спосіб для уникнення великого навантаження на сервери. Таблицю подібності можна оновлювати, коли сервер має невелике навантаження або ж взагалі винести ці обчислення на інший сервер.

На жаль, цей спосіб теж має недоліки, до яких належать: рекомендація очевидно пов’язаних товарів (той самий автор, та сама серія) або ж товарів, про які користувач вже знає.

Зважаючи на зазначені переваги другого способу формування рекомендацій (колаборативна фільтрація на основі порівняння товарів) розглянемо його детальніше.

Покроковий алгоритм роботи цього методу [16]:

1. Збір інформації про поведінку користувача щодо товару. Це може бути інформація у вигляді наборів значень: «купив»/«не купив», «купив»/«переглядав»/«не купив», «сподобалося»/«не голосував»/«не сподобалося», представлена у певному визначеному числовому форматі (наприклад 1, 0; -1, 0, 1).

Також можна використовувати інформацію про оцінки від користувачів. Кожен клієнт системи, при бажанні, може написати відгук та оцінити товар (наприклад у межах від 1 до 5), система зберігає таблицю цих оцінок та використовує її для подальшого аналізу. У цій роботі використовується саме цей підхід.

2. Обчислення коефіцієнтів подібності між всіма парами товарів на основі зібраних значень оцінок цих товарів. До уваги беруться лише оцінки тих користувачів, що оцінили обидва товари.

Для обчислення коефіцієнтів подібності існують різні метрики. Найпопулярнішими є косинусна міра подібності та коефіцієнт кореляції Пірсона. Для обчислень у цій роботі використовується коефіцієнт кореляції Пірсона, який поданий у формулі (1.1).

$$simil(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})(r_{y,i} - \overline{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \overline{r_y})^2}}, \quad (1.1)$$

де x, y – товари, що порівнюються, I_{xy} – це множина користувачів, що оцінили обидва товари, $r_{x,i}$ – оцінка (rate) товару x, яку поставив користувач $i, i \in I_{xy}$, $\overline{r_x}$ – середнє значення оцінок товару x. Ця метрика повертає значення в межах $[-1, 1]$: -1 означає, що товари зовсім різні, тобто якщо користувач поставив одному товару позитивну оцінку, то іншому негативну, а 1 – що товари надзвичайно схожі. У контексті цієї роботи відрізок $[-1, 1]$ було спроєктовано на відрізок $[0, 1]$ для спрощення обчислення рекомендацій під час виконання наступних кроків.

3. Формування матриці коефіцієнтів подібності між парами товарів, отриманих під час виконання кроку 2. Варто зауважити, що $simil(x, y) =$

$simil(y, x)$, тобто ця матриця є симетричною. Це дозволяє зменшити кількість обчислень та обсяг використаної пам'яті для зберігання.

Обчислену під час виконання цього кроку матрицю коефіцієнтів подібності між парами товарів можна використати для різних цілей, що описані у наступних кроках.

4. Формування пропозицій на основі товару. Користувач заходить на сторінку з детальним описом товару, під яким знаходиться секція на кшталт «Схожі товари». У цій секції відображають найбільше схожі на вибраний товари. Для цього в обчисленій в попередньому кроці матриці знаходять елементи з найбільшими коефіцієнтами подібності до вибраного товару і формують з них список пропозицій.

5. Формування індивідуальних пропозицій для користувачів. Для цього обчислюються прогнозовані оцінки конкретних користувачів на неоцінені ними товари, та відбираються товари з найвищими прогнозованими оцінками.

Для кращого розуміння алгоритму наведемо приклад.

Додаток, для якого реалізовується рекомендаційна система, – це інтернет-магазин книг. У базі даних є 10 користувачів, 10 книг та оцінки, які поставили користувачі цим книгам. Для зручності ці оцінки записано у таблицю, зображену на рисунку 1.1, яка відповідає розрідженій матриці.

	клієнт #1	клієнт #2	клієнт #3	клієнт #4	клієнт #5	клієнт #6	клієнт #7	клієнт #8	клієнт #9	клієнт #10
книга #1	3		1		4	2			5	2
книга #2		2	5				5	2		
книга #3		3		4	4	1			3	1
книга #4	5		5				3	2		
книга #5	2		3		1	4			3	3
книга #6	2	1		2				5		4
книга #7		2			2	3	3		1	
книга #8		3	4	5		1	2	3	4	
книга #9	1		5		5			1		5
книга #10		2		1	4	2	5		3	

Рисунок 1.1 – Оцінки книг

Допустимо, що необхідно сформувати список пропозицій на основі книги «книга #5» та для користувача «клієнт #8».

На основі таблиці оцінок (рисунок 1.1) було обчислено таблицю коефіцієнтів подібності між книгами, використовуючи коефіцієнт кореляції Пірсона та беручи до уваги лише користувачів, що оцінили обидва товари. Ця таблиця зображена на рисунку 1.2.

	книга1	книга2	книга3	книга4	книга5	книга6	книга7	книга8	книга9	книга10
книга1	1.000	0.000	0.926	0.000	0.281	0.000	0.009	0.639	0.371	0.827
книга2	0.000	1.000	0.000	0.878	0.000	0.000	1.000	0.500	1.000	1.000
книга3	0.926	0.000	1.000	0.000	0.081	0.107	0.176	0.985	0.000	0.590
книга4	0.000	0.878	0.000	1.000	0.000	0.000	0.000	0.827	0.750	0.000
книга5	0.281	0.000	0.081	0.000	1.000	1.000	0.664	0.000	0.587	0.009
книга6	0.000	0.000	0.107	0.000	1.000	1.000	0.000	0.361	0.594	0.000
книга7	0.009	1.000	0.176	0.000	0.664	0.000	1.000	0.028	0.000	0.592
книга8	0.639	0.500	0.985	0.827	0.000	0.361	0.028	1.000	1.000	0.291
книга9	0.371	1.000	0.000	0.750	0.587	0.594	0.000	1.000	1.000	0.000
книга10	0.827	1.000	0.590	0.000	0.009	0.000	0.592	0.291	0.000	1.000

Рисунок 1.2 – Таблиця подібності між книгами

Для виконання першого завдання (формування пропозицій на основі книги «книга #5») необхідно лише відсортувати коефіцієнти подібності цієї книги з іншими у порядку спадання, вибрати необхідну кількість елементів з найбільшими значеннями коефіцієнтів подібності та відобразити їх.

Для книги «книга #5» список пропозицій з трьох найбільш подібних елементів складається з таких книг: «книга #6», «книга #7» та «книга #9».

Формування пропозицій для користувача «клієнт #8» вимагає додаткових обчислень. Необхідно обчислити прогнозовані оцінки користувача «клієнт #8» на товари «книга #1», «книга #3», «книга #5» «книга #7» та «книга #10».

Прогнозовані оцінки обчислюються на основі вже наявних оцінок та коефіцієнту подібності оцінених та неоцінених товарів. Для розуміння алгоритму розглянемо результати обчислень, подані на рисунку 1.3. У першій колонці перераховані назви оцінених користувачем «клієнт #8» книг, а у другій – їхні оцінки. У колонці «схожість з книгою #x» наведені значення коефіцієнтів

подібності відповідних оцінених книг та книги *x*. У колонці «очікувана оцінка для книги #x» наведені значення очікуваних оцінок для книги *x*, обчислені за допомогою множення доступної оцінки на коефіцієнт подібності між книгою *x* та відповідною книгою у рядку.

книги, що вже мають відгуки	оцінки від клієнта #8	схожість з книгою #1	очікувана оцінка для книги #1	схожість з книгою #3	очікувана оцінка для книги #3	схожість з книгою #5	очікувана оцінка для книги #5	схожість з книгою #7	очікувана оцінка для книги #7	схожість з книгою #10	очікувана оцінка для книги #10
книга #2	2	0	0	0	0	0	0	1	2	1	2
книга #4	2	0	0	0	0	0	0	0	0	0	0
книга #6	5	0	0	0,107	0,535	1	5	0	0	0	0
книга #8	3	0,639	1,917	0,985	2,955	0	0	0,028	0,084	0,291	0,873
книга #9	1	0,371	0,371	0	0	0,587	0,587	0	0	0	0

Рисунок 1.3 – Проміжні результати для обчислення пропозицій

Для отримання остаточного прогнозу оцінки для книги *x* необхідно поділити суму значень у колонці «очікувана оцінка для книги #x» на суму значень у колонці «схожість з книгою #x». Отримані результати прогнозованих оцінок зображені на рисунку 1.4. Рекомендувати товари користувачеві «клієнт #8» варто у зазначеному порядку. У випадку більшої кількості товарів вибирають лише необхідну кількість книг із відсортованого списку.

	0	1
0	книга5	3.520
1	книга3	3.196
2	книга1	2.265
3	книга10	2.225
4	книга7	2.027

Рисунок 1.4 – Пропозиції для користувача «клієнт #8»

У цьому прикладі добре проілюстрована перевага колаборативної фільтрації на основі порівняння товарів, що полягає у відсутності необхідності часто оновлювати таблицю коефіцієнтів подібності між книгами. Значення не змінюються різко і їхній вплив на кінцевий результат пропозицій не є дуже суттєвим. Звісно, будуть невеликі втрати в точності передбачених оцінок, але у випадку інтернет-магазину це не є критичним аспектом.

1.3 Рекомендаційна система на основі вмісту

У випадку рекомендаційних системи на основі алгоритму колаборативної фільтрації враховуються лише оцінки або відгуки користувачів, але, де-факто, властивості товарів ніяк не впливають на формування пропозицій. На противагу, у рекомендаційних системах на основі вмісту (content-based recommendation systems) властивості товарів відіграють ключову роль. Детальніше цей тип описано в [4].

Як і для рекомендаційних систем на основі алгоритму колаборативної фільтрації існує два способи формування пропозицій: на основі товару (тобто підбір схожих товарів з метою їх рекомендації) та для користувача (тобто підбір товарів, що можуть зацікавити конкретного користувача). Для цих двох видів використовуються два різні підходи.

У випадку рекомендацій на основі товару порівнюють властивості товару та пропонують найбільш схожі. Для порівняння використовують одну з доступних метрик подібності. Так, крім розглянутих вже косинусної міри подібності та коефіцієнту кореляції Пірсона, використовують ще і евклідову відстань.

Натомість для персональних рекомендацій будують модель, що складається з властивостей товарів, на які вже є відгуки, та самих відгуків. На основі цієї моделі намагаються передбачити відгуки на ще неоцінені товари, а потім пропонують товари з найвищими передбаченими оцінками. Для передбачення нових відгуків можна використовувати класифікацію або регресію, наприклад класифікацію методом найближчих сусідів (Nearest Neighbors Classification), наївний класифікатор Баєса (Naive Bayes) або класифікацію на основі правил.

Такий підхід до формування рекомендації має свої переваги та недоліки. До переваг належать:

а) немає проблеми «холодного старту» для нових товарів, бо одразу ж після додавання нового товару його властивості можна порівнювати з властивостями інших товарів і додавати до рекомендацій, якщо він відповідає вимогам;

б) відгуки та оцінки інших користувачів не впливають на рекомендації для конкретного користувача, оскільки враховується лише інформація, зібрана щодо нього;

в) рекомендаційні системи на основі вмісту надають пояснення, чому саме такі пропозиції були сформовані, беручи за основу властивості товарів.

Водночас недоліками є:

а) проблема «холодного старту» для нових користувачів, адже для роботи алгоритм потребує якнайбільше інформації про вподобання користувача для того, щоб ефективно передбачати відгуки та формувати рекомендації, а тому, якщо інформації є мало або немає взагалі, робота алгоритму буде дуже неефективною;

б) система може рекомендувати дуже близькі, очевидні товари, про які користувач вже знає, що не спонукатиме користувача здійснювати покупки.

Перш ніж використовувати будь-який з описаних вище алгоритмів необхідно підготувати доступні дані, щоб на їхній основі можна було проводити обчислення. Цей крок називається попередня обробка даних (data preprocessing) і є надзвичайно важливим етапом побудови будь-якої моделі.

Як уже було згадано, рекомендаційні системи на основі вмісту (content-based recommendation systems) працюють з властивостями товарів. Прикладом товару оберемо книгу. Її основними властивостями є автор, назва, жанри, анотація. Також можна згадати про ціну, видавництво, кількість сторінок та мову видання, але вони не є ключовими, тому їх можна не враховувати. Отже, перший крок попередньої обробки є очевидним: необхідно виділити найважливіші властивості, щоб не мати надлишку даних.

Наступним кроком є необхідність визначити, чи всі ключові властивості будуть впливати однаково, чи все ж є більш та менш важливі критерії. Для цього використовують ваги. Характеристика з більшою вагою сильніше впливає на кінцевий результат, ніж характеристика з меншою вагою. У контексті цієї роботи всі властивості є однаково важливими.

