Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: **"Рекомендаційна система онлайн – книгарні на основі засобів машинного навчання"**

Виконав: студент **VI** курсу, групи **КН-61м**
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Годованець А. Д.
(прізвище та ініціали)
Керівники **Мокрицька О. В.**
(прізвище та ініціали)
Головата С. Б.
(прізвище та ініціали)
Рецензент **Поберейко Б. П.**
(прізвище та ініціали)

Львів – 2025 року

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"

ЗАТВЕРДЖУЮ
Завідувачка кафедри КН
_____Борецька І.Б.
„10" грудня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Годованець Адам Данилович
(прізвище, ім'я, по батькові)

1.Тема роботи "Рекомендаційна система онлайн – книгарні на основі засобів машинного навчання"

Керівник роботи Мокрицька О. В., к.т.н, доцент, Головата С. Б., ст. викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "29" квітня 2025 р. № С-288

2.Термін подання студентом проекту(роботи) 10 грудня 2025 р.

3. Вихідні дані до роботи: Формулювання задачі та її формалізація. Аналіз попередніх досліджень. Огляд програмних засобів для реалізації поставленого завдання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Розділ 1. Стан проблемної області
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
Розділ 5. Розроблення стартап – проєкту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди доповіді, актуальність теми, постановка завдання, аналіз отриманих результатів, висновки

6. Дата видачі завдання 1 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних та інших джерел згідно досліджуваної теми. Збір потрібних матеріалів.	03.05.2025-05.06.2025 р.	виконано
2	Постановка задачі та її формалізація.	06.06.2025-12.06.2025 р.	виконано
3	Вибір та обгрунтування методів і засобів розв'язання завдання.	13.06.2025-16.07.2025 р	виконано
4	Програмна реалізація системи.	17.07.2025-18.10.2025 р.	виконано
5	Оформлення опису створеної програми.	19.10.2025-20.10.2025 р.	виконано
6	Аналіз отриманих результатів виконання програми.	21.10.2025 р.	виконано
7	Здача пояснювальної записки на перевірку та виправлення виявлених помилок.	22.10.2025-10.12.2025 р.	виконано

Студент _____ Годованець А. Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Мокрицька О. В.
(підпис) (прізвище та ініціали)

_____ Головата С. Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота містить 89 сторінок пояснювальної записки, 35 рисунків, 8 таблиць, 3 додатки та 21 джерело.

У ході виконання дипломної роботи розроблено рекомендаційну систему онлайн – книгарні, яка дає змогу підвищити коефіцієнт конверсії онлайн – книгарні за рахунок надання списку книг, який відповідає інтересам користувача.

В першому розділі представлено аналіз предметної сфери та існуючих аналогів. Здійснена специфікація вимог. У другому розділі описано процес проєктування та моделювання системи. В третьому розділі досліджено існуючи рекомендаційні алгоритми та метрики оцінки точності рекомендацій. В четвертому розділі продемонстровано основні етапи реалізації застосунку.

Ключові слова: Python, Flask, Scikit – learn, web scraping, TF-IDF, машинне навчання, рекомендаційні системи.

ABSTRACT

The master’s thesis contains 89 pages of explanatory note, 35 figures, 8 tables, 3 appendices and 21 sources.

In the course of the thesis, a recommendation system for online bookstores was developed, which allows increasing the conversion rate of online bookstores by providing a list of books that match the user's interests.

The first chapter presents an analysis of the subject area and existing analogues. The requirements have been specified. The second chapter describes the process of designing and modelling the system. The third chapter examines existing recommendation algorithms and metrics for evaluating the accuracy of recommendations. The fourth chapter demonstrates the main stages of the application's implementation.

Keywords: Python, Flask, Scikit – learn, web scraping, TF-IDF, machine learning, recommendation systems.

Технічне завдання

- Необхідно створити рекомендаційну систему онлайн – книгарні на основі засобів машинного навчання.
- Головними вимогами до системи є:

– Визначити логічну та фізичну модель даних.
– Охарактеризувати бізнес-процеси у системі.
– Здійснити специфікацію вимог.
– Спроєктувати архітектуру програмного забезпечення.
– Розробити алгоритм надання рекомендацій.
– Реалізувати та протестувати рекомендаційну систему онлайн-книгарні на основі засобів машинного навчання.
– Код застосунку має бути розроблений мовою програмування *Python.*
– В якості database engine необхідно використати *SQLite.*
– Збір даних про активність користувача на сайті.
– Формування вектору інтересів користувача.
– Надання рекомендацій згідно з інтересами.
– Підбір книг, подібних до обраної книги.
– Фільтрація та пошук даних за різними параметрами (назва, автор, видавець, рік видання, тощо).

- Розроблена система, повинна містити весь перерахований вище функціонал й надавати його користувачам системи.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД	База даних
ЕОМ	Електронно – обчислювальна машина
МН	Машинне навчання
ООП	Об’єктно – орієнтоване програмування
ПЗ	Програмне забезпечення
JSON	JavaScript Object Notation
ORM	Object – Relational Mapping
URL	Uniform Resource Locator

ВСТУП

Актуальність роботи

Рекомендаційні алгоритми використовуються у багатьох популярних сервісах для адаптації контенту під потреби та інтереси кожного окремого користувача. Успішність онлайн-книгарні залежить від того чи вдасться запропонувати релевантні книги та підвищити конверсію.

Рекомендаційні системи є поширеними допоміжними інструментами для здійснення онлайн-маркетингу. Чим більше даних накопичується у інформаційній системі, тим більш складні алгоритми персоналізації контенту необхідно реалізувати. У нагромадженні всіх доступних товарів та послуг досить важко знайти «вручну» щось потрібне, актуальне та якісне. Релевантні інтересам пропозиції здатні зацікавити користувача набагато більше, ніж випадкова підбірка контенту або товарів. Зацікавлення товаром сприяє підвищенню конверсії, а отже робить бізнес більш ефективним та прибутковим. Саме тому тема магістерської роботи є актуальною та має практичне значення.

Предметом дослідження є методи розробки рекомендаційної системи онлайн-книгарні на основі засобів машинного навчання.

Об'єктом дослідження є процес рекомендування книг на основі даних про інтереси користувача.

Мета роботи – підвищення коефіцієнта конверсії онлайн-книгарні за рахунок надання списку книг, який відповідає інтересам користувача.

Завдання:

1. Проаналізувати предметну сферу, існуючі аналоги та джерела даних.
2. Визначити логічну та фізичну модель даних.
3. Охарактеризувати бізнес-процеси у системі.
4. Здійснити специфікацію вимог.
5. Спроєктувати архітектуру програмного забезпечення.
6. Розробити алгоритм надання рекомендацій.
7. Реалізувати та протестувати рекомендаційну систему онлайн-книгарні на основі засобів машинного навчання.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів полягає в розробленні рекомендаційної системи онлайн-книгарні, що поєднує контентний підхід, колаборативну фільтрацію та моделі глибинного навчання. Запропоновано метод формування профілю користувача, який інтегрує поведінкові дані з семантичними характеристиками опрацьованих книг. Удосконалено обробку текстових описів творів шляхом використання сучасних NLP-моделей для побудови якісних векторних представлень. Створено програмну реалізацію, що підтверджує ефективність запропонованих методів у підвищенні точності рекомендацій.

Практичне значення одержаних результатів

Отримані результати мають значне практичне значення для підвищення ефективності роботи онлайн-книгарень. Розроблена рекомендаційна система дозволяє автоматизовано формувати персоналізовані добірки книжок, що скорочує час пошуку потрібної літератури та підвищує задоволеність користувачів. Використані алгоритми машинного навчання забезпечують більш точні рекомендації навіть за умов обмеженої кількості даних або появи нових користувачів і книг. Запропонована програмна реалізація може бути інтегрована в існуючі платформи електронної комерції, сприяючи зростанню продажів та активності користувачів. Система також може бути адаптована для інших галузей, що потребують персоналізованих рекомендацій, таких як музичні чи відеосервіси. Виконана робота створює основу для подальшого вдосконалення інтелектуальних сервісів персоналізації в сфері онлайн-торгівлі.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Сутність рекомендаційних систем

Рекомендаційна система – це система фільтрації інформації, призначена для формування рейтингового переліку об'єктів. Метою рекомендаційної системи є персоналізація контенту та пошук індивідуального підходу до користувачів.
Наприклад, при придбанні смартфону пропонуються сумісні з обраною моделлю аксесуари, у магазині одягу – прийнятний за розміром та фасоном варіант тощо.
Отже, у ситуаціях вибору однієї з багатьох альтернатив люди схильні покладатися на поради аби не витрачати час на самостійне покрокове повторення процесу аналізу та перегляд всіх доступних альтернатив для прийняття зваженого рішення.

Поради від фахівців є історично першими рекомендаціями. Що стосується програмного забезпечення, то до 90-х років XX століття ЕОМ використовувалися для аналізу даних про перехресні продажі та для розрахунку базових статистичних показників. Отримана у такий спосіб рекомендація вдосконалювалася та адаптувалася до реального життя вручну відповідно до інтуїції та практичного досвіду продавця.

В період з 1992 по 1996 рік розроблено більш складні алгоритми для рекомендаційних систем. В даній ґалузі активно працювали як науковці (*«GroupLens»*), так і комерційні організації (*«Amazon»*). Шалений успіх рекомендаційних систем та занадто висока оцінка їх важливості завершилися крахом та розчаруванням у 2000-х роках. Ситуація змінилася завдяки *«Netflix Prize»* у період з 2006 по 2009 рік. Компанія *«Netflix»* пообіцяла 1 мільйон доларів США переможцю конкурсу, завдання якого полягало у вдосконаленні алгоритму рекомендацій. *«Netflix Prize»* відновив популярність рекомендаційних систем та заохотив багатьох талановитих науковців та розробників працювати в даній галузі [1]. Ключовими постатями в сфері вивчення рекомендаційних систем стали науковці *John T. Riedl* (США), *Barry Smyth* (Ірландія), *Pearl Pu* (Швейцарія) та *Pasquale Lops* (Італія).

На сьогоднішній день всі рекомендаційні алгоритми вичерпно досліджені, реалізовані та протестовані. Але все одно кожна окрема рекомендаційна система все

ще залишається ***унікальним*** продуктом, оскільки її принцип дії напряму залежить від властивостей обраного набору даних та задач, які перед цією системою ставить бізнес.

Рекомендаційні системи мають свої переваги та недоліки, які варто враховувати при їх впровадженні.

Переваги використання рекомендаційних систем:

1.*Розширення цільової аудиторії.* Рекомендаційні системи сприяють тому що користувачі затримуються на сайті на довший час через зацікавлення.

2.*Покращення користувацького досвіду.* Актуальні рекомендації та інші інструменти для фільтрації контенту підвищують зручність сайту.

3.*Підвищення конверсії.* Конверсія – це співвідношення між кількістю активних користувачів та кількістю всіх відвідувачів сайту. Онлайн-маркетинг вважається успішним за умови досягнення високого рівня конверсії, тобто якщо вдалося спонукати відвідувачів придбати товар, підписатися на сторінку бізнесу в соцмережах або здійснити будь-яку іншу подібну взаємодію. Якщо користувач отримує релевантний інтересам контент, то ймовірність взаємодії зростає.

4. *Автоматизація процесу вибору.* Рекомендації економлять час, оскільки не потрібно самостійно шукати та фільтрувати всю доступну інформацію.