Характеристики товару можуть бути числовими або текстовими даними. З числовими даними все зрозуміло, але текстові необхідно підготувати для роботи з

ними. Наприклад, розглянувши вибрані характеристики книги: автор, назва, анотація, жанр, легко зауважити, що всі вони подані у текстовому форматі, тому наступний крок полягає у перетворені їх у числові формати, щоб на їхній основі можна було проводити обчислення. Існують різні підходи до такого перетворення, найпопулярнішими серед яких є кодування міток (label encoding) [2], запис тексту в вигляді векторів на основі частоти вживання слів (count vectorization, також відоме під назвою one-hot encoding) [6] та TF-IDF векторизація (Term Frequency Inverse Document Frequency – частота слова-зворотна частота документа) [8].

У випадку, коли властивість набуває категоріальних значень, то кожному значенню із набору можливих ставиться у відповідність якесь конкретне число. Це перетворення називається кодуванням міток. Наприклад, якщо властивість описує кольори і має такі значення: «білий», «червоний», «синій» та інші, то їх перетворюють у числа, тобто зрозумілу форму для машинних алгоритмів. Приклад перетворення показано на рисунку 1.5. Варто зауважити, що перед цим кодуванням необхідно переконатися, що всі категоріальні значення записані у нижньому регістрі.

колір	код
білий	0
синій	1
червоний	2
білий	0
чорний	3

Рисунок 1.5 – Приклад алгоритму кодування міток

Векторизація тексту на основі частоти вживання слів (count vectorization) – це також перетворення тексту в зрозумілий для машинних алгоритмів вигляд, але використовується він для складніших текстів, ніж просто мітки, наприклад опис товару, а у випадку книги – її анотація.

Однак варто зауважити, що для використання цього перетворення необхідно мати текстові характеристики, наприклад описи, всіх товарів одразу. Це необхідно,

бо таке перетворення формує словник з усіх слів, доступних у всіх описах, і записує кожний опис у вигляді вектора, що має довжину цього спільного словника, та у якому значення елементу дорівнює кількості входжень відповідного слова у цей опис. Наприклад, нехай є доступними два описи: «Чорний чай з лимоном. Запашний чорний чай.» та «Зелений чай з додаванням лимону.». У такому випадку результат векторизації тексту буде виглядати таким чином, як наведено у рисунку 1.6.

	Чорний	чай	з	лимоном	Запашний	чорний	Зелений	додаванням	лимону	.
опис 1	1	2	1	1	1	1	0	0	0	2
опис 2	0	1	1	0	0	0	1	1	1	1

Рисунок 1.6 – Приклад алгоритму векторизації тексту без попередньої обробки

Варто зауважити, що векторизація, зображена на рисунку 1.6, залежить від регістру букв у словах та відмінків слів. Наприклад, слова «Чорний» та «чорний», а також «лимоном» та «лимону» є різними елементами у словнику, але є очевидним, що зміст опису вони не змінюють. Тому алгоритм векторизації також потребує окремої попередньої підготовки даних. Для цього використовують обробку природньої мови (natural language processing), що передбачає такі кроки:

а) переведення всіх слів у нижній регістр;

б) видалення знаків пунктуації;

в) видалення стоп-слів, тобто слів, що не несуть лексичного значення: наприклад сполучники, прийменники, артиклі (у англійській мові) тощо;

г) виділення частини слова, що відповідає за лексичне значення (stemming). Цей крок гарантує те, що векторизація тексту не залежить від відмінків, особи та числа, бо враховується лише та його частина, що містить лексичне значення.

Результат векторизації вищеописаного прикладу з попередньою обробкою тексту зображений на рисунку 1.7. Очевидно, що такий результат є інформативнішим.

	чорн	чай	лимон	запашн	зелен	додав
опис 1	2	2	1	1	0	0
опис 2	0	1	1	0	1	1

Рисунок 1.7 – Приклад алгоритму векторизації тексту з попередньою обробкою

Зазвичай використовують не просто векторизацію тексту на основі частоти вживання слова, а дещо ускладнений алгоритм перетворення тексту в зрозумілий для машинних алгоритмів вигляд, а саме алгоритм FT-IDF. FT-IDF – це абревіатура до Term Frequency-Inverse Document Frequency, дослівно «частота слова-зворотна частота документа». У цьому контексті «термін» означає слово в документі. Цей алгоритм, поданий у формулі (1.2), оцінює важливість слів у документі.

$$TF\text{-}IDF = TF(t,d) * IDF(t), \tag{1.2}$$

де $TF(t,d)$ – це частота слова, тобто скільки разів слово *t* з'являється в документі *d*, а $IDF(t)$ – це обернена частота документів, тобто величина, обернена до кількісті документів, у яких з'являється слово *t*.

$$IDF(t) = log\frac{n}{DF(d,t)}, \tag{1.3}$$

де *n* – це загальна кількість документів, а $DF(d,t)$ – це кількість документів, в яких трапляється слово *t*.

Алгоритм FT-IDF перетворює опис кожного доступного товару у відповідний вектор, у якому значення показують не кількість разів, скільки відповідне слово вживається в цьому документі, а його важливість у цьому документі у порівнянні з іншими. Цей алгоритм найчастіше використовується для перетворення властивостей з текстового формату у числовий для подальшої роботи.

Отже, попередня підготовка даних для побудови моделі передбачає такі кроки:

а) вибір властивостей товарів, які будуть враховуватися при розрахунках;

б) при потребі – надання ваг вибраним властивостям;

в) перетворення властивостей з текстовими даними у числові формати.

Результатом такої підготовки для кожного товару є одна або декілька властивостей у числових форматах, які можна використовувати для класифікації

або регресії. У випадку декількох властивостей їх необхідно об'єднати в один вектор. Далі ці вектори об'єднують в одну матрицю властивостей, де кожен рядок відповідає за інший товар. На основі цієї матриці і проводять обчислення рекомендацій.

У випадку рекомендацій на основі товару за допомогою вибраної метрики подібності порівнюють вектор властивостей поточного товару з векторами властивостей інших товарів та пропонують ті, з якими коефіцієнт подібності є найвищим. Для виконання цієї роботи як метрику було вибрано косинусну міру подібності, яка подана у формулі (1.4) [15].

$$simil(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \quad (1.4)$$

де A, B – товари, описані двома ненульовими векторами властивостей, θ – це кут між цими векторами. Фактично це скалярний добуток двох векторів, поділений на добуток їх евклідових норм.

У випадку персональних рекомендацій передбачення реакції користувача (оцінка, відгук) формується на основі властивостей товарів та вже відомих оцінок. Тобто необхідно порівняти оцінені та неоцінені товари на основі їхніх характеристик, а потім передбачати близькі оцінки для схожих товарів.

Для передбачення нових оцінок можна використовувати методи класифікації або регресії. Для виконання цієї роботи було вибрано наївний класифікатор Баєса.

Наївний класифікатор Баєса – це алгоритм класифікації, основою якого є теорема Баєса. Цей алгоритм обчислює ймовірність належності об'єкту до кожного можливого класу і зараховує його до того класу, для якого ймовірність найбільша [10].

Теорема Баєса обчислює ймовірність події B після того, як відбувала подія A, тобто значення $P(B|A)$ [1] за формулою (1.5):

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}. \quad (1.5)$$

У випадку класифікації, формулу (1.5) можна переписати у вигляді, поданому у формулі (1.6):

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (1.6)$$

де y – мітка класу (оцінка товару від користувача), а X – вектор властивостей об'єкту, який необхідно класифікувати. Він має вигляд, поданий у формулі (1.7):

$$X = (x_1, x_2, x_3, \dots, x_n). \quad (1.7)$$

Класифікатор Баєса називається наївним, тому що існує «наївне» припущення, що всі властивості є незалежними між собою, що зображено у формулі (1.8):

$$P(A, B) = P(A)P(B). \quad (1.8)$$

Отже, з формул (1.6) та (1.8) можна отримати:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\cdots P(x_n|y)P(y)}{P(x_1)P(x_2)\cdots P(x_n)}. \quad (1.9)$$

Формулу (1.9) ще можна записати у вигляді:

$$P(y|x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)\cdots P(x_n)}. \quad (1.10)$$

Оскільки знаменник залишається константою для кожного можливого значення y, то його можна не враховувати. Отже, кінцева формула має такий вигляд

$$P(y|x_1, \dots, x_n) = P(y)\prod_{i=1}^{n} P(x_i|y). \quad (1.11)$$

За допомогою формули (1.11) можна знайти ймовірність приналежності об'єкту до кожного можливого класу. Далі необхідно знайти клас з найбільшою ймовірністю і приписати цей клас об'єкту, що розглядався, тобто обчислити y за допомогою формули (1.12):

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y), \quad (1.12)$$

Важливим етапом роботи з наївним класифікатором Баєса є вибір ймовірності, за допомогою якої будуть проводитися обчислення. Найпопулярнішою ймовірністю є нормальний розподіл Гауса, зображений у формулі (1.13):

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_1-\mu_y)^2}{2\sigma_y^2}\right), \quad (1.13)$$

де μ – математичне сподівання, а σ – дисперсія випадкової величини.

У контексті рекомендаційної системи формулу (1.12) використовують для знаходження найбільш ймовірної оцінки від користувача серед доступних, наприклад у певних межах ([1-5]), або «сподобався» / «не сподобався». Далі формуються пропозиції в порядку зменшення передбаченої оцінки.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

2.1 Загальний опис додатку

У рамках виконання цієї роботи був реалізований вебдодаток, що складається з декількох частин: інтернет-магазину книг та двох рекомендаційних систем. Інтерфейс інтернет-магазину є двомовним. Доступними є українська та англійська мови.

Інтернет-магазин книг – це клієнт-серверний додаток. Клієнтська частина була реалізована із використанням таких технологій:

а) HTML, CSS – базові інструменти для реалізацій користувацьких інтерфейсів;

б) React.js – це бібліотека для побудови користувацьких інтерфейсів вебсайтів та додатків. Її основними перевагами є гнучкість та сумісність з різними типами додатків та платформ. Ця бібліотека дозволяє створювати різноманітні компоненти та повторно їх використовувати, також характеризується високою продуктивністю[12];

в) Ant Design – бібліотека користувацького інтерфейсу, що полегшує роботу над розробкою інтерфейсу, бо дозволяє користуватися вже реалізованими компонентами та їх функціоналом, за потреби змінюючи їхній вигляд відповідно до заданого дизайну.

Серверна частина розроблена за допомогою вебфреймфорку ASP.Net Core. Це кросплатформний вебфрейворк з відкритим програмним кодом, продуктивність, гнучкість і масштабованість якого сприяє написанню простого в обслуговуванні та ефективного програмного коду, що можна використовувати повторно. Також суттєвими перевагами цього фрейморку є автоматичне керування пам'яттю, незалежність від мови програмування, можливість асинхронного програмування та інші [7].

База даних має нереляційну модель, системою керування бази даних є MongoDB.

Рекомендаційні системи, що побудовані на основі описаних вище алгоритму колаборативної фільтрації та алгоритму на основі вмісту, реалізовані за допомогою мови програмування Python та вебфреймфорку Flask. Ці технології були вибрані, тому, що бібліотеки та фрейморки мови програмування Python стали загальноприйнятим інструментом для розв'язування задач аналізу даних та машинного навчання, оскільки дозволяють використовувати вже розроблені структури даних, алгоритми та методи. Водночас вебфреймфорк Flask – один з найпопулярніших фреймворків мови програмування Python для побудови прикладних програмних інтерфейсів вебзастосунків.

Для кожної рекомендаційної системи створено прикладний програмний інтерфейс, який надає методи для формування пропозицій на основі товару (книги) або для конкретного користувача. Комунікація між рекомендаційними системами та інтернет-магазином відбувається за допомогою HTTP-запитів, які формує та відправляє серверна частина інтернет-магазину на кожний з прикладних програмних інтерфейсів. Також вона відповідає за передачу результатів роботи рекомендаційних систем на інтерфейс користувача. Діаграма комунікації зображена на рисунку 2.1.

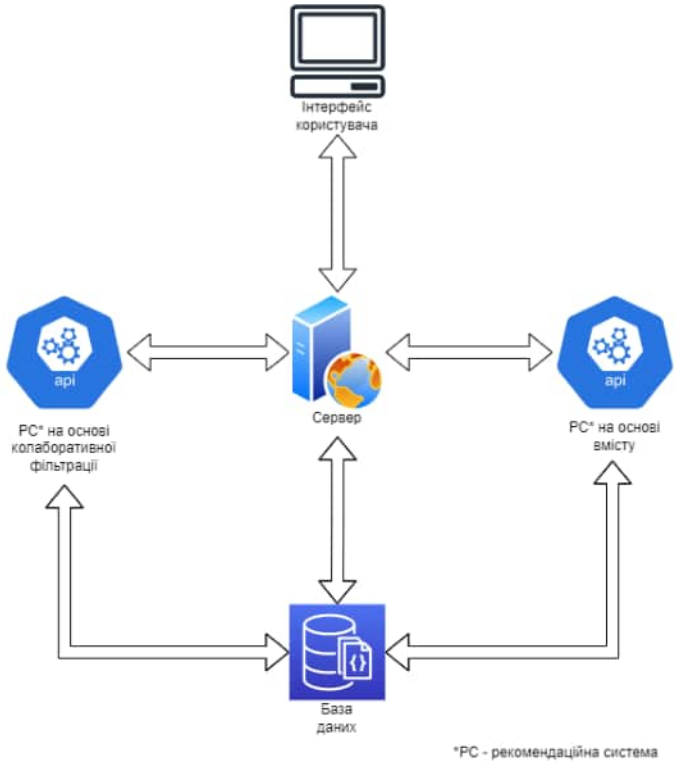

Рисунок 2.1 – Діаграма комунікації між всіма складовими додатку

2.2 Деталі реалізації інтернет-магазину

Діаграма прецедентів [17] інтернет-магазину зображена на рисунку 2.2.

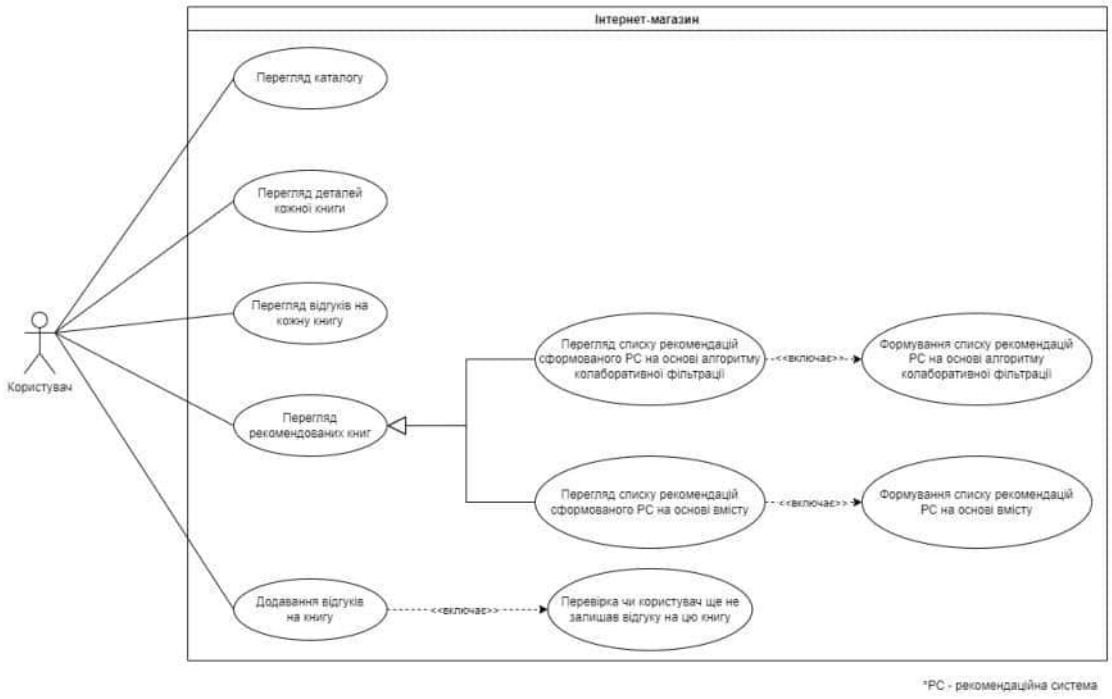

Рисунок 2.2 – Діаграма прецедентів інтернет-магазину

Інтернет-магазин книг реалізовує такий функціонал:

а) перегляд всього каталогу книг, також є доступними функції пошуку, фільтрування та сортування. Сторінка каталогу книг зображена на рисунку 2.3;

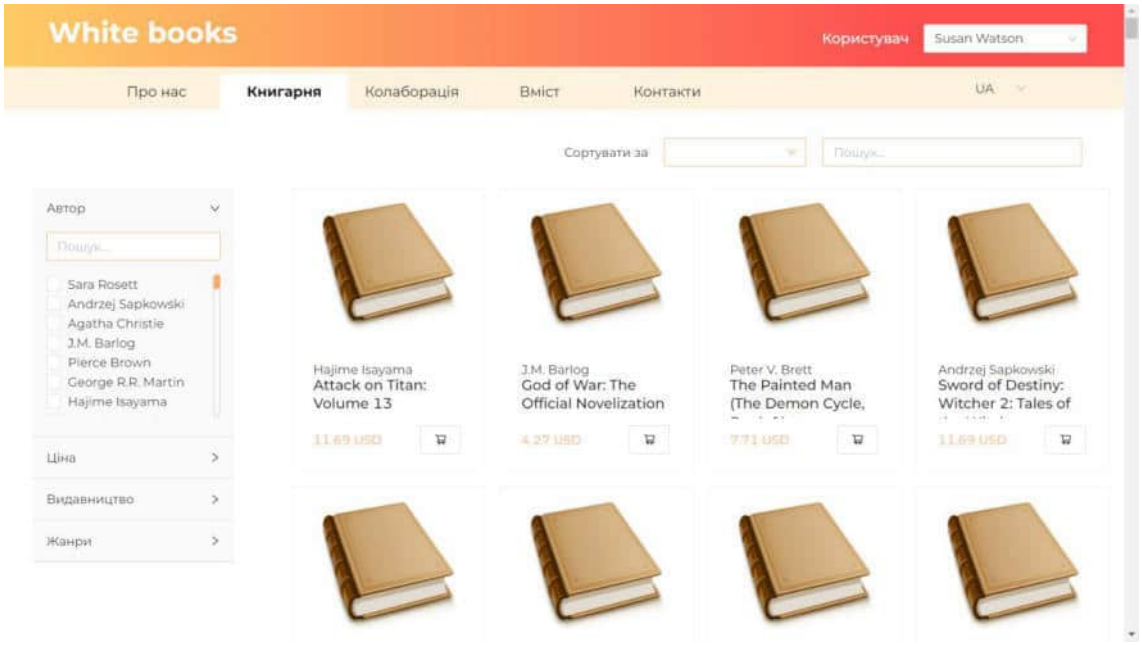

Рисунок 2.3 – Каталог товарів

б) перегляд деталей з описом до кожної книги (сторінка опису вибраної книги зображена на рисунку 2.4);

Рисунок 2.4 – Сторінка опису вибраної книги

в) перегляд відгуків на кожну книгу (секція відгуків зображена на рисунку 2.5);

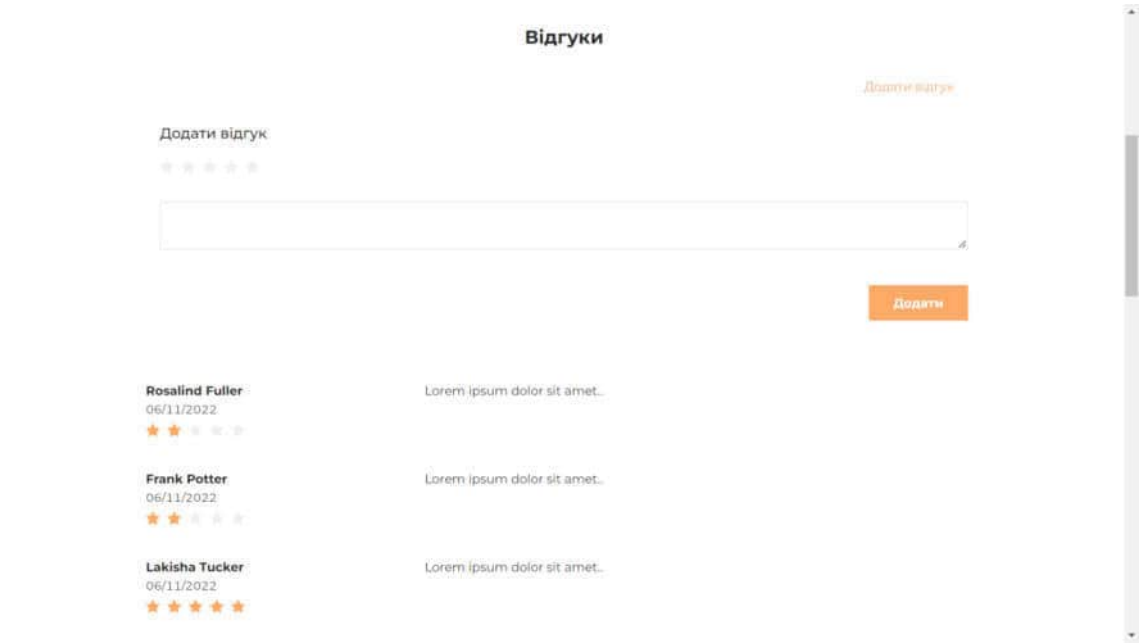

Рисунок 2.5 – Секція додавання нових відгуків на книгу

г) перегляд рекомендованих книг. Цей функціонал передбачає: рекомендації на основі книги та для конкретного користувача. Рекомендації на основі книги можна переглянути у секції «Схожі товари», що розташована на сторінці опису книги під відгуками. Ця секція зображена на рисунку 2.6. У цій секції відображаються книги, які мають високий коефіцієнт подібності з книгою, сторінка опису якої відкрита. Секція «Схожі товари» складається з двох частин, які демонструють пропозиції, сформовані рекомендаційними системи на основі алгоритму колаборативної фільтрації та на основі вмісту відповідно.

Рекомендації, які система сформувала персонально для користувача, можна переглянути на вкладках «Колаборація» та «Вміст». Вони відображають пропозиції, сформовані рекомендаційною системною на основі алгоритму колаборативної фільтрації та рекомендаційною системою на основі вмісту відповідно. Ці вкладки виглядають так само, як і каталог книги, та мають ті самі можливості.

г) додавання відгуку на книгу. Кожен користувач має можливість оцінити кожну книгу за п’ятибальною системою та додати на неї відгук, за умови, що він ще цього не робив. Це можна зробити у секції «Відгуки» на сторінці опису книги (рисунок 2.5).

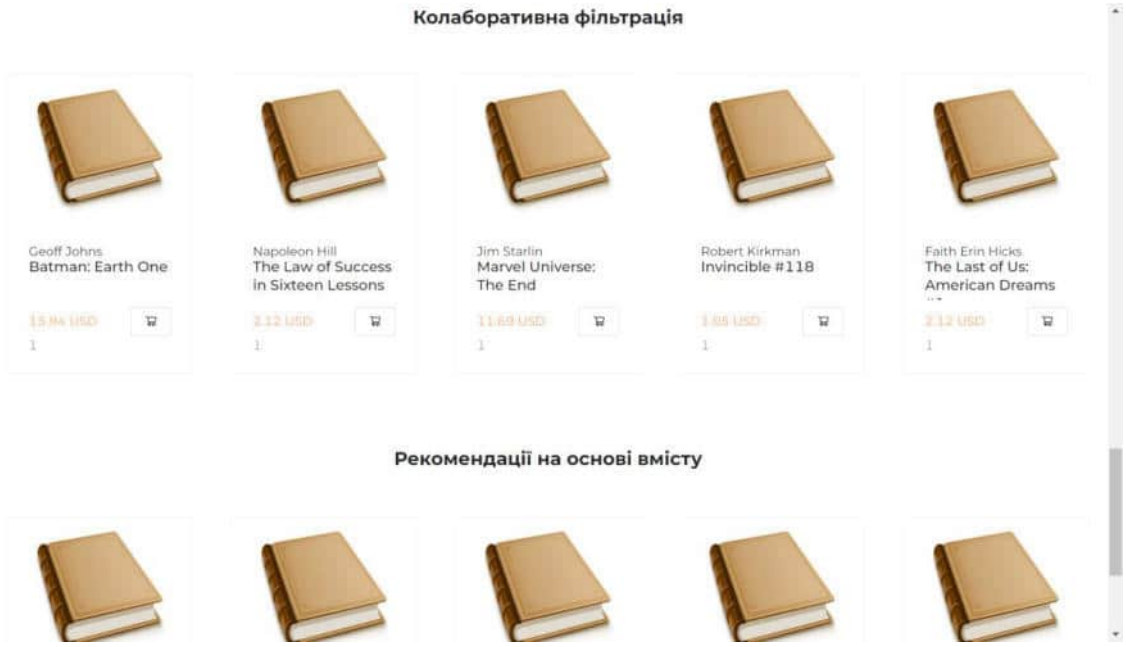

Рисунок 2.6 – Секція «Схожі товари» на сторінці з деталями вибраної книги

2.3 Особливості бази даних

Для реалізації цього додатку було використано базу даних з нереляційною моделлю, а системою керування базами даних вибрано MongoDB.