Недоліки використання рекомендаційних систем:

1. *Проблема «бульбашки».* Рекомендаційні системи обмежують різноманітність та новизну, оскільки показують лише те що відповідає вподобанням. Це може мати досить серйозні наслідки, адже людина замикається у своїй «бульбашці», втрачає можливість ознайомитися з іншими точками зору та отримує постійне підкріплення своїх старих звичок та вподобань (якими б умовно «правильними» чи «неправильними» вони не були). Особливо небезпечна така цензура у випадку підбору контенту, який формує цінності та світогляд. Що стосується звичайних товарів та послуг, то «бульбашка» заважає вийти з зони комфорту та спробувати щось нове.

2. *Конфіденційність.* Чим більше рекомендаційна система знає про користувача, тим точніше будуть рекомендації. Але збір великої кількості персональних даних є досить контраверсійнм з точки зору приватності та безпеки.

3. *Невідповідність інтересам.* Не завжди можливо точно визначити інтереси користувача або передбачити їх зміну.

4. *Вартість впровадження та обслуговування.* Розробка, впровадження та обслуговування рекомендаційних систем може вимагати значних витрат (особливо якщо йдеться про великі обсяги даних та складні інформаційні системи).

5. *Етичність.* Рекомендаційна система здатна робити популярне ще більш популярним. В такому випадку важко забезпечити справедливу конкуренцію, оскільки нові товари або новий контент так і залишаться непоміченими. Також рекомендації часто є стереотипними або навіть образливими, що не прийнятно в контексті поваги до індивідуальності та свободи вибору кожної окремої людини.

Крім того, рекомендаційні системи обслуговують інтереси бізнесу і можуть нехтувати інтересами споживача у випадку конфлікту. Наприклад, проблемною є демонстрація цигарок у списку рекомендованих товарів. З одного боку, цей товар дійсно входить в сферу інтересів користувача (що підтверджується історією покупок) і його вигідно продавати. З іншого боку, не дуже етично нагадувати про паління людині, яка, можливо, якраз намагається кинути палити. Таких прикладів можна навести безліч, тому не варто забувати про етичну сторону розробки ПЗ.

Враховуючи наведені переваги та недоліки необхідно розробити рекомендаційну систему у такий спосіб щоб вона покращувала користувацький досвід не порушуючи приватність та надаючи простір для новизни. Наприклад, у випадку книжкових рекомендацій не можна пропонувати лише детективи користувачу який явно полюбляє саме цей жанр. Варто також запропонувати певну кількість найбільш популярних зразків літератури інших жанрів.

1.2. Вплив рекомендаційних алгоритмів на конверсію

Як було зазначено вище, конверсія – це співвідношення кількості покупців та загальної кількості лідів. Лідом називається потенційний покупець (відвідувач сайту). Забезпечення високої конверсії вважається вигідною стратегією онлайн-маркетингу. Прагнення до охоплення більшої аудиторії не гарантує інтерес до товару та здійснення покупки. Отже, вкладення коштів у рекламу для привернення максимальної кількості випадкових лідів не є доцільним кроком. Важливіше якісно проаналізувати цільову аудиторію та направити (таргетувати) на неї ретельно продуману рекламу. Надважливим кроком є утримання уваги потенційних відвідувачів. Дана задача стає складнішою з плином часу через зниження рівня уважності та концентрації внаслідок впливу соціальних мереж [19]. Для того щоб конкурувати з яскравим та привабливим розважальним контентом у соцмережах необхідно ретельно пропрацювати дизайн вебсайту. Але лише візуальної складової та зручності мало для утримання уваги. Користувач буде охоче взаємодіяти лише з релевантним контентом. Саме тому впровадження рекомендаційного алгоритму є важливою частиною успішної стратегії ведення бізнесу онлайн.

Крім того, онлайн-торгівля має специфічні риси. Наприклад, у «звичайній» книгарні кількість представлених книг регулюється просторовими обмеженнями. Оренда або придбання приміщення зазвичай надто дорого коштує для того, щоб захаращувати місця на стендах вузькоспеціалізованою літературою, яка не надто активно продається. У кращому випадку у «фізичній» книгарні можуть запропонувати привезти певну книгу під замовлення.

Онлайн-вітрина є необмеженим простором для демонстрації товару. Онлайн-магазин нічого не втрачає при торгівлі нішевими виданнями. Можливість придбати майже будь-що є однією з найсильніших переваг онлайн-торгівлі. Саме тому виникає ризик перенавантаження користувача надто великою кількістю доступних альтернатив.

Дослідженням впливу кількості альтернатив на поведінку покупців займаються фахівці у ґалузі психології. Досить логічно припустити що серед великої кількості варіантів майже всі покупці зможуть знайти щось для себе. Але у 2000

році було опубліковано дослідження зі спростуванням цього твердження [18]. В ході відповідного експерименту покупців розділили на дві групи. Одна група обирала серед 24 альтернатив, а інша – серед 6. З'ясувалось, що вища конверсія у групі з меншою кількістю варіантів товару (30% проти 3%). Хоча можливість обирати серед багатьох варіантів привабила 60% респондентів. Отже, доцільніше пропонувати потенційним покупцям обмежену кількість товарів. Схематичне представлення результатів дослідження наведено на рисунку 1.1.

Рисунок 1.1 – Вплив кількості альтернатив на конверсію [18]

З психологічної точки зору така поведінка пояснюється тим що необхідність враховувати велику кількість варіантів при аналізі ситуації та здійсненні вибору перевантажує людський мозок. Також відомо що невизначеність є фактором стресу, а при збільшенні числа альтернатив пропорційно збільшується ймовірність зустрітись з чимось незнайомим та нестандартним. Крім того, покупці які мали аж 24 альтернативи частіше залишалися незадоволені своїм вибором та мали сумніви стосовно його правильності. Маркетологи прийшли до висновків про користь обмеженого асортименту набагато раніше за дослідників та науковців. Так, скорочення кількості варіантів шампуню від корпорації «Proctor & Gamble» з 26 варіантів до 15 призвело до зростання продажів на 10% [18].

1.3. Класифікація рекомендаційних систем

Загальноприйнятою є класифікація рекомендаційних систем за галуззю, метою, походженням оцінки, рівнем персоналізації та інтерфейсом. Таксономія рекомендаційних систем представлена на рисунку 1.2.

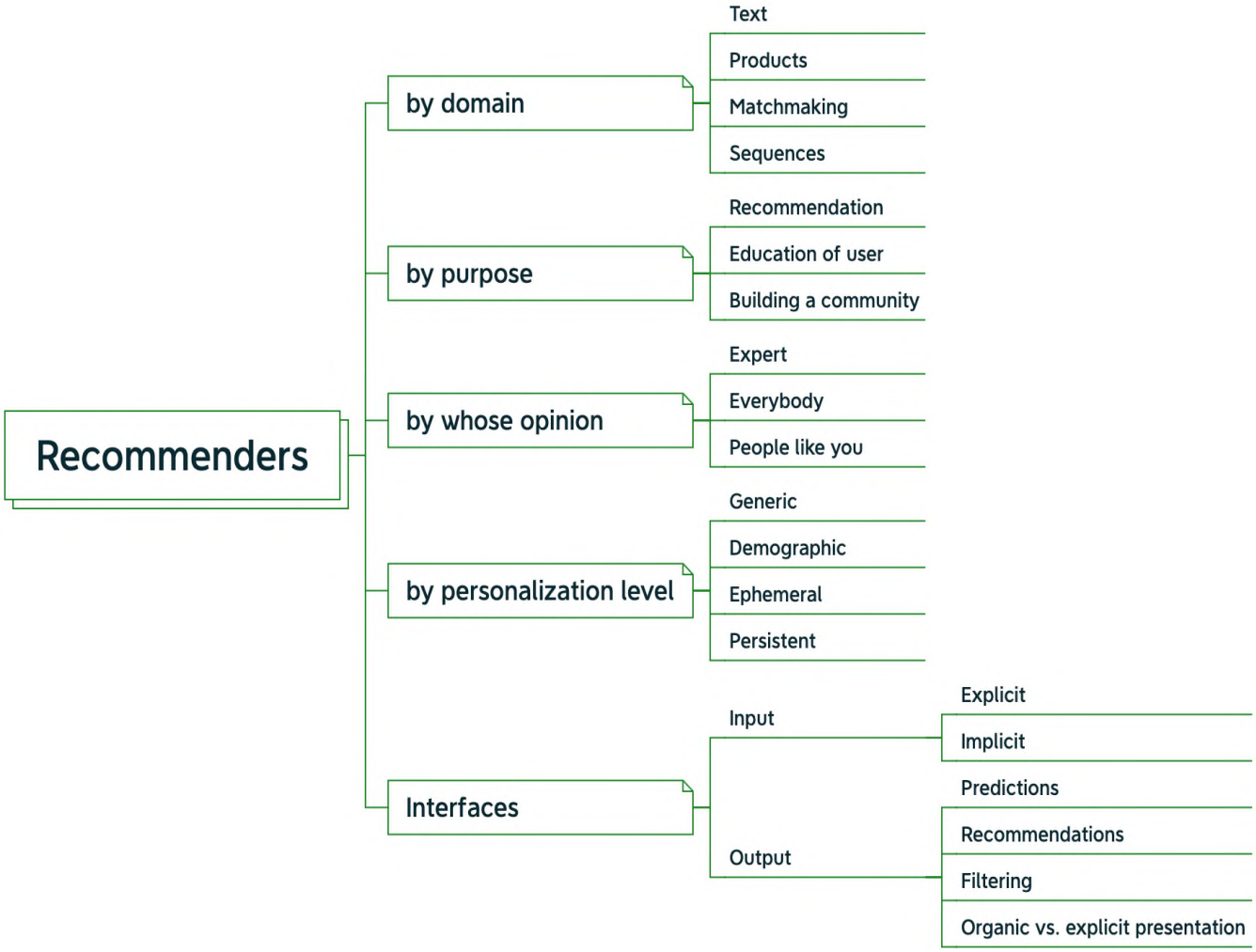

Рисунок 1.2 – Таксономія рекомендаційних систем

Також до цієї класифікації входить рекомендаційний алгоритм, але, враховуючи складність та варіативність цієї класифікації, таксономію рекомендаційних алгоритмів представлено окремо (рис. 1.6).

Що стосується *галузі (domain)*, то найбільш яскравим прикладом рекомендації з метою знайомства є «люди, яких ви можете знати» у *«Facebook»*, релевантні вакансії та роботодавці у *«LinkedIn»* (рис. 1.3) тощо. Товари рекомендують в інтернет-магазинах, відповідні інтересам тексти та інші види контенту – в соціальних мережах.