MongoDB дає можливість зберігати дані у неструктурованому форматі, а також, завдяки вкладеності документів, зменшити кількість необхідних операцій об’єднання (join) для швидшого пошуку потрібних даних у базі. Для ефективнішої роботи можна використати індекси [11].

Модель «сутність-зв’язок» в нотації Чена (Chen notation) [14] для бази даних реалізованої аплікації зображена на рисунку 2.7.

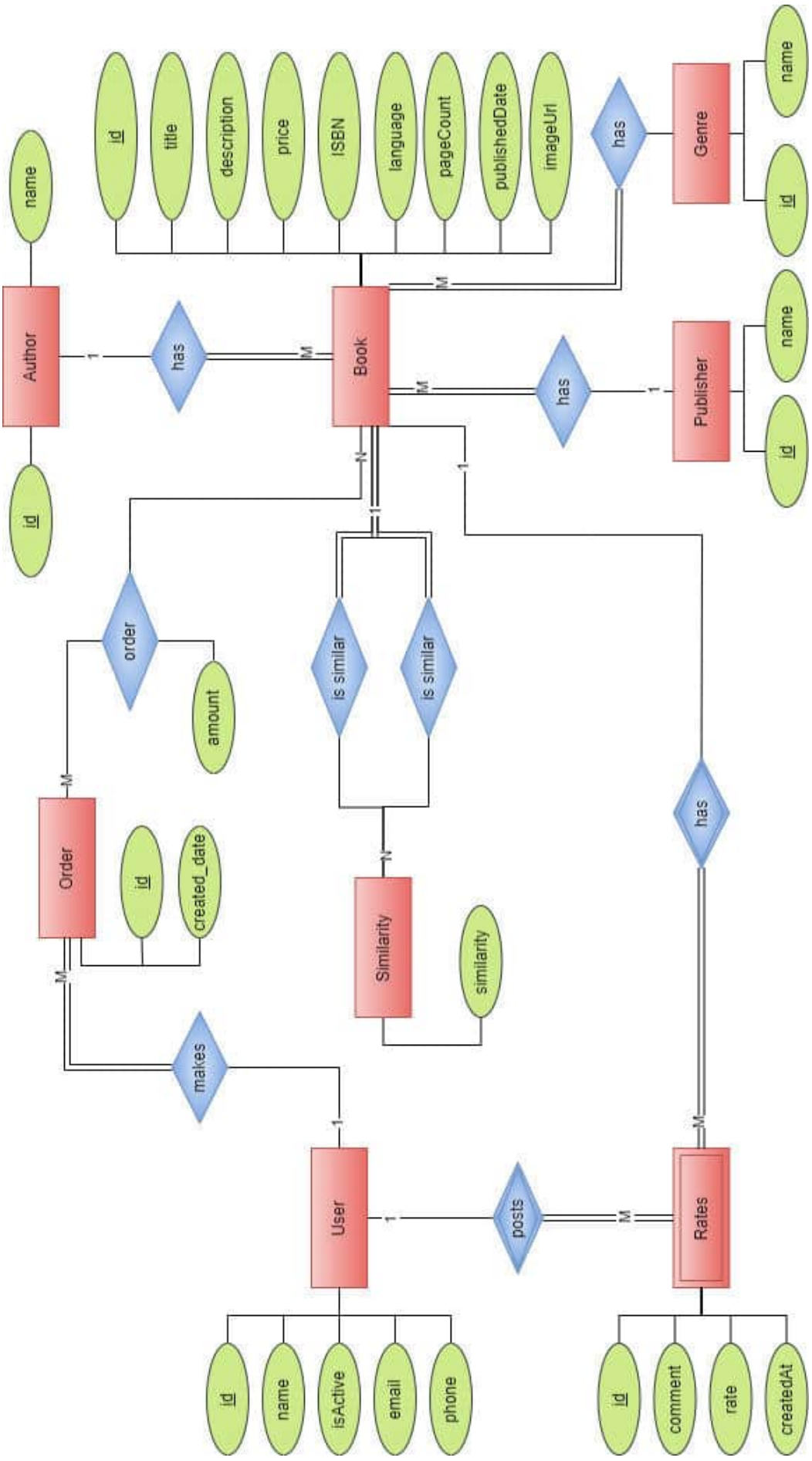

Рисунок 2.7 – Модель «сутність-зв’язок»

У таблиці 2.1 поданий детальний опис реалізації моделі «сутність-зв’язок» у MongoDB. Варто зауважити, що за замовчуванням MongoDB створює для документів поле ідентифікатора «_id» типу ObjectId. Опис полів у таблиці є у формі («назва поля у базі даних», «тип»).

Таблиця 2.1 – Реалізація моделі «сутність-зв’язок» у MongoDB

book
{
"_id": "",
"title": "",
"author": {
"_id": "",
"name": ""
},
"price": 0,
"description": "",
"publisher": {
"_id": "",
"name": ""
},
"ISBN": "",
"language": "",
"genres": [
{
"_id": "",
"name": ""
}
],
"pageCount": 0,
"publishedDate": "",
"imageUrl": ""
}
	Основною сутністю є книга, вона зберігається в колекції books. Документ, що її описує, складається з таких полів: ідентифікатор (_id, ObjectId), назва (title, стрічка), ціна (price, число з плаваючою комою), анотація (description, стрічка), номер ISBN (стрічка), мова видання (language, стрічка), кількість сторінок (pageCount, ціле число), дата публікації (publishedDate, дата) та посилання на картинку (imageUrl, стрічка).
	Також є поля, що мають деякі особливості, а саме поля автор (author), видавництво (publishing) та жанри (genres).
	У реляційній базі даних прийнято виносити такі сутності в окремі таблиці в рамках нормалізації для того, щоб уникнути повторення великої кількості даних, а отже, й зменшити необхідну кількість пам’яті для збереження цих даних. Таке рішення спричиняє потребу робити операції об’єднання таблиць (join), щоб потім зібрати всі необхідні дані для роботи з ними.
	У випадку з MongoDB прийнято об’єднувати всі необхідні дані в одну колекцію, щоб уникати операцій об’єднання і тим самим зменшувати

<table>
<tr><td></td><td>час, необхідний для виконання запиту. Звісно, внаслідок документи потребують більше пам’яті для збереження, але це не є пріоритетом нереляційних баз даних. Варто також зауважити, що максимальна кількість пам’яті виділена на один документ в MongoDB, – це 16 мегабайт [5], що варто враховувати при створенні дизайну бази даних.
Отже, враховуючи описані вище причини, було прийнято рішення включити сутності автор, видавництво та жанри в сутність книга. Включені були не лише їх назви (поля name), які потрібні просто для відображення інформації, але повністю об’єкти з ідентифікаторами (_id) для того, щоб зробити можливими посилання на конкретного автора/видавництво/жанр зі сторінки опису книги.</td></tr>
<tr><td>author
{
"_id": "",
"name": ""
}</td><td rowspan="3">Незважаючи на те, що було прийнято рішення включити сутності автор, видавництво та жанри у сутність книга, вони також були винесені в окремі колекції. Це було зробленого, оскільки додаток, що розробляється, – це інтернет-магазин і фільтрація книг за значеннями цих сутностей є одною з його ключових функціональних задач. Тому значно зручніше мати всі доступні значення для фільтрації в одному місці, ніж щоразу шукати унікальні значення, перебираючи всю колекцію книги.</td></tr>
<tr><td>publisher
{
"_id": "",
"name": ""
}</td></tr>
<tr><td>genre
{
"_id": "",
"name": ""
}</td></tr>
</table>

	Опис цих сутностей є однаковим, для нього використовують поля ідентифікатор (_id, ObjectId) та назва (name, стрічка).

user
{
"_id": "",
"name": "",
"email": "",
"phone": ""
}
	Для зберігання сутності користувачів була створена колекція user. Кожен документ описує користувача за допомогою таких полів: ідентифікатор (_id, ObjectId), ім'я (name, стрічка), адреса електронної пошти (email, стрічка) та номер телефону (phone, стрічка).

rate
{
"_id": "",
"bookId": "",
"reviews": [
{
"user": {
"_id": "",
"name": ""
},
"rate": 0,
"comment": "",
"createdAt": ""
}
]
}
	Очевидно, що сутність оцінок книги не може існувати без книги, а тому цілком логічним рішенням був би запис оцінок як вкладений об'єкт у сутність книги. Тим не менше, у зв'язку з тим, що одною з основних ідей додатку, що розроблявся, була реалізація рекомендаційних систем, які залежать від наявних оцінок, для спрощення роботи з ними було прийнято рішення винести оцінки в окрему колекцію. Звісно, негативним наслідком є потреба в операції об'єднання (join) при формуванні повної інформації про книгу, але він не є вагомим у контексті цієї роботи.
	Сутність оцінки описана такими полями: ідентифікатор оцінки (_id, ObjectId), ідентифікатор книги (bookId, ObjectId), яку ця оцінка описує, та масив відгуків (reviews, масив об'єктів). У свою чергу кожен з відгуків має такі поля: об'єкт, що описує користувача (_id, ObjectId та name, стрічка), що поставив оцінку, оцінка (rate, ціле число), яку він поставив,

	коментар до оцінки (comment, стрічка) та дата (createdAt, дата), коли ця оцінка була доданою. Було прийнято рішення включити сутність користувача у сутність відгуку у вигляді об'єкта, щоб уникнути потреби в операції об'єднання між документами різних колекцій.

similarity
{
"_id": "",
"book1Id": "",
"book2Id": "",
"similarity": 0
}
	Колекція similarity (подібність) була створена спеціально для рекомендаційної системи, що використовує алгоритм колаборативної фільтрації, для того, щоб зберігати коефіцієнти подібності між книгами у формі розрідженої матриці.
	Кожен документ цієї колекції має такі поля: власний ідентифікатор (_id, ObjectId), ідентифікатори двох книг (book1Id та book2Id, обидва типу ObjectId), подібність між якими зберігається у цьому документі, та саме значення подібності (similarity, число з плаваючою комою).

order
{
"_id": "",
"user": {
"_id": "",
"name": ""
},
"createdAt": "",
"books": [
{
"_id": "",
"title": "",
"price": "",
"amount": ""
}
]
}
	Колекція замовлень (orders) була реалізована відповідно до найкращих практик включення об'єктів у документ, щоб уникнути операції об'єднання між документами різних колекцій. Саме тому в документ кожного замовлення включені об'єкти користувача та книг, а не лише їхні ідентифікатори.
	Документ сутності замовлення складається з таких полів: ідентифікатор (_id, ObjectId), об'єкт користувача (user), дата створення

	замовлення (createdAt, дата), масив книг (books), які належать до цього замовлення. Об’єкт користувача має такі поля: ідентифікатор (_id, ObjectId) та ім’я (name, стрічка). Поле «книги» (books) містить масив об’єктів, що описують книги, що були куплені у цьому замовлені. Об’єкт книги має такі поля: ідентифікатор (_id, ObjectId), назва (title, стрічка), ціна (price, число з плаваючою комою) та кількість примірників (amount, ціле число).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКОМЕНТАЦІЙНИХ СИСТЕМ

3.1 Реалізація алгоритму колаборативної фільтрації

Результатом реалізації рекомендаційної системи на основі алгоритму колаборативної фільтрації є прикладний програмний інтерфейс (API).

На рисунку 3.1 зображено перелік методів цього прикладного програмного інтерфейсу, які можна переглянути, використовуючи Swagger. Swagger – це інструмент, що дозволяє автоматично створювати інтерактивну документацію прикладного програмного інтерфейсу [18].

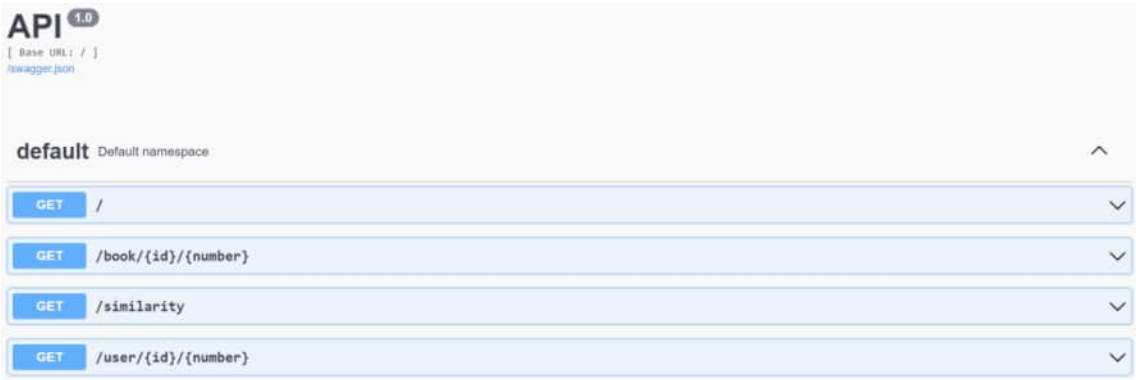

Рисунок 3.1 – Перелік методів у Swagger

Отже, були реалізовані такі методи:

а) отримання загальної інформації про прикладний програмний інтерфейс, створений для тестування (метод за шляхом «/»);

б) оновлення таблиці подібності між товарами, який необхідний для того, щоб вона не втрачала актуальності після додавання нових відгуків («`/similarity`»);

в) обчислення рекомендацій на основі конкретної книги («`/book/{id}/{number}`»);

г) обчислення індивідуальних пропозицій для конкретного користувача («`/user/{id}/{number}`»).

Для реалізації цього прикладного програмного інтерфейсу були використані такі технології: мова програмування Python та вебфреймфорк Flask.

Рекомендаційна система на основі алгоритму колаборативної фільтрації була реалізована відповідно до покрокового алгоритму, описаного у підрозділі 1.2. Обчислення проводяться на основі даних, що зберігаються в колекції rates, детальний опис якої подано у таблиці 2.1. Приклад одного з документів цієї колекції наведено на рисунку 3.2.

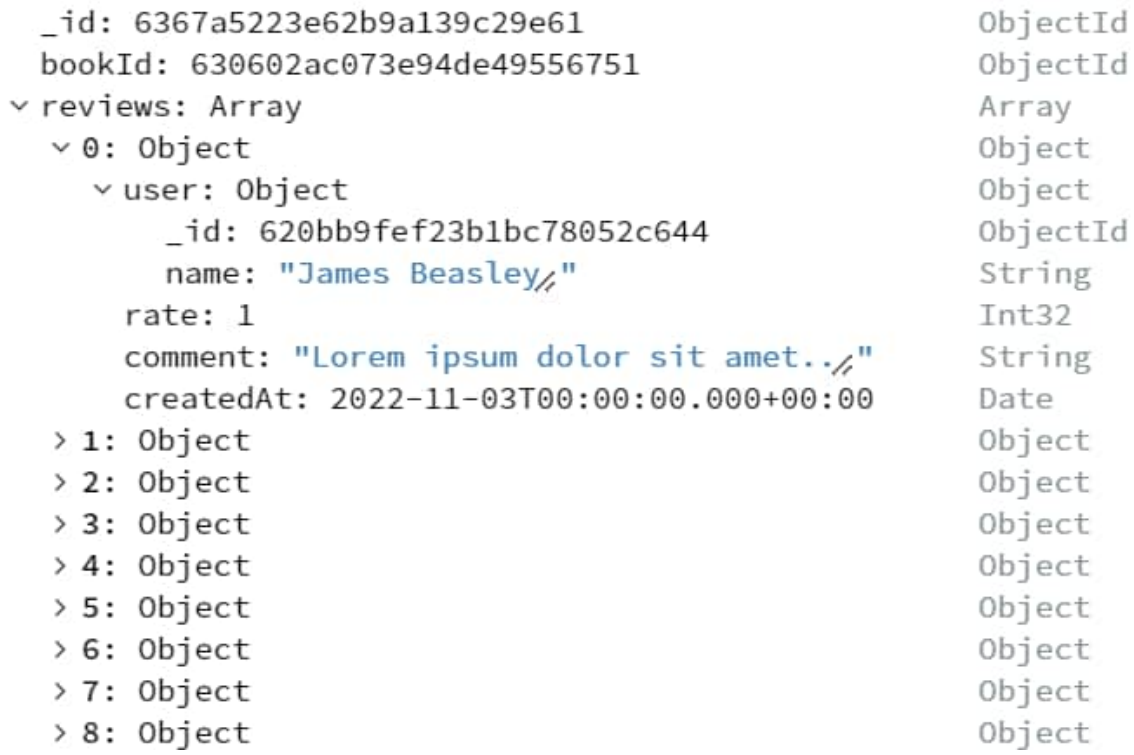

Рисунок 3.2 – Приклад документа з колекції rates

Для кожної книги, що характеризується ідентифікатором bookId є список відгуків, поля яких: об’єкт, який описує користувача, що створив відгук, оцінка, коментар та дата створення. Остання властивість необхідна для того, щоб оновлювати таблицю подібності лише перераховуючи значення для тих книг, оцінки для яких були додані нещодавно (в межах дня або тижня), а не для всіх.

Проміжним результатом реалізації алгоритму, а саме кроку 3, є зберігання розрідженої матриці коефіцієнтів подібності між книгами. Код обчислення коефіцієнтів подібності між книгами наведений в додатку А. Для зберігання цієї матриці була створена колекція similarity. Приклад документа з цієї колекції зображений на рисунку 3.3.

```
_id: 637551747e24e498dbe474c6                    ObjectId
book1Id: 630602ac073e94de49556747                ObjectId
book2Id: 630602ac073e94de49556751                ObjectId
similarity: 0.582                                Double
```

Рисунок 3.3 – Приклад документа з колекції similarity

Варто зауважити, що оскільки $simil(x, y) = simil(y, x)$, то такий формат зберігання матриці коефіцієнтів подібності є зручним для пошуку id книги як серед значень поля book1Id, так і серед book2Id.

Прикладний програмний інтерфейс надає доступ до методів, що відповідають за формування пропозицій на основі певної книги, а також персональних рекомендацій для конкретного користувача. Код реалізації наведений в додатку Б.

На рисунку 3.4 зображений результат виклику методу для формування пропозицій на основі певної книги. Цьому методу необхідно передати два параметри: ідентифікатор книги, на основі якої система формує рекомендації, та кількість книг (елементів рекомендації).

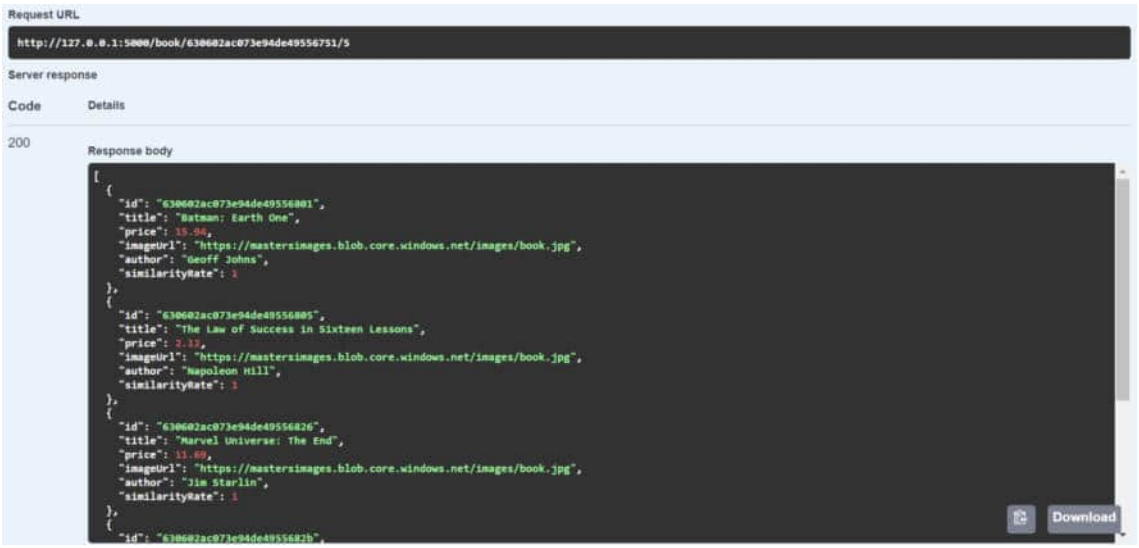

Рисунок 3.4 – Результат виклику методу формування рекомендацій на основі вибраної книги

Результат виконання цього методу використовується для наповнення першої половини секції «Схожі товари» на сторінці опису конкретної книги. Вигляд цієї секції на користувацькому інтерфейсі інтернет-магазину зображений на рисунку 3.5. Числове значення, подане під ціною, відповідає коефіцієнту подібності книги, яку рекомендують, з обраною.

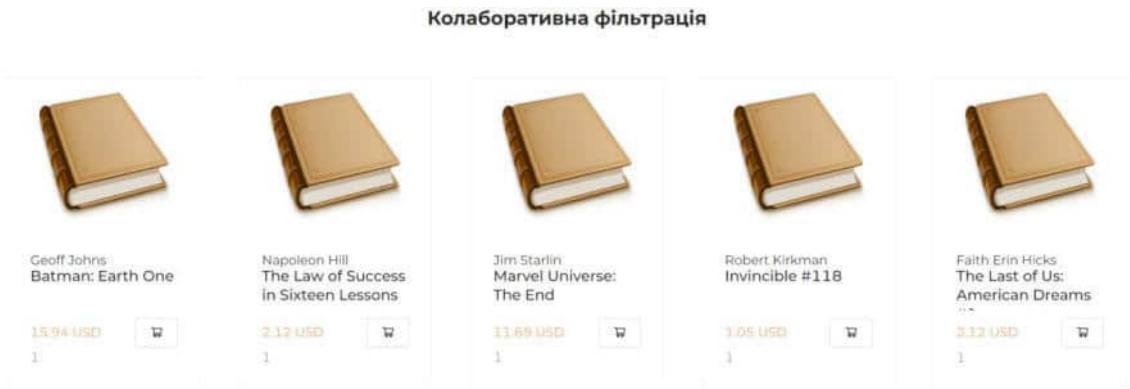

Рисунок 3.5 – Секція «Схожі товари», результат обчислень алгоритму колаборативної фільтрації

На рисунку 3.6 зображений результат виклику методу для формування персональних рекомендацій для конкретного користувача. Цьому методу необхідно передати два параметри: ідентифікатор користувача, для якого система формує рекомендації, та кількість книг (елементів рекомендації).

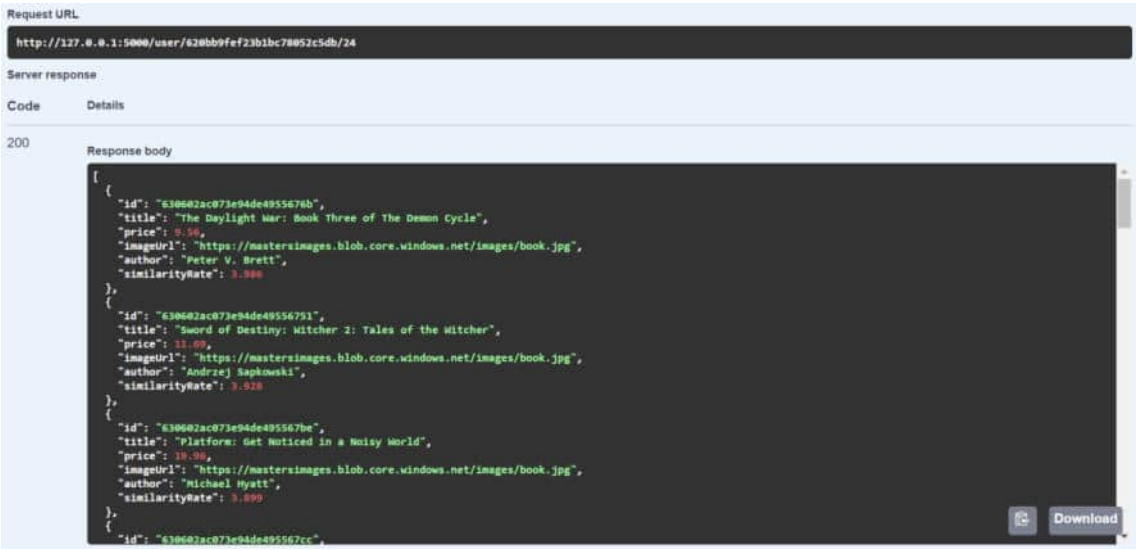

Рисунок 3.6 – Результат виклику методу формування індивідуальних рекомендацій для конкретного користувача

Результат виконання цього методу використовується для створення сторінки «Колаборація», на якій користувач може переглянути персональні пропозиції, які для нього сформувала рекомендаційна система. На рисунку 3.7 зображена ця сторінка. Числове значення під ціною книги відображає оцінку, яку система передбачила від поточного користувача цій книзі.