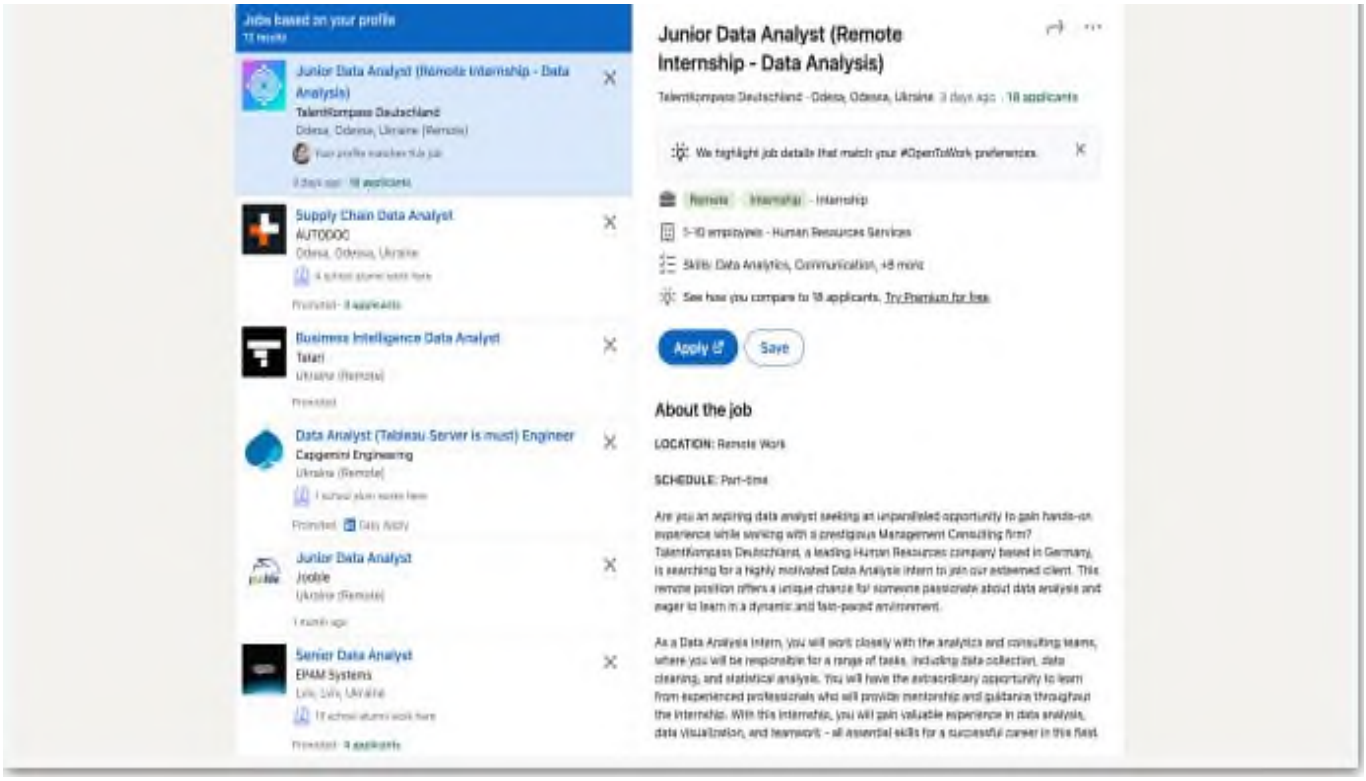

Рисунок 1.3 – Релевантні інтересам вакансії у *«LinkedIn»*

Мета (purpose) рекомендації може полягати як у зверненні уваги на вибірку об’єктів, так і у навчанні користувача (наприклад, підказка про сполучення клавіш для швидкого виконання задачі). Також метою може бути створення спільноти.

Походження оцінки (by whose opinion) від людей зі схожими інтересами означає що в системі використовується колаборативна фільтрація. Найкращий приклад оцінки від експерта – літературна критика. Оцінка від усіх користувачів враховується при визначенні загального рейтингу певного товару чи контенту.

Рівень персоналізації (personalization level) показує наскільки рекомендація адаптована під конкретного користувача. Прикладом узагальненої рекомендації (generic) є категорія «топ продажів» в будь-якому інтернет-магазині. Даний топ не враховує індивідуальні інтереси, оскільки важить лише те, що певний товар купують частіше за інші товари.

Демографічні рекомендації (demographic) – це рекомендації на основі належності користувача до певної демографічної групи. В даному випадку люди поділяються на групи за статтю, віком, місцем проживання та іншими демографічними параметрами. Для кожної групи визначається вектор інтересів. Такий підхід дещо обмежений, оскільки у демографічній групі можуть бути особи з

нетиповими інтересами та, зазвичай, важко визначити до якої категорії віднести користувача який вперше на сайті. Прикладом демографічних рекомендацій є демонстрація саме жіночої парфумерії у топі товарів для користувачів жіночої статі. Така рекомендація більш адаптована на цільову аудиторію, але все одно не є достатньо точною.

Рекомендація на основі тимчасового зацікавлення (ephemeral) робиться з огляду на поточну активність користувача. Наприклад, до цієї категорії відноситься пропозиція «разом дешевше» на сайті відомого українського ритейлера *«Rozetka»* (рис. 1.4).

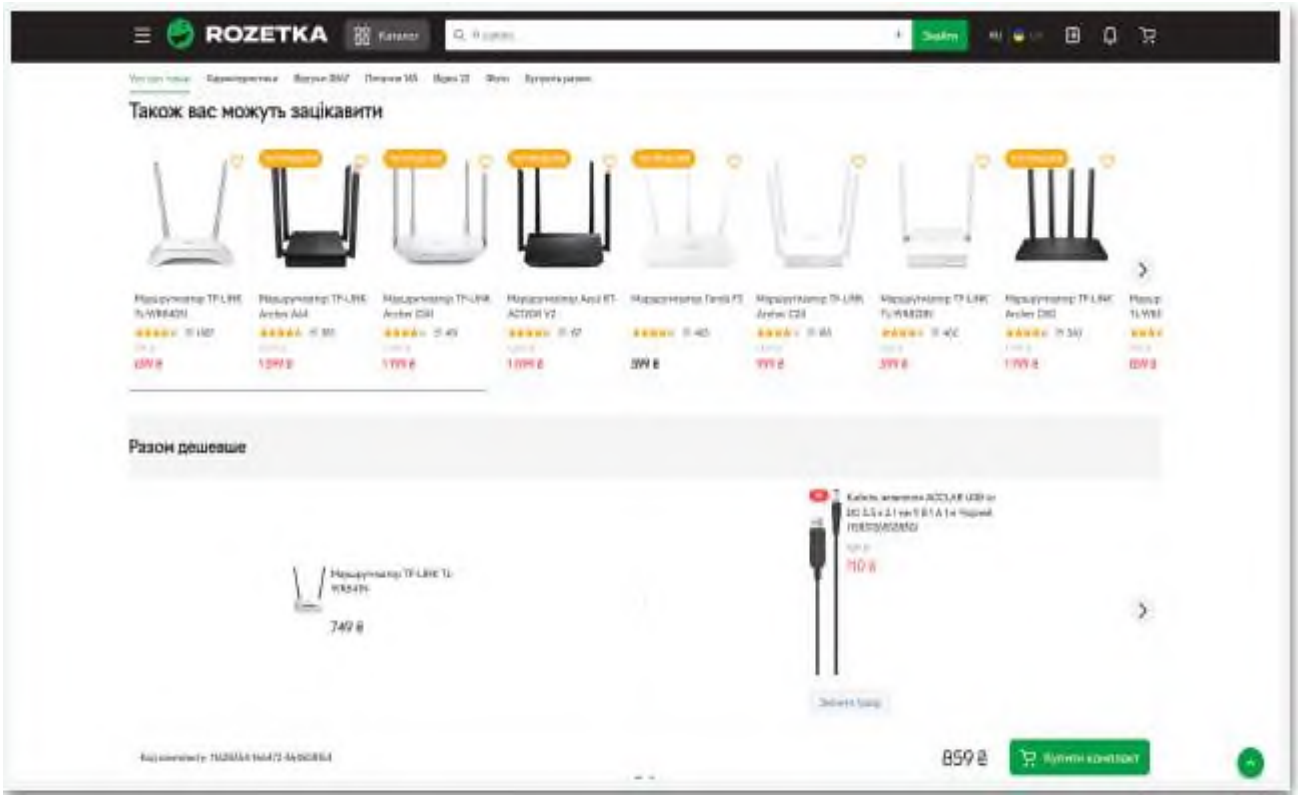

Рисунок 1.4 – Рекомендація на основі тимчасового зацікавлення

Досить часто одні товари купуються в комплекті з іншими товарами. Наприклад, до смартфона пропонують спеціалізовані аксесуари чи до маршрутизатора сумісний кабель живлення. Інформація про споріднені товари отримується з аналізу транзакцій.

До найпопулярніших способів визначення груп взаємопов’язаних товарів відносяться таблиці перехресних продажів (Cross-Sell Tables), теорема Байєса, Lift Metric, тощо. Теорема Байєса представлена за допомогою формули 1.1.

$$P(A \mid B) = \frac{P(A \mid B) * P(A)}{P(B)} \qquad (1.1)$$

де: P(A) – ймовірність настання події A

P(B) – ймовірність настання події B

P(A|B) – ймовірність настання події A за умови істинності B

P(B|A) – ймовірність настання події B за умови істинності A

Також «корзину» продуктів можна скласти з використанням так званої Lift Metric. Формула 1.2 використовується для розрахунку ступеню взаємозв'язку між товарами.

$$\frac{P(A \wedge B)}{P(A) * P(B)} \qquad (1.2)$$

де: P(A^B) – ймовірність одночасного настання подій A та B

P(A) – ймовірність настання події A

P(B) – ймовірність настання події B

Важливо також враховувати контекст та напрямок взаємозв'язку. Наприклад, люди які купують люксове авто часто обирають дорогі водійські шкіряні рукавички. Тому доречно запропонувати такий аксесуар людині, яка купує дороге авто. Але зовсім недоречно пропонувати дороге авто в якості аксесуару до рукавичок. Також в даному випадку *одночасність подій не доводить наявність кореляції між ними*. При порушенні цього правила можливі численні помилки та спекуляції. Найяскравішим прикладом подібного парадоксу є *«The Beer and Diapers Story»* [5]. Особливість ситуації полягала у тому що як пиво, так і дитячі підгузки є популярними товарами самі по собі, тому не дивно що вони з'являлись одночасно в одній корзині. Така одночасність не означала що існує певна логічний взаємозв'язок між цими явищами.

Рекомендації на основі стабільних інтересів (persistent) можливі лише для постійних користувачів, чию поведінку вдалося дослідити. Це точні та найбільш персоналізовані рекомендації. Даний підхід використовується у алгоритмах *«YouTube», «Netflix», «Coursera»* (рис. 1.5) та інших подібних ресурсах.