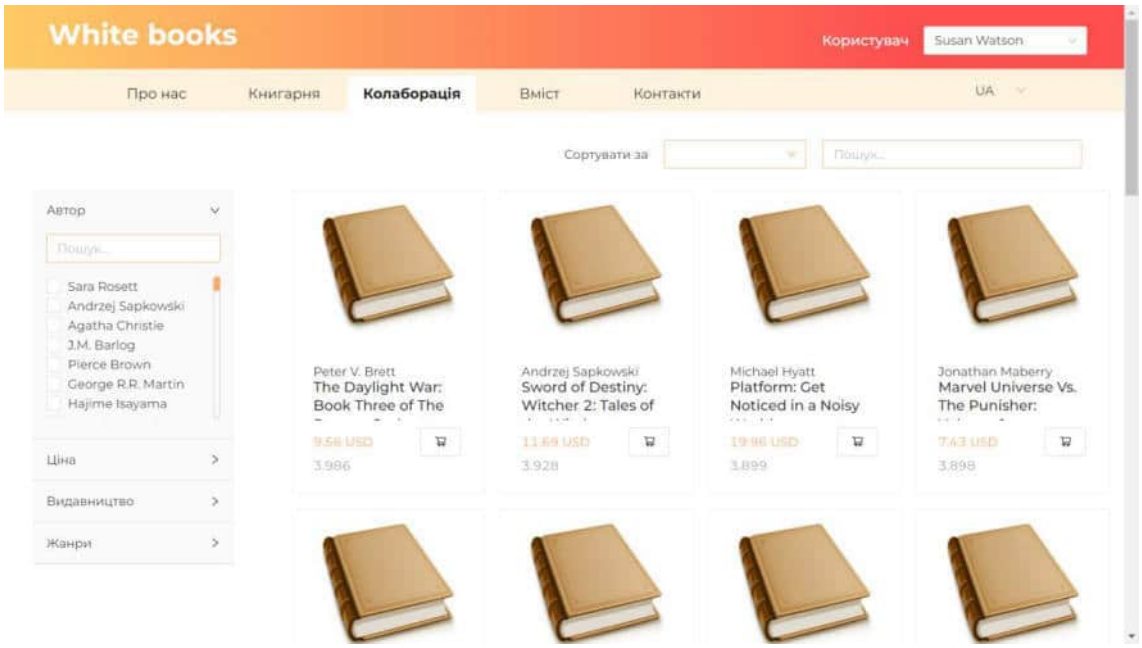

Рисунок 3.7 – Сторінка персональних пропозицій для користувача «Колаборація», результат обчислень алгоритму колаборативної фільтрації

3.2 Реалізація рекомендаційної системи на основі вмісту

Результатом реалізації рекомендаційної системи на основі вмісту (content-based recommendation system) є прикладний програмний інтерфейс (API).

На рисунку 3.8 зображено перелік методів цього прикладного програмного інтерфейсу, які можна переглянути, використовуючи Swagger.

Отже, були реалізовані такі методи:

а) отримання загальної інформації про прикладний програмний інтерфейс, створений для тестування (метод за шляхом «/»);

б) обчислення рекомендацій на основі конкретної книги («`/book/{id}/{number}`»);

в) обчислення індивідуальних пропозицій для конкретного користувача («/user/{id}»).

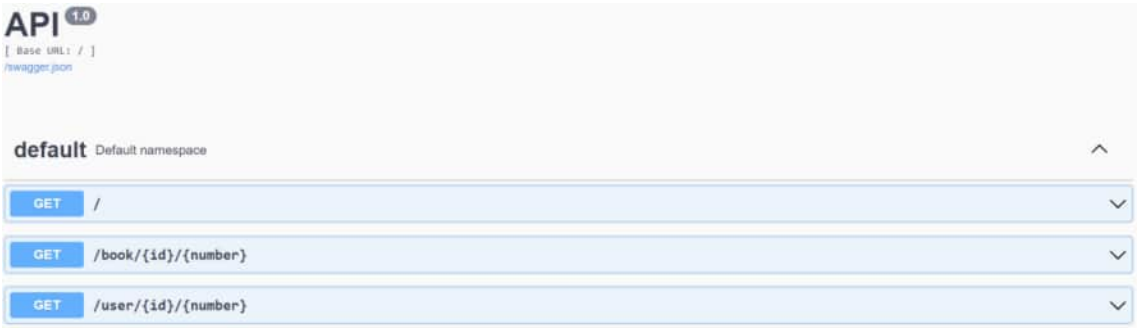

Рисунок 3.8 – Перелік методів у Swagger

Для реалізації цього прикладного програмного інтерфейсу були використані ті ж технології, що і у випадку рекомендаційної системи на основі алгоритму колаборативної фільтрації, а саме: мова програмування Python та вебфреймфорк Flask.

Рекомендаційна система на основі вмісту була реалізована відповідно до алгоритму, описаного у підрозділі 1.3. Як вже було зазначено вище, ця рекомендаційна система вимагає не лише дані про відгуки користувачів, але й дані про самі товари, у цьому випадку книги. Тому для проведення обчислень необхідні записи з двох колекцій бази даних: книги (books), що містить властивості книг, та відгуки (rates) від користувачів. Приклад одного з документів колекції rates був наведений на рисунку 3.2, а приклад документа з колекції books наведений на рисунку 3.9.

Детальніше документи колекції books описані в таблиці 2.1.

Отже, для обчислення рекомендацій необхідно сформувати модель, основою якої є властивості книг. Для цього потрібно провести попередню обробку даних, щоб вони були зрозумілими для машин.

Передусім вибрати властивості, які варто враховувати у модель. З огляду на особливості товару (книги) доцільно використовувати такі його характеристики: назва, анотація, автор та жанри. Мова видання та посилання на зображення

обкладинки не несуть ніякої інформації у контексті змістового наповнення книги, тому не використовуються для побудови моделі.

Далі необхідно вирішити, чи всі властивості впливають на модель однаково, чи варто акцентувати на якійсь із них, тобто надати властивостям різних ваг. У контексті цієї роботи, всі властивості мають однаковий вплив на модель.

Рисунок 3.9 – Приклад документа з колекції books

Очевидно, що всі обрані властивості мають текстовий формат, тобто для подальшої роботи їх необхідно перетворити у числовий. Для цього було виконано декілька кроків.

Перш за все, оскільки жанри кожної книги в колекції books записані масивом об’єктів, то їх було перетворено у стрічку, щоб з ними можна було працювати як з текстом. Далі назва книги, її анотація та перелік жанрів були об’єднані в одну стрічку, щоб методи обробки природньої мови могли працювати з нею як із однією збірною текстовою властивістю.

Далі було виконано: переведення всіх слів у нижній регістр, видалення знаків пунктуації та стоп-слів, виділення частини слова, що відповідає за лексичне значення (stemming). Детальніше ці кроки були описані в 1.3. Далі на основі перетворених даних було сформовано матрицю, що складається з векторів. Ці вектори – це перетворені описи кожної з книг алгоритмом TF-IDF.

Водночас авторів книги теж було опрацьовано. Оскільки існує декілька книг, що мають одного і того ж автора, то імена авторів було перетворено в числа із використанням алгоритму кодування міток, описаного в 1.3.

В результаті було отримано матрицю, що описує книгу в загальному, та вектор закодованих авторів. Об’єднавши ці властивості, було отримано таблицю значень, кожен рядок якої складається з важливості слів у контексті загального опису книги та номеру автора. Схематично ця матриця зображена у таблиці 2.2. Код реалізації попередньої обробки даних наведений в додатку В.

Таблиця 2.2 – Схематичне зображення матриці опрацьованих властивостей книг

	Важливість слів				Код автора
	слово 1	слово 2	слово 3	…	
Книга 1	0	0.4	0.78	…	1
Книга 2	0.35	0	0	…	2
Книга 3	0.67	0.72	0	…	1
…	…	…	…	…	…
Книга n	0.28	0	0.58	…	3

Для обчислення рекомендацій на основі книги цієї матриці є достатньо. Завдання полягає у порівнянні вектора властивостей поточної книги з усіма іншими та виборі книг з найбільшим коефіцієнтом подібності. У цій роботі для обчислень було використано косинусну міру подібності.

Обчислення персональних рекомендацій для конкретного користувача відбуваються у дещо інший спосіб, оскільки зводиться до розв’язання задачі

класифікації книг. Оскільки можливими значеннями оцінок є 1, 2, 3, 4 або 5, то кожне з цих значень буде відігравати роль мітки класу, а книги, оцінені одним числовим значенням, будуть утворювати клас.

Завдання полягає у класифікації книг, що ще не мають міток, тобто тих книг, яким цей користувач ще не поставив оцінки. Для цього використовується вищеописана таблиця, доповнена оцінками (мітками класу), якщо вони існують. Схематично ця модель зображена у таблиці 2.3. У цій таблиці є книги, що мають оцінки (мітки). Вони формують тренувальну вибірку. А неоцінені книги необхідно зарахувати до одного з класів, тобто поставити їм оцінки.

Таблиця 2.3 – Схематичне зображення вхідних даних для класифікації

	Властивості книг					Мітка
	Важливість слів				Закодований автор	Оцінка від користувача
	Слово 1	Слово 2	Слово 3	…		
Книга 1	0	0.4	0.78	…	1	2
Книга 2	0.35	0	0	…	2	3
Книга 3	0.67	0.72	0	…	1	5
Книга 3	0	0.33	0	…	3	5
…	…	…	…	…	…	…
Книга n-1	0.12	0.43	0	…	14	??
Книга n	0.28	0	0.58	…	256	??

Рекомендаційна система на основі вмісту вирішує цю задачу, використовуючи наївний класифікатор Баєса, описаний у 1.3. Для цього було використано бібліотеку scikit-learn мови програмування Python [10].

Як результат, рекомендаційна система формує індивідуальні пропозиції для конкретного користувача з товарів, що мають найвищі передбачені оцінки.

Прикладний програмний інтерфейс, що є результатом реалізації цієї рекомендаційної системи, надає доступ до методів, що відповідають за формування

пропозицій на основі певної книги, а також персональних рекомендацій для конкретного користувача. Код реалізації наведений в додатку Г.

На рисунку 3.10 зображений результат виклику методу для формування пропозицій на основі певної книги. Цьому методу необхідно передати два параметри: ідентифікатор книги, на основі якої система формує рекомендації, та кількість книг (елементів рекомендації).

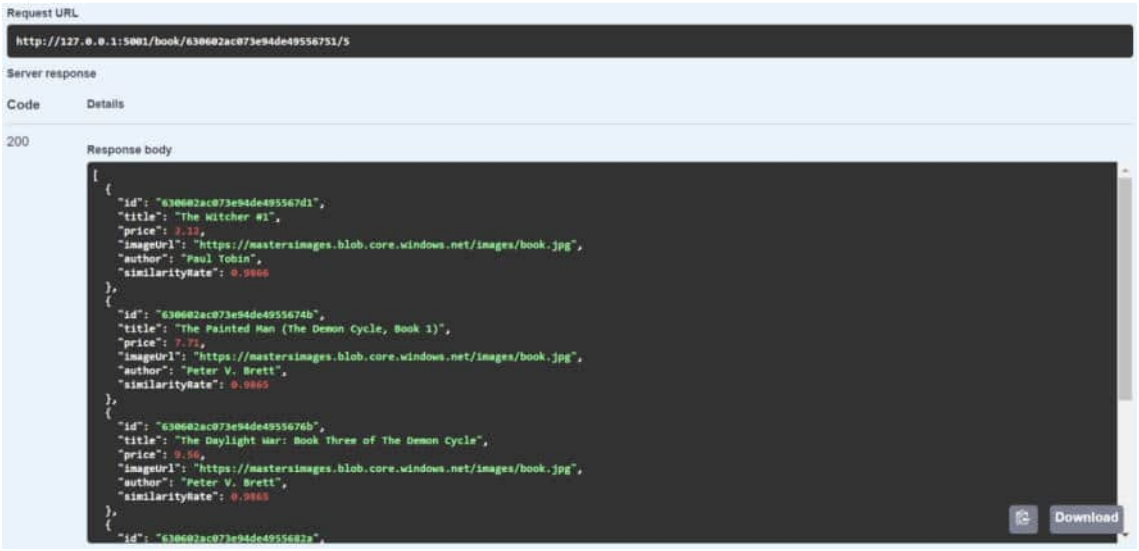

Рисунок 3.10 – Результат виклику методу формування рекомендацій на основі вибраної книги

Результат виконання цього методу використовується для наповнення другої половини секції «Схожі товари» на сторінці деталей конкретної книги. Вигляд цієї секції на користувацькому інтерфейсі інтернет-магазину зображений на рисунку 3.11. Числове значення, подане під ціною, відповідає коефіцієнту подібності книги, яку рекомендують, з обраною.