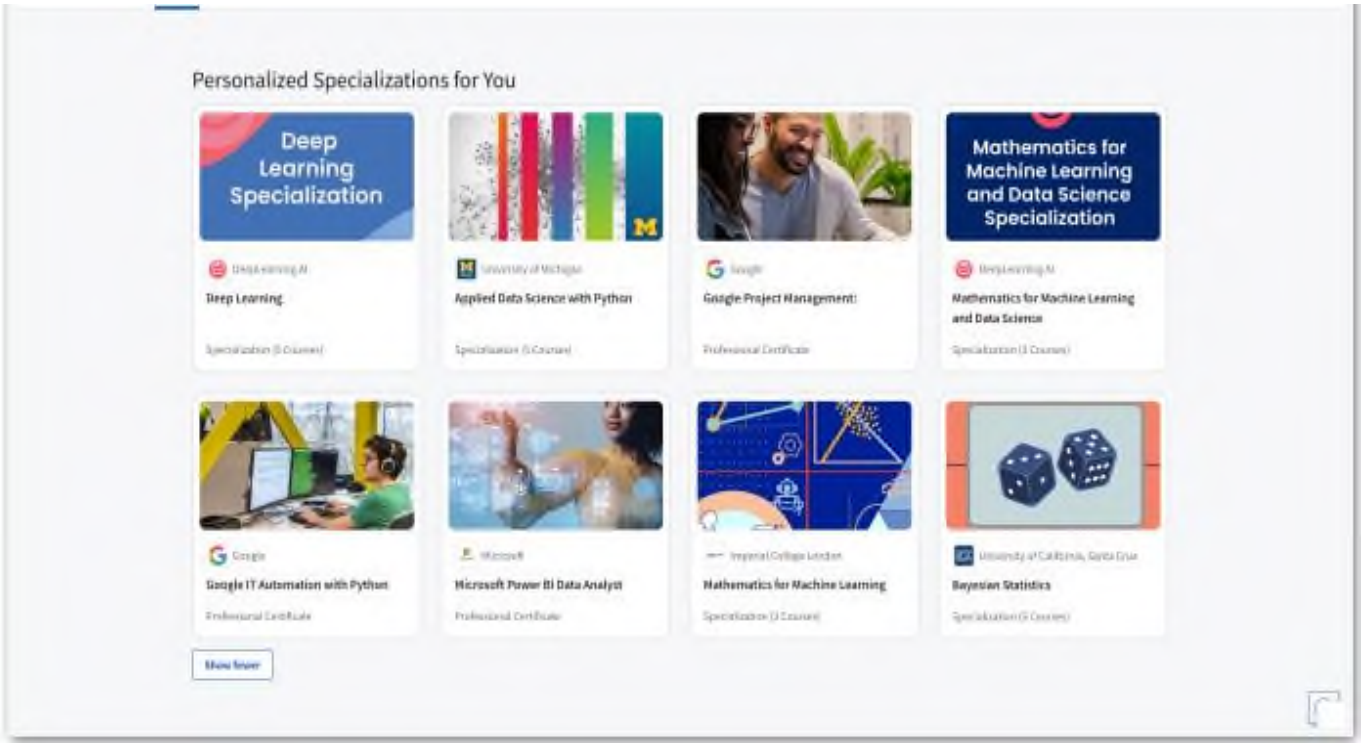

Рисунок 1.5 – Рекомендація на основі стабільних патернів у поведінці

Зазвичай у рекомендаційних системах поєднують декілька рівнів персоналізації для того щоб знайти підхід як до нових, так і до постійних клієнтів.

Рекомендаційна система отримує *вхідні дані* в явному (лайки, відгуки, рейтинги, рецензії) та неявному (час перебування на сайті, перехід за посиланнями, тощо) форматі. Результатом роботи системи можуть бути рекомендації, прогнози та фільтрація даних. Рекомендації краще представляти ненав'язливо та як перелік можливих опцій, а не як агресивну вказівку подібну до «обов'язково подивіться серіал» чи «вам зі 100% ймовірністю сподобається товар». Також варто зважати на те що з часом потреби та інтереси користувачів можуть змінюватися, отже важливо дати простір для пошуку нових та нетипових рішень.

Класифікація рекомендаційних алгоритмів наведена на рисунку 1.6.

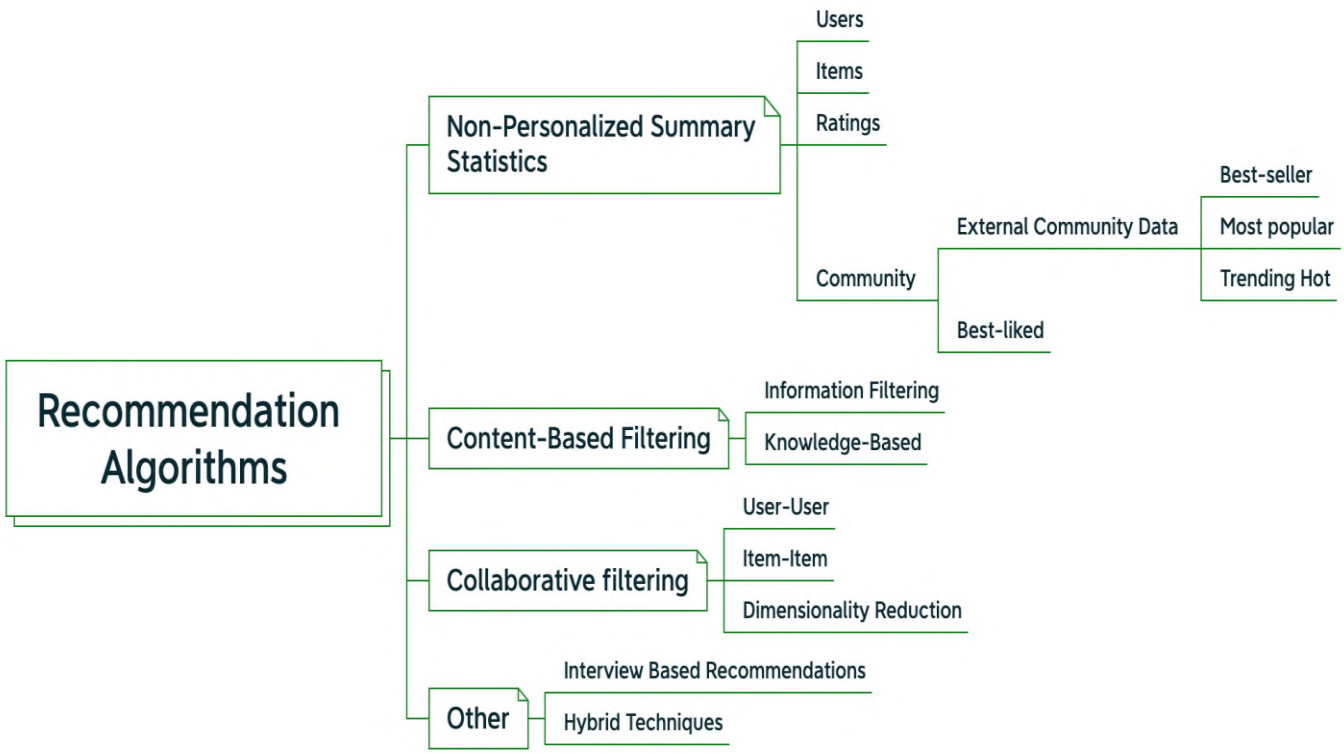

Рисунок 1.6 – Класифікація рекомендаційних алгоритмів

Історично першим та найпростішим є неперсоналізований статистичний підхід (Non-Personalized Summary Statistics), який фокусується на аналізі статистичних даних. Фільтрація на основі контенту (Content-Based Filtering) передбачає детальне дослідження якостей товару. Колаборативна фільтрація (Collaborative Filtering) переміщує фокус уваги на користувачів, подібності між ними та закономірності у поведінці людей зі схожими інтересами (рис. 1.7).

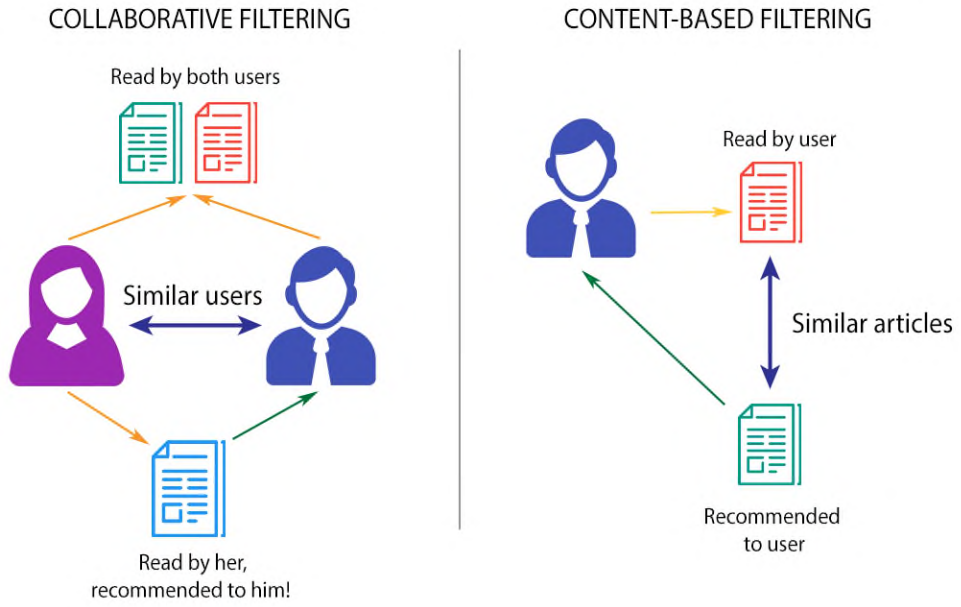

Рисунок 1.7 – Порівняння рекомендаційних алгоритмів [6]

Отже, вибір алгоритму залежить від поставлених задач та доступу до потоків даних.

Існують різні варіанти фільтрації на основі контенту. Варто назвати наступні різновиди: pure information filtering systems, case-based reasoning systems та knowledge-based navigation systems. В будь-якому випадку принцип роботи рекомендаційного алгоритму складається з наступних кроків:

1. Визначення вектору атрибутів, які характеризують контент.
2. Визначення ***вектору інтересів користувача***.
3. Поєднання зазначених векторів з метою виявлення найбільш релевантного контенту.

Користувач може напряму визначити вектор інтересів (в опитуванні чи профілі), можна зібрати дані явно (рейтинги, лайки, тощо) або неявно (з активності на сайті). Основою алгоритму є ***тег (ключове слово)***. Від того наскільки точно теги характеризують контент залежить якість рекомендацій. В даному випадку рекомендації будуть дуже типовими та передбачуваними.

Фільтрація на основі контенту не призначена для пошуку складних взаємозв'язків.

Колаборативна фільтрація добре працює у системах, які володіють великою кількістю даних про поведінку репрезентативної вибірки користувачів. Але на збір такого датасету йдуть роки. Крім того, відповідна система має бути досить популярна для того, щоб взаємодіяти з достатньою кількістю людей. Якщо даних про поведінку користувачів недостатньо (відповідна *utility matrix* містить надто багато пустих комірок), то варто застосувати фільтрацію на основі контенту. Фільтрація на основі контенту ефективна у випадку впровадження рекомендаційних алгоритмів у нову систему, яка ще не накопичила дані про користувачів. Також даний підхід є гарним варіантом у випадку взаємодії з незареєстрованим чи «новим» користувачем, про якого майже нічого невідомо.

При виборі підходу до надання рекомендацій варто спиратися на структуру та вміст датасету, особливості предметної сфери та мету системи. Не можна сказати що, наприклад, колаборативна фільтрація краща (або) гірша за фільтрацію на основі контенту, бо доцільність використання певного алгоритму залежить від контексту.

Інколи краще всього працює гібридний підхід, який поєднує в собі декілька різних підходів.

Отже, варто віднести рекомендаційну систему онлайн-книгарні на основі засобів машинного навчання до певних категорій відповідно до наведеного вище фреймворку для класифікації. Це допоможе визначити її місце серед споріднених застосунків та задати напрямок подальшого дослідження.

Система буде рекомендувати книги на основі їх вмісту. Технічно книга є товаром, але її характеристики визначаються саме на основі ***тексту***. Метою системи є надання ***рекомендацій***. Тему книги (теги) визначено вручну на сайтах книгарень ***експертами*** в галузі літератури. За рівнем персоналізації рекомендації будуть як ***ефемерні*** (на основі тимчасового зацікавлення), так і ***постійні*** (засновані на стабільних інтересах та патернах поведінки). Вхідні дані про вектор інтересів користувача отримуються явно, а вихідні дані представляють собою рекомендації. В якості алгоритму обрано ***фільтрацію на основі контенту (Content-Based Filtering)***.

У таблиці 1.1 представлено результати класифікації рекомендаційної системи онлайн-книгарні на основі засобів машинного навчання.

Таблиця 1.1 – Характеристики системи що розробляється

Classification	Category
by domain	**text**
by purpose	**recommendation**
by whose opinion	**expert**
by personalization level	**ephemeral and persistent**
by interfaces	**explicit input, recommendation as output**
by recommendation algorithm	**content-based filtering**

Після класифікації рекомендаційної системи варто проаналізувати існуючі аналоги та визначити надійні джерела даних. Всі ці кроки дозволяють поступово викристалізувати характерні риси та унікальність кваліфікаційної роботи магістра.

1.4. Аналіз існуючих аналогів

Компанія «Amazon» одна з найперших почала активно розробляти та використовувати рекомендаційні системи. Відомо що рекомендаційна система генерує приблизно 35% їх прибутку [7]. Вказана компанія навіть створила серію електронних книг «Amazon Kindle», що вказує на те що торгівля книгами (паперовими та електронними) є важливою складовою цього бізнесу. Інженери постійно вдосконалюють рекомендаційну систему та активно використовують рейтинги, коментарі, дані про користувачів, дані про товари тощо для розробки складних алгоритмів для якнайточніших рекомендацій. Приклад інтерфейсу «Amazon» представлено на рисунку 1.8.

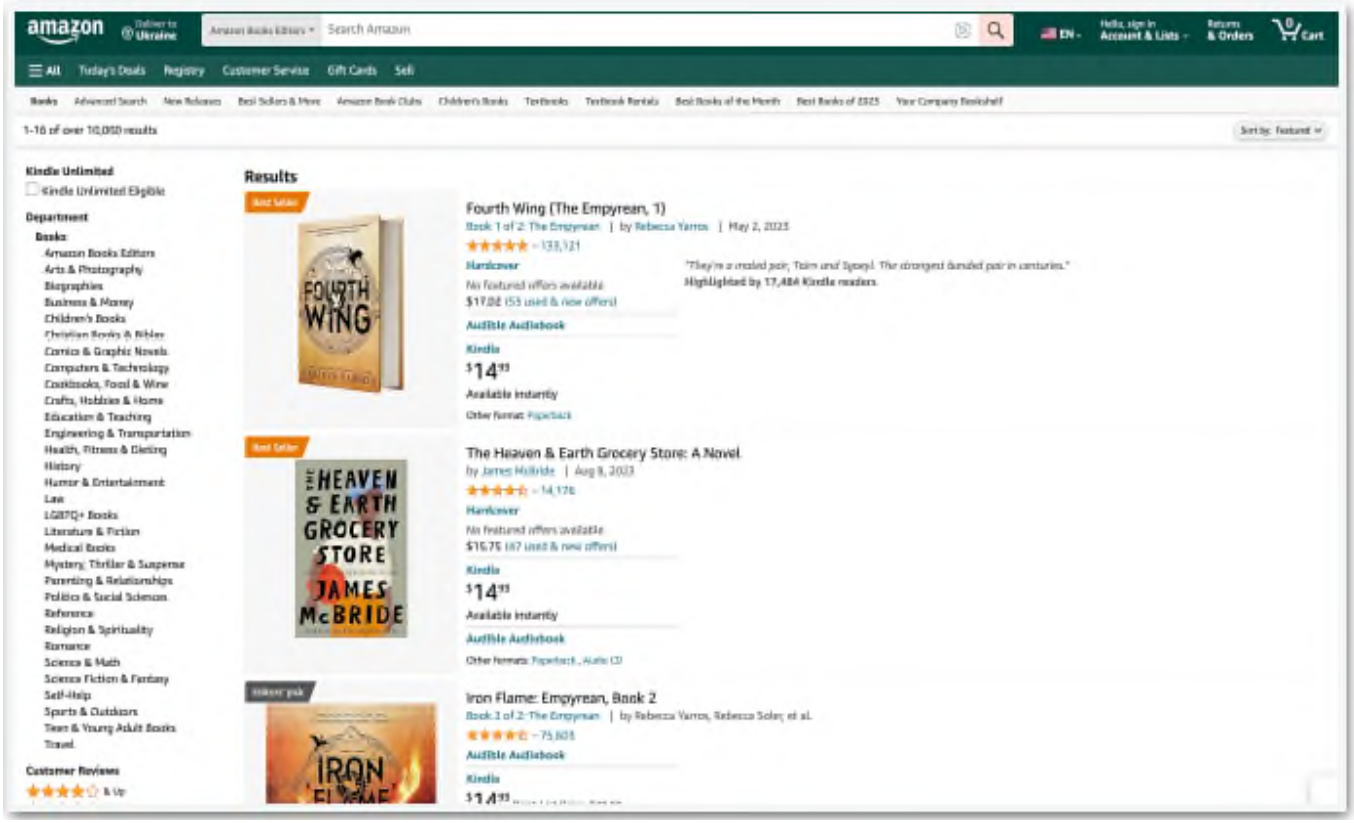

Рисунок 1.8 – Приклад інтерфейсу «Amazon»

«Amazon» спеціалізується на книгах англійською мовою, тому власна рекомендаційна система про українські книги за своїм вмістом буде суттєво відрізнятися від даного аналога. Крім того, система що розробляється має пропонувати варіанти з різних інтернет-магазинів, а не обслуговувати потреби лише одного бізнесу.

Інший цікавий ресурс – це портал *«Goodreads»*. Він є поєднанням

«автоматизованої» рекомендаційної системи та своєрідної соціальної мережі. На відповідному сайті користувачі вручну визначають список рекомендованої літератури, оцінюють книги, пишуть відгуки, обирають найбільш характерні цитати, формують книжкові полички тощо. Даний портал став настільки впливовим, що започаткував власну нагороду *«Goodreads Choice Awards»*.

Рекомендаційна система *«Goodreads»* орієнтована на аналіз даних про користувачів. Тобто формуються кластери читачів зі спільними інтересами (наприклад, з використанням алгоритму ***k-найближчих сусідів***) на основі подібності оцінок, обраних цитат, книжкових поличок тощо. Приклад інтерфейсу наведено на рисунку 1.9.

Рисунок 1.9 – Приклад інтерфейсу «Goodreads»

Система, що розробляється буде орієнтована на аналіз даних про книги та їх теги, а не на аналіз активності користувачів з наступним створенням груп зі спільними інтересами. Варто запозичити ідею про те що необхідно надати користувачам можливість явно вказувати свої інтереси. Інколи активність на сайті каже сама за себе, але зворотній зв’язок від користувача стосовно якості контенту та актуальності рекомендацій теж дуже цінний.

Існують універсальні технології для фільтрації контенту, які не прив’язані до певної компанії. Найбільш популярними з таких систем є пошукові двигуни *Apache Lucene, Apache Solr, Elasticsearch, Lunr.js* та *Xapian* [1]. Зазначені системи дозволяють шукати інформацію швидко та ефективно. Але кожна з них має свої переваги та недоліки. Крім того, у випадку датасетів та баз даних досить важко розробити алгоритм, який був би настільки гнучким щоб його можна було адаптувати абсолютно до будь-якої задачі. Тому реалізація власного алгоритму рекомендацій зазвичай краще відповідає вимогам і не містить нічого зайвого.

Apache Lucene надає широкі можливості для текстового пошуку та індексації. Ефективне керування індексами дозволяє підвищити швидкість пошуку. Серед недоліків варто назвати відсутність вбудованого сервера.

Apache Solr підтримує HTTP API, що полегшує інтеграцію з іншими системами. Даний застосунок є різновидом розподіленої системи, що робить її масштабованою. Недоліком є складна конфігурація.

Elasticsearch легко масштабується та має вбудовані засоби візуалізації даних, але в той самий час споживає значні ресурси при обробці великих датасетів.

Lunr.js швидкий та «легкий» пошуковий двигун, що ідеально підходить для вебсайтів та застосунків з невеликим обсягом даних. Працює безпосередньо на стороні клієнта, зменшуючи навантаження на сервер. Недоліком є обмежена функціональність.

Xapian може бути легко розширений, забезпечує високу продуктивність та індексацію даних. Відноситься до категорії бібліотек, що потребують доопрацювання для перетворення на повноцінну пошукову систему.

Отже, варто врахувати переваги та недоліки аналогічних систем з метою покращення власного застосунку. Аналіз існуючих аналогів довів унікальність системи та актуальність її розробки.

1.5. Визначення якості існуючих наборів даних

Надзвичайно важливою частиною будь-якого data-продукту є набір даних. Перш ніж перейти до пошуку датасетів необхідно зазначити критерії оцінки якості даних [8]:

1. Походження. Варто використовувати датасети, які створені гідними довіри організаціями. Якщо походження невідоме чи сумнівне, то є висока ймовірність помилок при зборі та обробці даних. Крім того, дані можуть бути навмисно чи ненавмисно спотворені, що робить результати роботи програми хибними. Існує наступна класифікація походження даних:

1.1. First-party (зібрано дослідником, зібрано власником data-продукту, отримано з баз даних організації, тощо).

1.2. Second-party (придбано у організації, яка спеціалізується на зборі даних, наприклад, придбано у компанії «DataOx»).

1.3. Third-party data (отримано в будь-який інший спосіб).

2. Повнота. Важливо визначити чи містить набір даних всі необхідні атрибути, наскільки часто у окремих записах містяться пусті «комірки» та чи є достатнім обсяг даних. Варто переконатися в тому що датасет є репрезентативною вибіркою. Чим менше обсяг, тим більше кожен окремий елемент спотворює загальну картину. Чим більший обсяг, тим більше часу та обчислювальних ресурсів вимагає обробка даних. Отже, варто визначити доцільну кількість даних виходячи з поставленого завдання.

3. Актуальність. Мається на увазі не лише уникнення застарілих даних, а і локалізація. Наприклад, не можна збирати дані про продажі книг в США для того, щоб проаналізувати стан видавничої справи в Україні. Також датасети містять рядки, тому використана мова має значення. Інколи застарілі дані необхідні для дослідження динаміки розвитку певної ґалузі, але це точно не випадок рекомендаційної системи. Наприклад, з 2000-го року видано надто багато нових творів щоб рекомендувати користувачам лише застаріли варіанти.

4. Достовірність. Помилки у даних можуть з’явитися через людський фактор, неправильну обробку чи недостатньо об’єктивний спосіб збору даних.

Наприклад, якщо йдеться про збір даних за допомогою опитування, то постановка питань часто визначає відповідь. Варто запитувати нейтрально та без примусу. Але якщо дослідники навіть у питаннях намагалися схилити респондента до певної відповіді, то таку відповідь не можна вважати чесною та об'єктивною.

Рекомендаційні системи книгарень є поширеною темою для досліджень, тому не дивно що існують досить популярні датасети, на основі яких можна розробити свій варіант рекомендаційної системи. В результаті аналізу існуючих джерел даних відзначено наступні варіанти:

1. Дані про книги та їх рейтинг з «Amazon» [2]. Вказаний набір даних містить відомості про продажі книжок з 1995 по 2013 рік. На жаль, датасет занадто застарілий та не стосується української видавничої справи. Крім того, походження датасету не є прийнятним.

2. Дані про буккросинг від «GroupLens» [3]. Датасет застарілий (за 2004 рік), не стосується електронної комерції та не охоплює книги українською мовою. Отже, зовсім не підходить до теми кваліфікаційної роботи магістра. Але збір даних здійснено гідною довіри спільнотою, тому походження та достовірність є перевагами вказаного датасету.

3. Набір даних за 2017 рік для рекомендаційної системи на основі даних порталу «Goodreads» [4] найкраще пасує до теми КРМ за своєю структурою та вмістом, але його недоліком є відносна застарілість, відсутність книг українською мовою та походження.