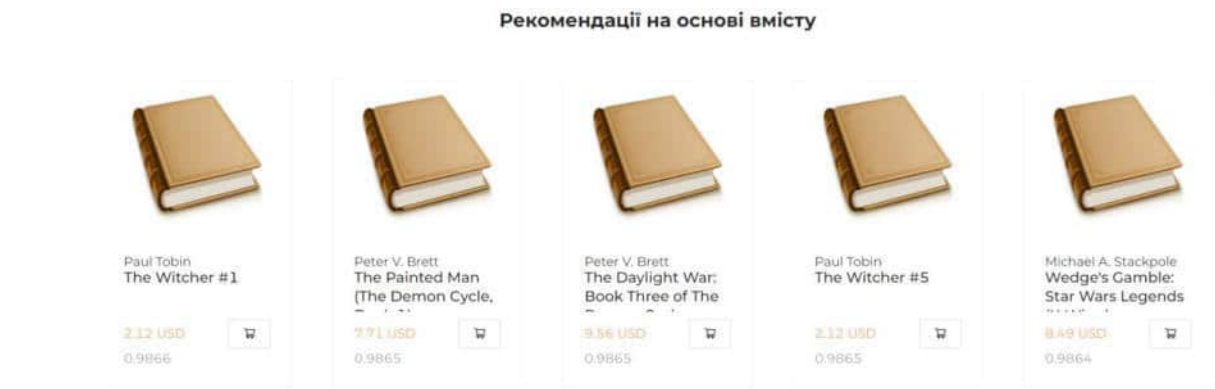

Рисунок 3.11 – Секція «Схожі товари», результат обчислень рекомендаційної системи на основі вмісту

На рисунку 3.12 зображений результат виклику методу для формування персональних рекомендацій для конкретного користувача. Цьому методу необхідно передати два параметри: ідентифікатор користувача, для якого система формує рекомендації, та кількість книг (елементів рекомендації).

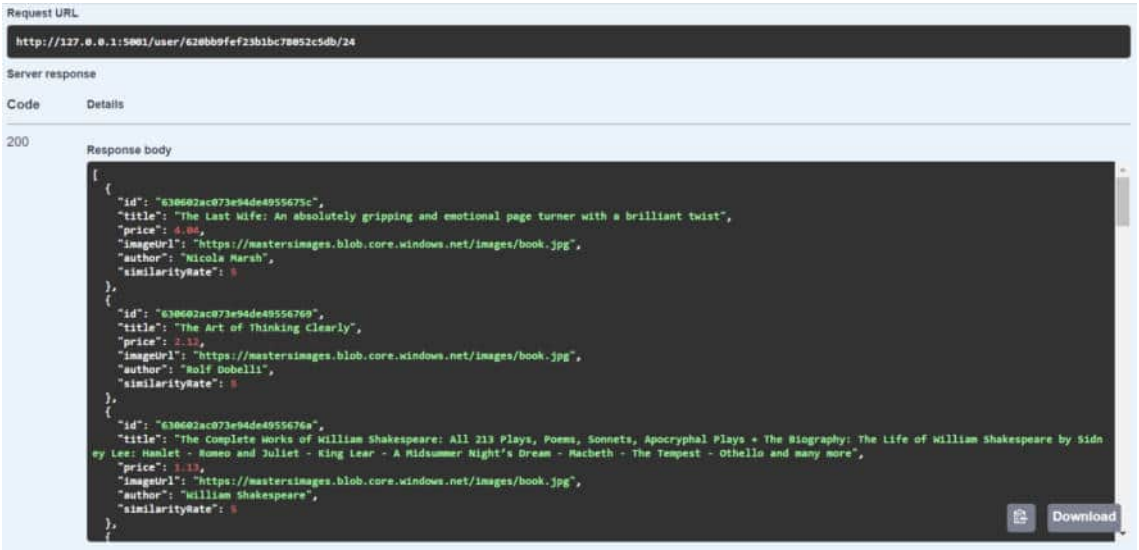

Рисунок 3.12 – Результат виклику методу формування індивідуальних рекомендацій для конкретного користувача

Результат виконання цього методу використовується для створення сторінки «Вміст», на якій користувач може переглянути персональні для себе пропозиції, які йому сформувала ця рекомендаційна система. На рисунку 3.13 зображена ця сторінка. Числове значення під ціною книги відображає оцінку, яку система передбачила від поточного користувача цій книзі.

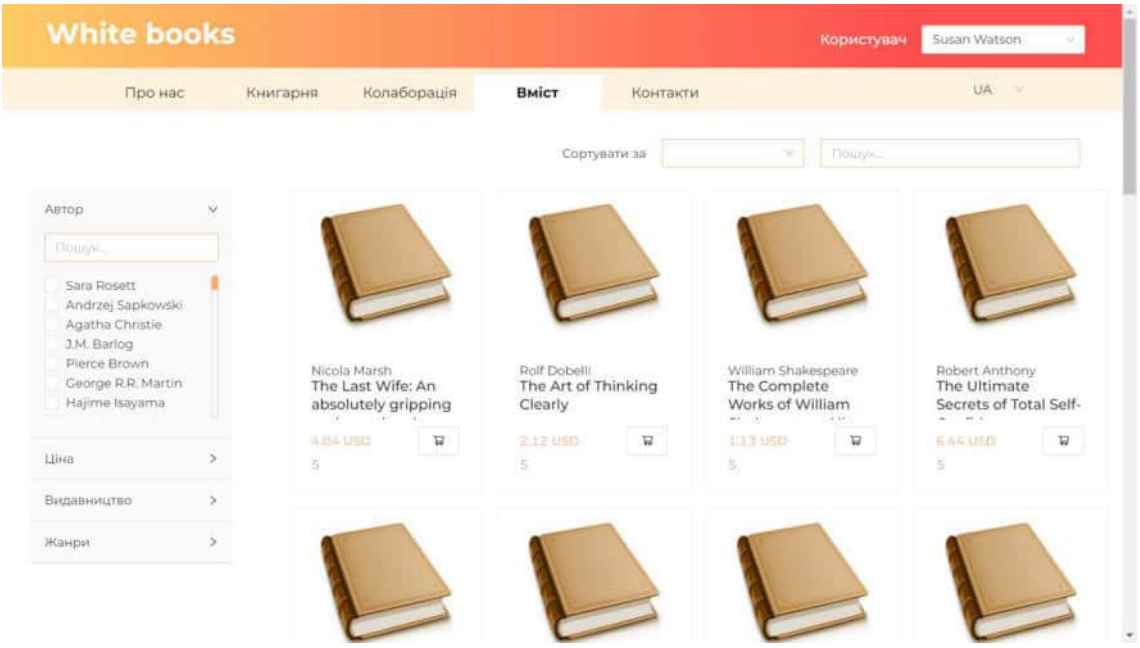

Рисунок 3.13 – Сторінка персональних пропозицій для користувача «Вміст», результат обчислень рекомендаційної системи на основі вмісту

3.3 Результати обчислень реалізованих рекомендаційних систем

На рисунках 3.5, 3.7, 3.11 та 3.13 зображені результати обчислень реалізованих рекомендаційних систем. Як і очікувалося, вони видали різні результати. Не можна стверджувати, що одні з них є кращі, а інші гірші, бо остаточне рішення приймає користувач суб’єктивно, але можна зауважити деякі закономірності.

Прикладом товару, на основі якого формувалися пропозиції, було вибрано книгу Анджея Сапковського «Sword of Destiny: Witcher 2: Tales of the Witcher». Серед п’яти найближчих пропозицій, які сформувала рекомендаційна система на основі вмісту, є дві книги з тієї ж тематики («The Witcher #1» та «The Witcher #5»). Той факт, що вони схожі за описом з обраною є очевидним, але як інтерпретувати такий результат може вирішити лише користувач. Якщо він зацікавлений у цій тематиці, рекомендовані книги будуть для нього актуальними, якщо не зацікавлений, то він сприйме їх як очевидну рекомендацію, якою не буде задоволений.

Водночас рекомендаційна система на основі колаборативної фільтрації сформувала зовсім інші пропозиції. Проте вони також можуть бути не очевидними

для користувача, бо головною ідеєю цієї колаборативної фільтрації є припущення, що користувачі, що мали схожі вподобання у минулому, матимуть їх і в майбутньому [7], але це не обов’язково так. Було продемонстровано, що результати обчислень рекомендаційної системи на основі вмісту можна пояснити більш обґрунтовано, ніж результати обчислень рекомендаційної системи на основі алгоритму колаборативної фільтрації.

Перевагою реалізованого додатку є те, що він дає можливість переглянути пропозиції від обох рекомендаційних систем. Однак найкращим способом формування рекомендацій є використання гібридного підходу, що дозволяє використовувати переваги обох методів та нейтралізувати недоліки. Особливо актуальним такий метод буде для бізнесу, який має орієнтуватися конкретно на свою цільову аудиторію.

ВИСНОВКИ

У результаті виконання роботи було досліджено різні підходи до побудови рекомендаційних систем, вивчено алгоритми, що лежать в їх основі, та окреслено їх переваги та недоліки. Особливу увагу приділено алгоритму колаборативної фільтрації та алгоритму формування рекомендацій на основі вмісту. Розглянуто та використано різні метрики, які застосовуються для обчислення коефіцієнтів подібності між товарами. Досліджено загальні підходи обробки природньої мови та застосування підходів розв’язання задачі класифікації до формування переліку рекомендацій.

Розроблено клієнт-серверний додаток інтернет-магазину та дві рекомендаційні системи: на основі алгоритму колаборативної фільтрації та на основі вмісту, що були інтегровані у цей додаток. Ця реалізація передбачала підбір відповідних технологій, вивчення різних моделей бази даних та розробку оптимальної архітектури додатку. Клієнтська частина інтернет-магазину реалізована з використанням HTML, CSS, React.js та Ant Design, а серверна – вебфреймфорку ASP.Net Core, база даних – NoSQL, системою керування бази даних є MongoDB. Рекомендаційні системи реалізовані засобами мови програмування Python та вебфреймфорку Flask.

При побудові рекомендаційних систем для обчислення коефіцієнтів подібності між товарами використано косинусу міру подібності та коефіцієнт кореляції Пірсона, а для класифікації – наївний класифікатор Баєса.

Пропозиції, сформовані реалізованими рекомендаційними системи, є різними, але жоден результат не можна назвати помилковими. Головне завдання цих систем – зацікавити користувача у якихось товарах. Оскільки, обидва набори пропозицій є обґрунтованими, то їх можна вважати правильними. Остаточне рішення є суб’єктивним та його приймає користувач.

У контексті переваг та недоліків розглянутих алгоритмів можна впевнено стверджувати, що, на жаль, жоден із них не є достатньо ефективним, бо не вирішує проблеми холодного старту.

З результатів зрозуміло, що найкращим способом формування рекомендацій є використання гібридного підходу, щоб скористатися перевагами обох методів одночасно і, відповідно, максимально нейтралізувати недоліки. Проте навіть поєднання підходів не є достатнім, адже ключовим елементом формування рекомендацій є саме інформація від користувача, тобто дані про його покупки та вподобання. Без них жодна рекомендаційна система не буде ефективною.

Отже, ефективний алгоритм формування рекомендацій є складнішим, ніж здається, та складається з декількох кроків. Перш за все, необхідно спонукати користувача створити власний обліковий запис на платформі, щоб можна було відслідковувати його дії, тобто покупки та вподобання. Оскільки на початку про нього немає ніякої інформації, необхідно обмежитися рекомендаціями, що складаються з найпопулярніших товарів або товарів з найвищим рейтингом. На цьому етапі ключовими є кроки користувача: чи здійснив він покупку або чи залишив відгук на якийсь товар?

Якщо користувач не виявить бажання «спілкуватися» з системою, тобто дати їй якусь інформацію, то, на жаль, система теж не може йому нічого запропонувати.

Якщо користувач вирішить бути активним, то необхідно, що система використовувала всю наявну інформацію для свої роботи, тобто формувала рекомендації на основі даних і про покупки, і про відгуки, а не обмежувалася лише чимось одним. У такому випадку можна отримати багатогранніші рекомендації, адже відгуки користувача на товар не завжди збігаються з його покупками на цій платформі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сеньо П.С. Теорія ймовірностей та математична статистика / П.С. Сеньо – Київ: «Знання», 2007. – С. 45-46/
2. Aakarsha Chugh ML | Label Encoding of datasets in Python / Aakarsha Chugh [Electronic resource]. – 2022. – Режим доступу: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/
3. Collaborative Filtering In Recommender Systems: Learn All You Need To Know [Electronic resource]. – 2021. – Режим доступу: https://www.iteratorshq.com/blog/collaborative-filtering-in-recommender-systems/
4. Dietmar Jannach Recommender Systems. An Introduction / Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich – New York: Cambridge University Press, 2011. – С. 51-79
5. Documents / MongoDB, Inc. [Electronic resource]. – 2022. – Режим доступу: https://www.mongodb.com/docs/manual/core/document/
6. Hunter Heidenreich Natural Language Processing: Count Vectorization with scikit-learn / Hunter Heidenreich [Electronic resource]. – 2018. – Режим доступу: https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e
7. Maitray Gadhavi Top 8 Reasons ASP.NET Core is the Best Framework for Web Application Development / Maitray Gadhavi [Electronic resource]. – 2021. – Режим доступу: https://radixweb.com/blog/8-reasons-asp-dot-net-core-is-best-framework
8. Mukesh Chaudhary TF-IDF Vectorizer scikit-learn / Mukesh Chaudhary [Electronic resource]. – 2018. – Режим доступу: https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
9. Mustafa Katipoğlu User-based vs Item-based Collaborative Filtering / Mustafa Katipoğlu [Electronic resource]. – 2020. – Режим доступу: https://medium.com/recommendation-systems/user-based-vs-item-based-collaborative-filtering-d40bb49c7060.