Всі наведені вище датасети не підходять для рекомендаційної системи, що розробляється. Найкращим варіантом є самостійний збір даних (використання first-party data). В цьому випадку датасет буде дійсно актуальний. Можна самостійно пересвідчитись у достовірності даних, правильно їх обробити та зібрати саме таку кількість інформації, яка потрібна для розв'язку поставленого завдання. Крім того, дані будуть стосуватися саме українських реалії та найновіших українських видань. Отже, самостійно зібрані дані нададуть ще більшу унікальність кваліфікаційній роботі магістра.

Для збору даних необхідно здійснити процес сканування вебсторінок (web scraping) відомих українських онлайн-книгарень. Неможливо зібрати дані про абсолютно всі доступні книги. Література часто продається на вторинному ринку, деякі видавництва не мають власного сайту, інші мають занадто вузьку спеціалізацію і не є характерними тощо. Тому варто створити репрезентативну вибірку.

На основі практичного досвіду і дослідження інформації про цінності та спеціалізацію українських видавництв складено наступний список книгарень, чию продукцію варту включити в датасет: «Наш Формат», «Фабула», «Видавництво Старого Лева (ВСЛ)», «Vivat», «Комора» та «Лабораторія». Всі ці видавництва мають власні сайти, тому збір даних відбувається з декількох джерел. Крім того, кожен видавець розробив свій власний спосіб представлення інформації, тому процес об'єднання даних різного формату в один набір є важливим етапом розробки рекомендаційної системи.

Одним з найголовніших недоліків процесу сканування вебсторінок є залежність від сторонніх ресурсів. Саме тому датасет статичний. Небезпечно в режимі реального часу читати оновлення web-сторінок та збирати дані, оскільки немає жодних гарантій що в цей момент сайт буде працювати чи розробники не надумають внести зміни у макет сторінок. Сформований датасет можна вважати окремим продуктом (deliverable), який придатний для повторного використання іншими розробниками.

ВИСНОВКИ ДО РОЗДІЛУ

В ході розробки першого розділу кваліфікаційної роботи магістра здійснено аналіз предметної сфери. В результаті визначено місце застосунка серед інших подібних систем та доведено його унікальність. Також акцентовано унікальні риси системи, що необхідно для подальшого проєктування і специфікації вимог. Аналіз існуючих датасетів про книги показав що найкращим варіантом є самостійний збір даних. На основі інформації з даного розділу можливо здійснити проєктування та моделювання системи.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Модель даних

Система рекомендує релевантні книги читачам. Отже, «Читач» та «Книга» є головними сутностями у рекомендаційній системі онлайн-книгарні на основі засобів машинного навчання. Для відображення зв’язків між сутностями у системі використовується ERD (Entity Relationship Diagram). На рисунку 2.1 представлено різновиди нотацій ER-діаграм.

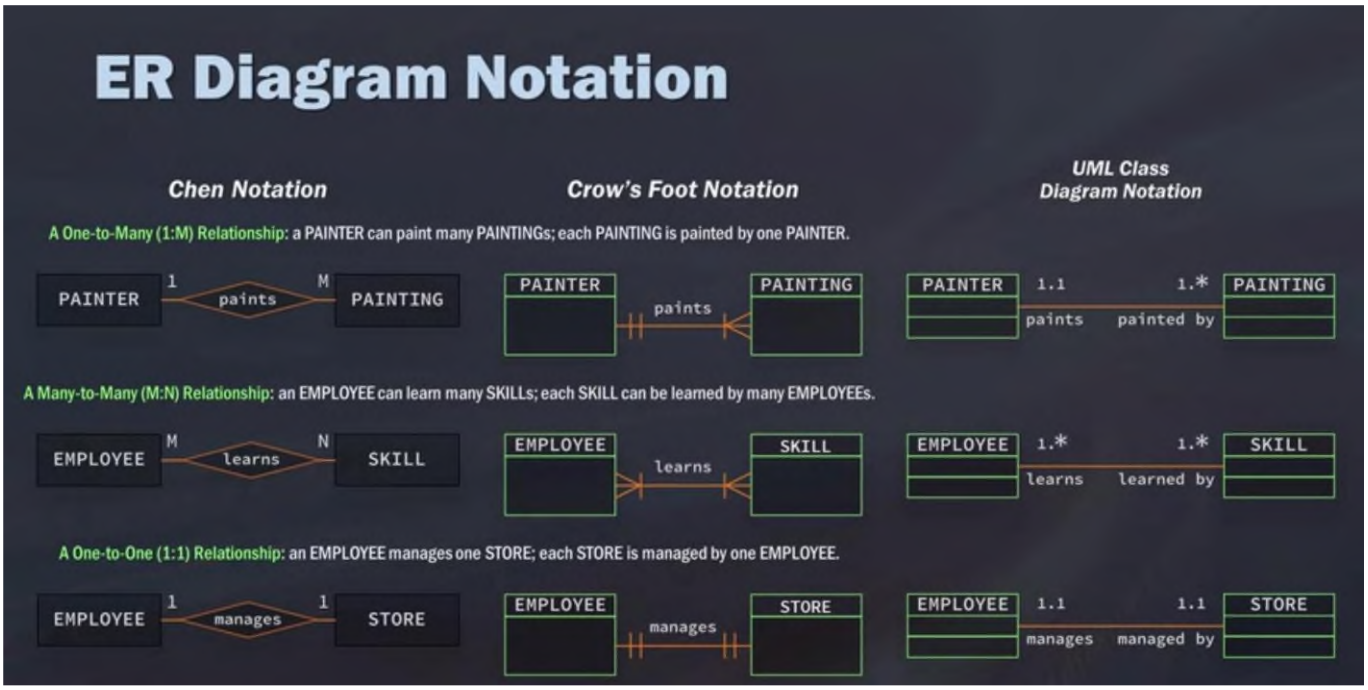

Рисунок 2.1 – Нотації ER-діаграм

В даній роботі *Chen Notation* використана для демонстрації логічної моделі даних. Логічна модель має високий рівень абстракції та призначена для загального розуміння сутності системи та використаного набору даних. Вказано лише сутності без атрибутів та зв’язки між ними.

Crow’s Foot Notation характеризує фізичну моделі даних. Дана діаграма більш детальна. У фізичній моделі даних вказуються типи даних, обмеження цілісності, первинні та вторинні ключі тощо.

В даній системі логічна та фізична модель даних дещо *відрізняються*, оскільки база даних обмежується лише необхідною для рекомендацій інформацією (а логічна модель описує предметну сферу без прив’язки до конкретного застосунка).

Будь-яка книга одночасно є і контентом, і товаром (предметом). Тобто письменник (*author*) створює (*creates*) унікальний твір (*content*). Даний твір може містити не лише текст, а і, наприклад, зображення у випадку коміксів. Теги (*tags*) характеризують контент. Тобто не важливо хто є видавцем та чи здійснено переклад. Задана тегами тема твору залишається незмінною в будь-якому виданні. Кожне видавництво (*publisher*) «відтворює» (*publishes*) контент у свій власний спосіб, тобто здійснює переклад, розробляє дизайн обкладинки, обирає формат, колір та матеріал сторінок, здійснює редагування тощо. Тобто для одного твору (*content*) може існувати безліч реалізацій (*book*). Саме від видавця залежить ціна книги, кількість сторінок та формат. Команда редакторів, коректорів, верстальників та інших фахівців (*staff*) безпосередньо створює (*make up*) книгу як товар. Читач (*reader*) взаємодіє (*interacts*) з книгою у різні способи – купує, додає в список улюблених книжок або корзину, ставить лайки тощо. Всі ці сутності та зв'язки між ними продемонстровано за допомогою ER-діаграми на рисунку 2.2.

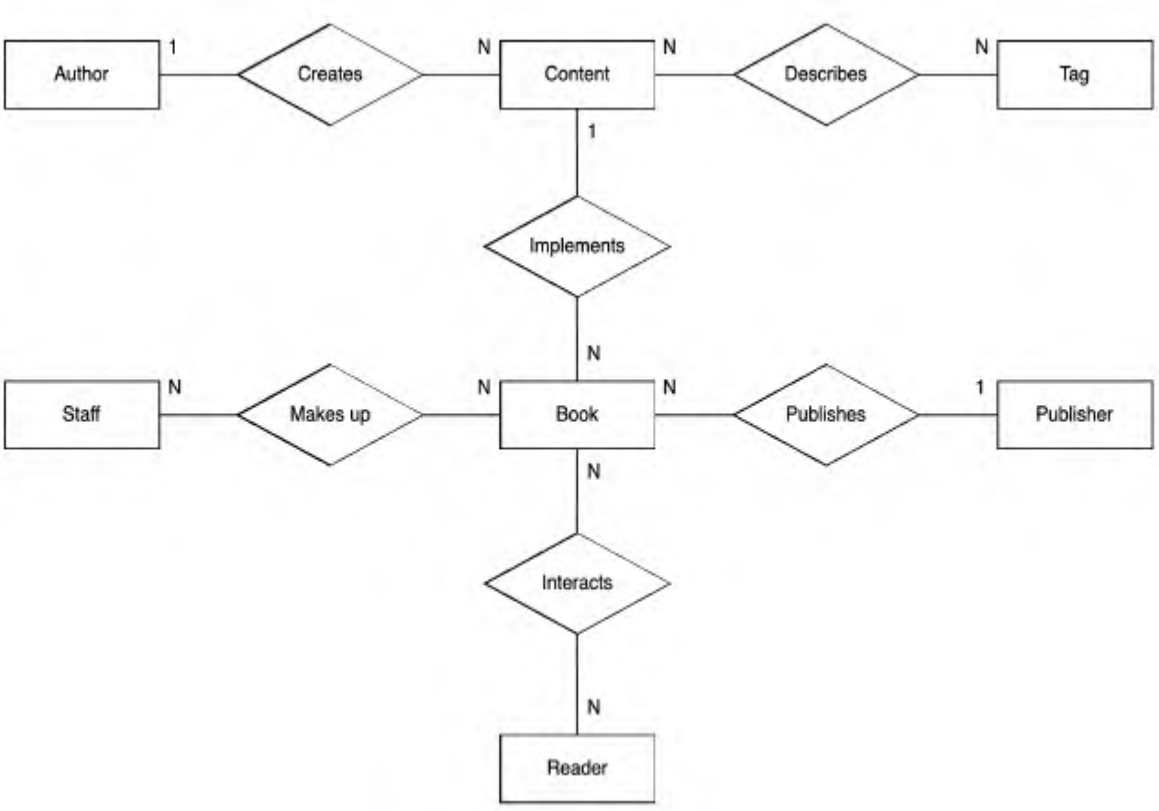

Рисунок 2.2 – ER-діаграма (логічна модель даних)

Логічна модель даних має бути уточнена та адаптована до проєкту за допомогою фізичної моделі даних. Фізична модель даних (рис. 2.3) має враховувати

специфіку та вимоги реляційних баз даних. В даному випадку варто визначити атрибути всіх сутностей та зазначити типи даних. Реляційна модель мінімізує дублювання інформації. Повторення небезпечні не лише через марнування ресурсів, а і тому що при внесенні змін можна помилково оновити тільки частину записів. Отже, з часом нагромаджуються помилки та суперечності. Реляційна модель дозволяє представити набір даних у вигляді таблиць, пов'язаних за допомогою первинних (однозначно ідентифікує запис в таблиці) та зовнішніх (однозначно ідентифікує запис в іншій таблиці, своєрідне посилання) ключів. Між таблицями можуть бути відношення наступних типів [9]:

1. *One-to-many* (наприклад, один автор для декількох книг).
2. *Many-to-many* (наприклад, декілька тегів для декількох книг).
3. *One-to-one* (наприклад, один директор для одного видавництва).

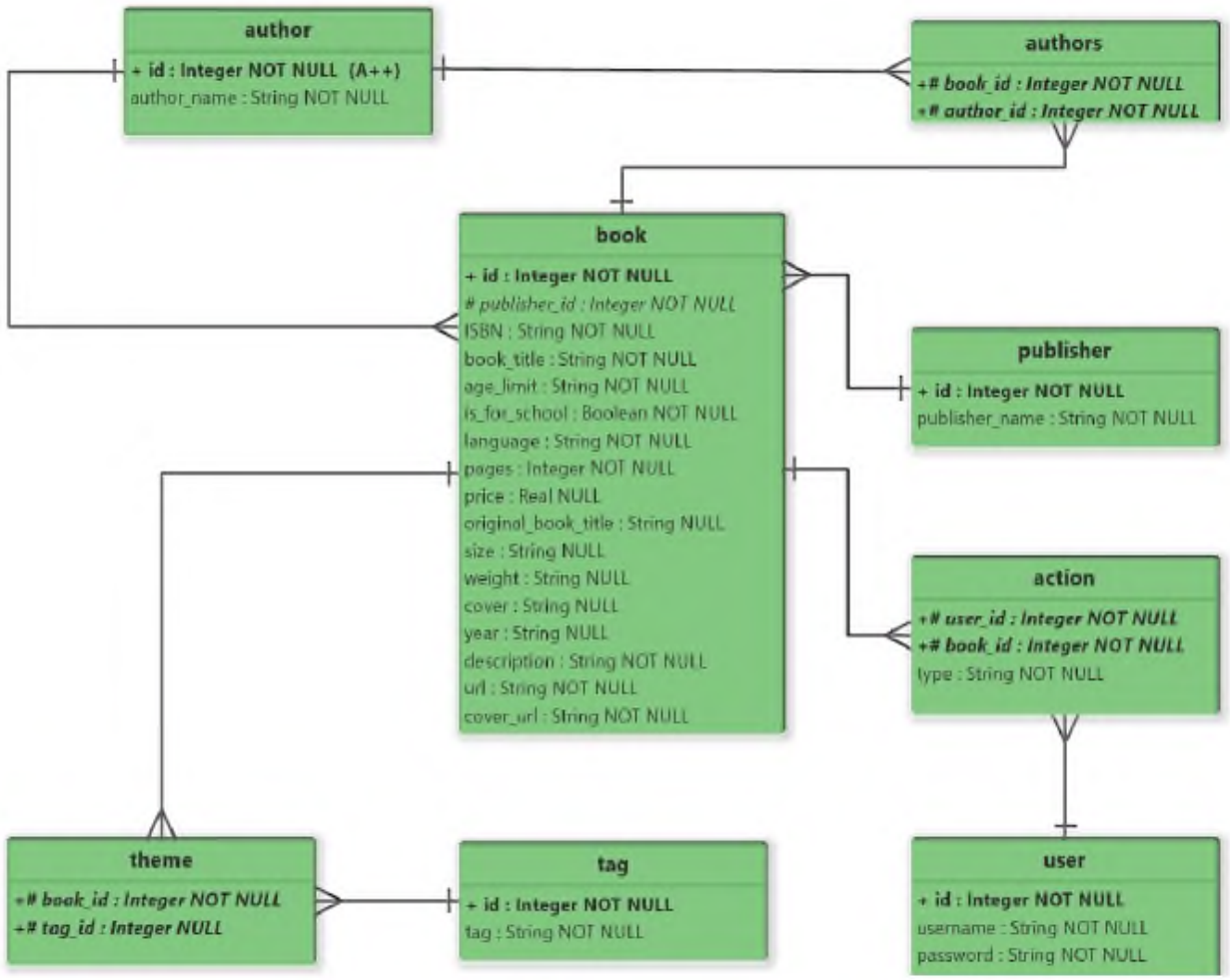

Рисунок 2.3 – ER-діаграма (фізична модель даних)

На основі розробленої фізичної моделі даних можливо згенерувати таблиці у базі даних. Для даної роботи обрано SQLite. Дані будуть записуватися в БД з використанням ORM.

2.2 Моделювання процесів у інформаційній системі

Мова моделювання IDEF0 дозволяє продемонструвати послідовність процесів в інформаційній системи з зазначенням вхідних даних, результуючих даних, механізмів та правил. Найголовнішим процесом у рекомендаційній системі онлайн-книгарні на основі засобів машинного навчання є процес «Рекомендувати книги». Він вимагає перелік всіх книг та відомості про читацьку активність в якості початкових даних, а в результаті повертає список рекомендацій. В процесі рекомендування задіяні онлайн-книгарня, рекомендаційна система та користувач. Закон «Про авторське право і суміжні права» [10] захищає авторське право та вміст книг від незаконного використання. Закон «Про захист персональних даних» [11] захищає приватність користувачів. Відповідна діаграма представлена на рисунку 2.4.

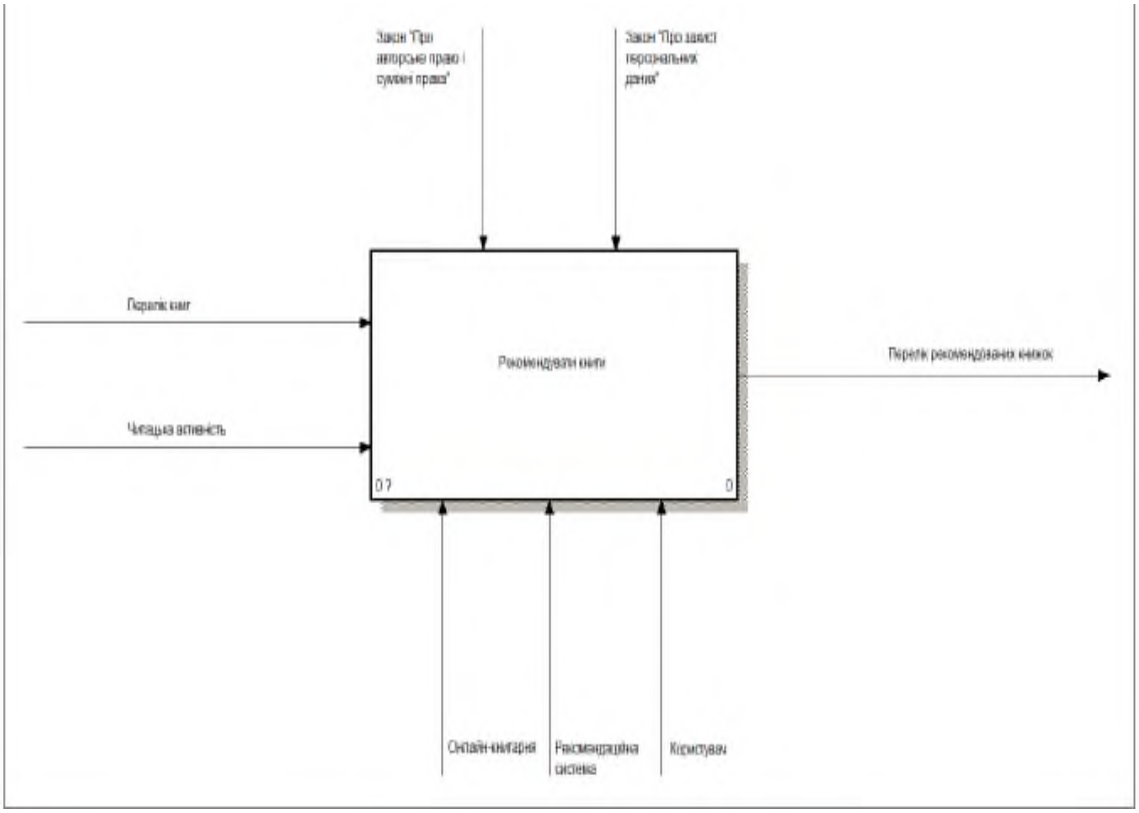

Рисунок 2.4 – Діаграма IDEF0

Декомпозиція головного процесу дозволяє уточнити кроки надання рекомендацій. В даному випадку основний процес розділено на 4 підпроцеси.

Для надання рекомендацій необхідно зібрати дані про книги, призначити всім книгам теги, визначити читацькі вподобання та, нарешті, скласти перелік рекомендованих книг (рис. 2.5).

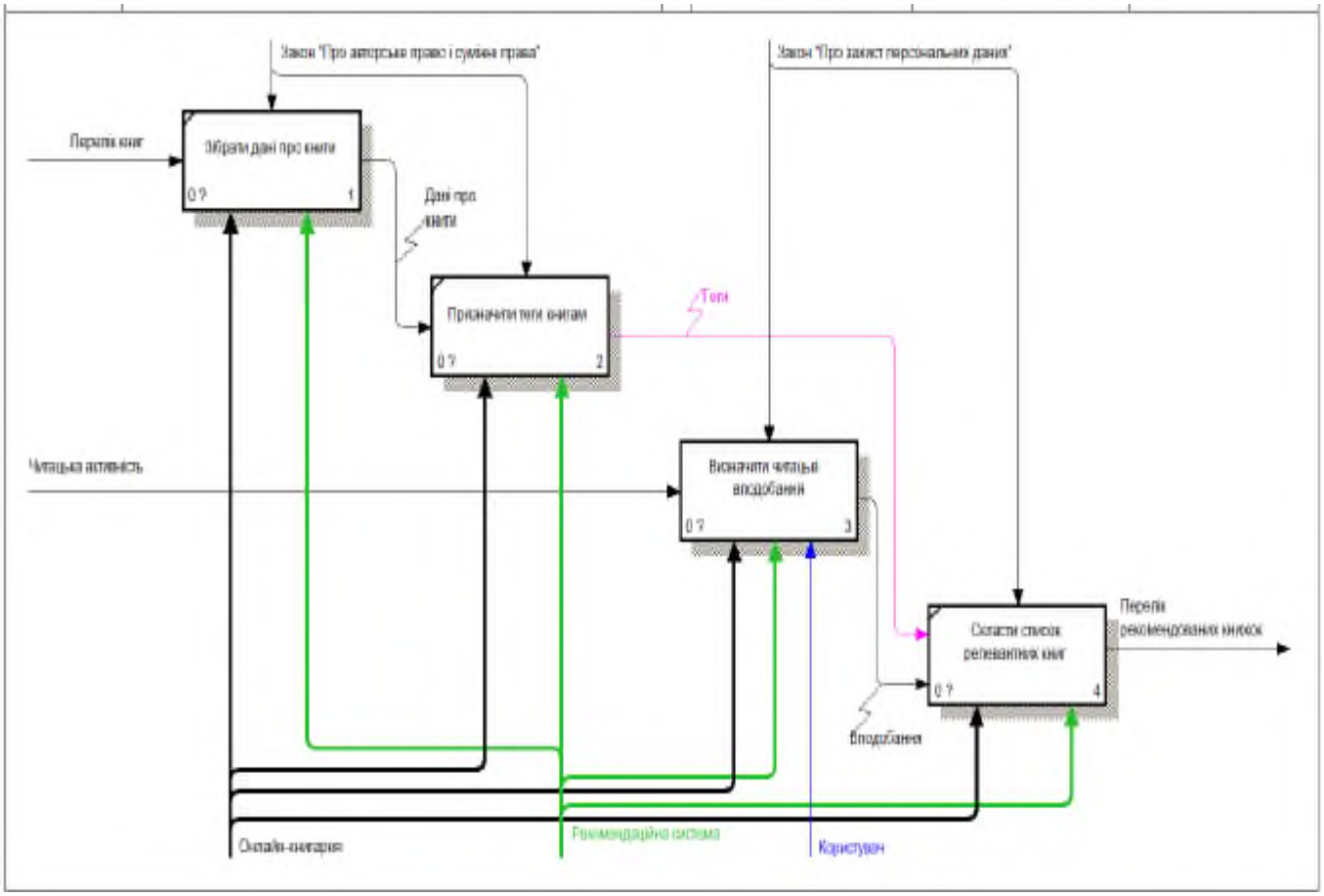

Рисунок 2.5 – Діаграма IDEF0

В межах даної системи використовується статичний датасет, тому процеси «Зібрати дані про книги» та «Призначити теги книгам» відбуваються одноразово. Динамічність системі надає задача «Визначити читацькі вподобання». Даний процес відбувається безперервно, оскільки активність на сайті доповнює профіль користувача та дозволяє вчасно реагувати на зміну вподобань. Що стосується списку релевантних книг, то не варто демонструвати надто багато рекомендацій. Кореляцій між вектором інтересів та тематикою книги повинна бути ***статистично значима***. Можна стверджувати що майже всі доступні книги певною мірою відповідають інтересам, але лише деякі з цих книг дійсно релевантні та актуальні.