10. Naive Bayes / scikit-learn developers (BSD License) [Electronic resource]. – Режим доступу: https://scikit-learn.org/stable/modules/naive_bayes.html
11. NoSQL vs. SQL Databases / MongoDB, Inc. [Electronic resource]. – 2022. – Режим доступу: https://www.mongodb.com/nosql-explained/nosql-vs-sql
12. Oleksandr Hutsulyak / 10 Key Reasons Why You Should Use React for Web Development / Oleksandr Hutsulyak [Electronic resource]. – 2022. – Режим доступу: https://www.techmagic.co/blog/why-we-use-react-js-in-the-development/
13. Paslavska Y. Investigation and Implementation of the Item-Based Collaborative Filtering / Y. Paslavska // Міжнародна студентська наукова конференція з питань прикладної математики та комп'ютерних наук (МСНКПМК2022), 5-6 травня 2022 р. – Львів: 2022. – С. 33-35. Режим доступу: https://ami.lnu.edu.ua/wp-content/uploads/2022/05/ISSCAMCS-2022.pdf
14. Patrycja Dybka Chen Notation / Patrycja Dybka [Electronic resource]. – 2014. – Режим доступу: https://vertabelo.com/blog/chen-erd-notation/
15. Shuyu Luo Introduction to Recommender System / Shuyu Luo [Electronic resource]. – 2018. –Режим доступу: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26
16. Toby Segaran Programming Collective Intelligence / Toby Segaran. – New York: O'Reilly, 2007. – С. 27-46.
17. UML Use Case Diagram Tutorial / Lucid Software Inc. [Electronic resource]. – Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram
18. What Is Swagger? / SmartBear Software [Electronic resource]. – 2022. – Режим доступу: https://swagger.io/docs/specification/2-0/what-is-swagger/

ДОДАТОК А. РЕКОМЕНДАЦІЙНА СИСТЕМА НА ОСНОВІ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ: КОД ОБЧИСЛЕННЯ ПОДІБНОСТІ ТОВАРІВ

```
import math

def koef_pearson(book1_reviews, book2):
    users = []

    book2_reviews = change_scheme_user_rate(book2)

    for item in book1_reviews:
        if item in book2_reviews:
            users.append(item)

    n = len(users)
    if n == 0:
        return 0

    avg1 = sum([book1_reviews[it] for it in users]) / n
    avg2 = sum([book2_reviews[it] for it in users]) / n


    num = 0
    sq_sum1 = 0
    sq_sum2 = 0

    for it in users:
        num += (book1_reviews[it] - avg1) * (book2_reviews[it] - avg2)
        sq_sum1 += pow((book1_reviews[it] - avg1), 2)
        sq_sum2 += pow((book2_reviews[it] - avg2), 2)

    den = math.sqrt(sq_sum1 * sq_sum2)
    if den == 0:
        return 0

    r = ((num / den) + 1) / 2
    return round(r, 3)

def change_scheme_user_rate(rate):
    new_reviews = {}
    for review in rate['reviews']:
        new_reviews[review['user']['_id']] = review['rate']

    return new_reviews

def change_scheme_book_rate(rate):
    new_reviews = {}
    for review in rate['reviews']:
        new_reviews[rate['bookId']] = review['rate']
```

```
    return new_reviews

def get_recommended_items(rates, book_similatity):
    scores = {}
    totalSim = {}

    for (item, rating) in rates.items():
        for (similarity, item2) in book_similatity[item]:
            if item2 in rates:
                continue

            scores.setdefault(item2, 0)
            scores[item2] += similarity * rating

            totalSim.setdefault(item2, 0)
            totalSim[item2] += similarity

    rankings=[(round(score/totalSim[item], 3),item) for item,score in scores.items()
if totalSim[item] > 0]

    rankings.sort( )
    rankings.reverse( )
    return rankings
```

ДОДАТОК Б. РЕКОМЕНДАЦІЙНА СИСТЕМА НА ОСНОВІ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ: КОД РЕАЛІЗАЦІЇ ФОРМУВАННЯ ПРОПОЗИЦІЙ

```
from collaborative_dal import *
from collaborative_utils import *

def calculate_similarity():
    new_rates = get_rates_for_previous_day()
    requests = {}
    yesterday_datetime = datetime.datetime.utcnow() - datetime.timedelta(days = 1)
    yesterday = datetime.datetime(yesterday_datetime.year, yesterday_datetime.month,
yesterday_datetime.day)

    for rate in new_rates:
        for review in rate['reviews']:
            if (review['createdAt'] == yesterday):
                if (rate['bookId'] in requests.keys()):
                    requests[rate['bookId']].append(review['user']['_id'])
                else:
                    print(review['user']['_id'])
                    requests[rate['bookId']] = [review['user']['_id']]

    books_to_delete_similarities = [item['bookId'] for item in new_rates]
    delete_similarity_by_books(books_to_delete_similarities)

    new_similaritites = []
    for book in new_rates:
        books_to_compare = get_rates_by_users(requests[book['bookId']])

        book_reviews = change_scheme_user_rate(book)

        for other in books_to_compare:
            if other['bookId'] != book['bookId']:
                if any(other['bookId'] in item.values() and book['bookId'] in
item.values() for item in new_similaritites):
                    continue

                scope = koef_pearson(book_reviews, other)
                if scope > 0:
                    similarity = {
                        'book1Id':book['bookId'],
                        'book2Id':other['bookId'],
                        'similarity':scope
                        }
                    new_similaritites.append(similarity)

    insert_books_similarity(new_similaritites)
    return len(new_similaritites)
```

```
def calculate_recomendations_by_book(book_id, number):
    book_similarities = get_similarity_by_book(book_id)
    similarities = []
    for item in book_similarities:
            similarities.append((item['similarity'], item['book1Id'] if
item['book1Id'] != book_id else item['book2Id']))

    similarities.sort()
    similarities.reverse()
    similarities = similarities[:number]
    bookIds_to_get = [item[1] for item in similarities]
    books = get_books_by_ids(bookIds_to_get)

    for book in books:
        book['similarityRate'] = [x for (x, y) in similarities if y == book['id']][0]
        book['id'] = str(book['id'])
        del(book['_id'])

    books = sorted(books, key=lambda d: d['similarityRate'], reverse=True)
    return books


def calculate_recomendations_for_user(user_id, number):
    user_reviews = get_rates_by_users([user_id])
    changed_user_reviews = {}

    book_similatity = {}
    for review in user_reviews:
        similarities = []
        changed_user_reviews.update(change_scheme_book_rate(review))
        similarity_dict = get_similarity_by_book(review['bookId'])
        for item in similarity_dict:
            similarities.append((item['similarity'], item['book1Id'] if
item['book1Id'] != review['bookId'] else item['book2Id']))

        book_similatity.update({review['bookId'] : similarities})

    recommended_items = get_recommended_items(changed_user_reviews,
book_similatity)[:number]

    bookIds_to_get = [item[1] for item in recommended_items]
    books = get_books_by_ids(bookIds_to_get)
    for book in books:
        book['similarityRate'] = [x for (x, y) in recommended_items if y ==
book['id']][0]
        book['id'] = str(book['id'])
        del(book['_id'])

    books = sorted(books, key=lambda d: d['similarityRate'], reverse=True)
    return books
```

ДОДАТОК В. РЕКОМЕНДАЦІЙНА СИСТЕМА НА ОСНОВІ ВМІСТУ: КОД РЕАЛІЗАЦІЇ ПОПЕРЕДНЬОЇ ОБРОБКИ ДАНИХ

```
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

from sklearn import preprocessing
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

def initial_df_preprocess(dataframe):
    dataframe.genres = [' '.join(map(str, l)) for l in dataframe.genres]

    columns_to_join = ["title", "description", "genres"]
    dataframe['full_desc'] = dataframe[columns_to_join].T.agg(' '.join)
    dataframe.drop(columns_to_join, axis=1, inplace=True)

    dataframe["full_desc"] = dataframe['full_desc'].apply(text_preprocess)
    label_encoder = preprocessing.LabelEncoder()

    dataframe['author']= label_encoder.fit_transform(dataframe['author'])
    return dataframe


def text_preprocess(sample):
    lower_text = sample.lower()

    entry_no_punct = lower_text.translate(str.maketrans('', '', string.punctuation))

    words = word_tokenize(entry_no_punct)

    stops = stopwords.words('english')
    clean = [word for word in words if word not in stops]

    ps = PorterStemmer()
    stemmed_list = [ps.stem(w) for w in clean]

    return ' '.join(stemmed_list)

def transform(text):
    cv_transf = CountVectorizer()
    tfidf_transf = TfidfTransformer()
    cv_transf.fit(text)
    cv_metrix = cv_transf.transform(text)
    tfidf_transf.fit(cv_metrix)
    result = tfidf_transf.transform(cv_metrix)
    return result
```

ДОДАТОК Г. РЕКОМЕНДАЦІЙНА СИСТЕМА НА ОСНОВІ ВМІСТУ: КОД РЕАЛІЗАЦІЇ ФОРМУВАННЯ ПРОПОЗИЦІЙ

```
import pandas as pd
import numpy as np
from scipy.sparse import hstack
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity

from content_dal import *
from content_utils import *

def calculate_recomendations_by_book(book_id, number):
    book = get_book_by_id(book_id)
    books = get_books_not_in_ids([book_id])

    books.insert(0, book)
    books_df = pd.DataFrame(books)

    books_df = initial_df_preprocess(books_df)

    data = transform(books_df["full_desc"])
    data = hstack((data,np.array(books_df['author'])[:,None]))

    result = cosine_similarity(data, [data.toarray()[0]])
    mask = books_df["_id"] != book_id
    result_df = pd.DataFrame()
    result_df["id"] = books_df[mask]["_id"]
    result_df["similarity"] = result[1:]

    result_df.sort_values('similarity', ascending=False, inplace=True)
    bookIds_to_get = result_df["id"].head(number).tolist()
    books = get_books_by_ids(bookIds_to_get)

    for book in books:
        book['similarityRate'] = round(result_df[result_df["id"] ==
book['id']]["similarity"].values[0], 4)
        book['id'] = str(book['id'])
        del(book['_id'])

    books = sorted(books, key=lambda d: d['similarityRate'], reverse=True)

    return books


def calculate_recomendations_for_user(user_id, number):
    user_reviews = get_rates_by_user(user_id)

    rates_df = pd.DataFrame(user_reviews)
```

```
    rates_df.drop("_id", axis=1, inplace=True)

    books = get_books()
    books_df = pd.DataFrame(books)

    main_df = pd.merge(books_df, rates_df, how="left", left_on="_id",
right_on="bookId")
    main_df.drop("bookId", axis=1, inplace=True)
    main_df = initial_df_preprocess(main_df)

    main_df.sort_values('rate', inplace=True)
    mask = main_df.index.isin(main_df.index.values) & main_df.rate.isnull()

    test_amount = len(main_df[mask])

    data = transform(main_df["full_desc"])
    data = hstack((data,np.array(main_df['author'])[:,None]))

    model = GaussianNB()

    X_train, X_test, y_train, y_test = train_test_split(data,
                                            main_df['rate'], test_size=test_amount,
                                            shuffle=False)

    trained_model = model.fit(X_train.toarray(), y_train)
    pred = trained_model.predict(X_test.toarray())

    recommended_items = pd.DataFrame()
    recommended_items["id"] = main_df[mask]["_id"]
    recommended_items["expected_rate"] = pred
    recommended_items.sort_values('expected_rate', ascending=False, inplace=True)
    bookIds_to_get = recommended_items["id"].head(number).tolist()
    books = get_books_by_ids(bookIds_to_get)

    for book in books:
        book['similarityRate'] = recommended_items[recommended_items["id"] ==
book['id']]["expected_rate"].values[0]
        book['id'] = str(book['id'])
        del(book['_id'])

    books = sorted(books, key=lambda d: d['similarityRate'], reverse=True)

    return books
```