Тобто складається рейтинг книг за рівнем відповідності інтересам та обирається топ екземплярів.

Діаграма потоків даних (DFD) дозволяє продемонструвати як дані переходять з одного процесу в інший. Найголовнішими сховищами в системі є HTML-файли (з яких «витягуються» дані під час web scraping) та база даних (у яку записуються оброблені дані). Відповідна діаграма представлена на рисунку 2.6.

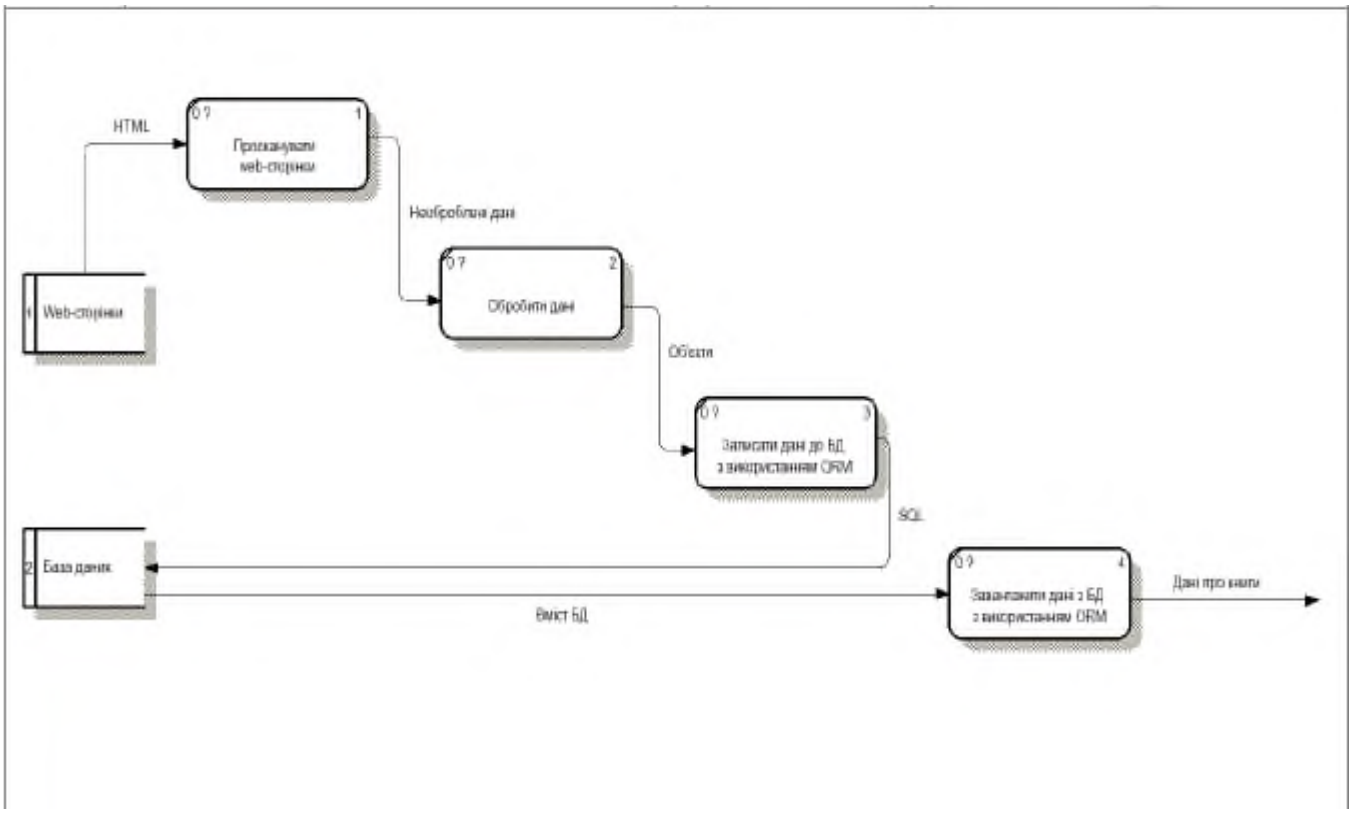

Рисунок 2.6 – Діаграма DFD

Важливо зазначити що зібрані в процесі сканування web-сторінок дані не відразу придатні для використання в рекомендаційній системі. Варто розробити та виконати SQL-запити та код мовою Python для очищення та форматування даних. ORM полегшує процес перетворення об’єктів на записи в базі даних. ORM не потрібно розробляти самостійно, оскільки існують якісні, протестовані та відлагоджені реалізації.

Декомпозиція процесу за допомогою діаграми IDEF3 демонструє динамічність системи (рис. 2.7). В даному випадку йдеться про те, що рекомендаційна система реагує на три типи активності користувача:

1. Налаштування параметрів фільтрації вручну (оновити перелік книг).

2. Оцінювання книги (в даному випадку оновити вектор інтересів).

3. Перегляд детальної інформації про окрему книгу (продемонструвати подібні книги).

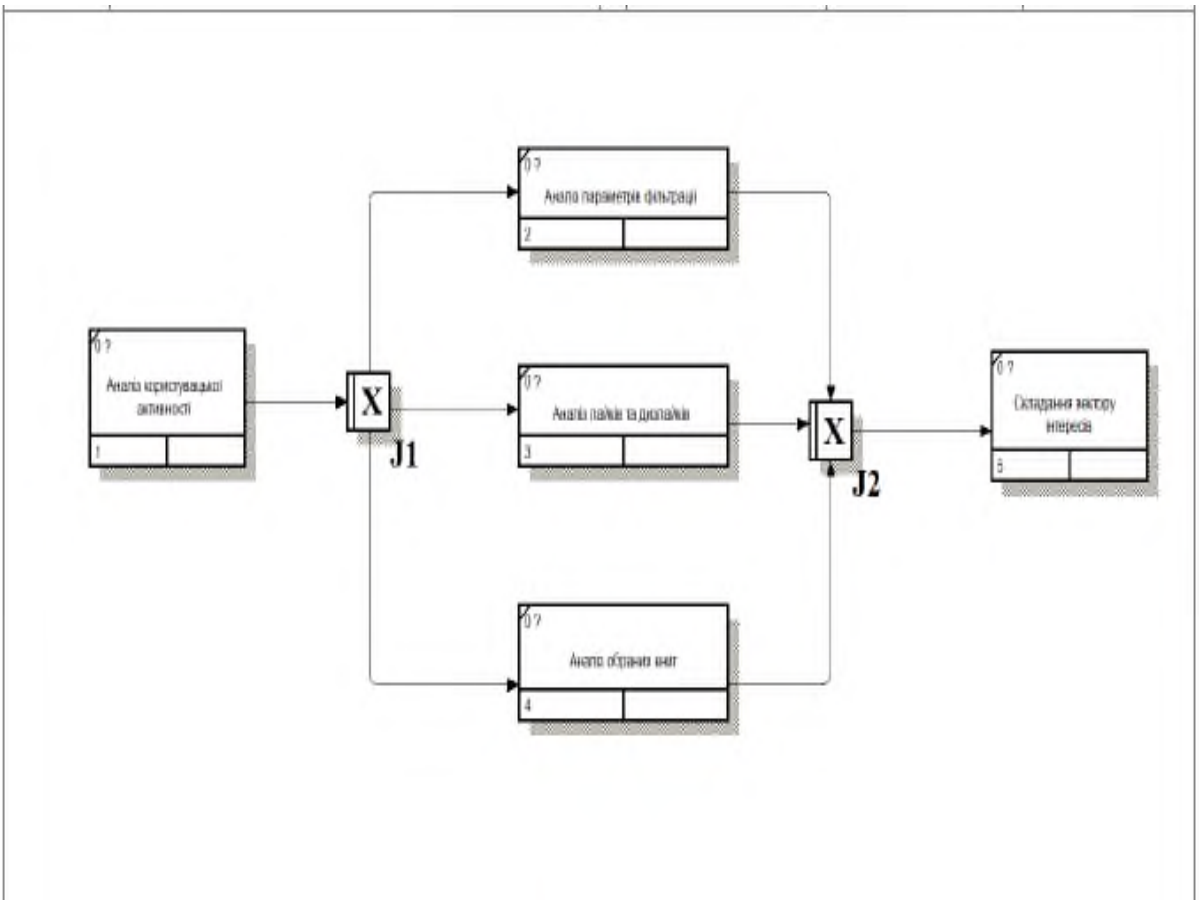

Рисунок 2.7 – Діаграма IDEF3

В наведених вище випадках алгоритм формування переліку рекомендованих книг дещо відрізняється. Але в будь-якому разі користувач має побачити видання, які якнайкраще відповідають вектору вподобань.

На основі розроблених діаграм можливо здійснити специфікацію вимог до система. Вимоги поділяються на функціональні та нефункціональні. IDEF0, IDEF3 та DFD акцентують увагу саме на процесах та функціональності, тому варто розпочати з визначення функціональних вимог.

2.3 Вимоги до системи

Функціональні вимоги:

1. Реєстрація та авторизація користувача.

2. Збір даних про книги за допомогою web scraping.

3. Збір даних про активність користувача на сайті.

4. Формування вектору інтересів користувача.

5. Надання рекомендацій згідно з інтересами (persistent recommendation).

6. Підбір книг, подібних до обраної книги (ephemeral recommendation)

7. Фільтрація та пошук даних за різними параметрами (назва, автор, видавець, рік видання, тощо).

Нефункціональні вимоги:

1. Персональні дані користувачів мають бути надійно захищені (Security and Privacy).

2. Максимальна затримка у роботі системи при розрахунку рекомендацій не повинна перевищувати 3 секунди (Productivity).

3. Необхідно регулярно робити backup бази даних на випадок виникнення необхідності відновити систему (Recoverability).

Обмеження:

1. Код застосунку має бути розроблений мовою програмування Python.

2. В якості database engine необхідно використати SQLite.

2.4 Проектування архітектури системи

Проєктування архітектури системи є високорівневим рішенням. Воно визначає подальший хід розробки. Існують різні архітектурні стилі программного забезпечення, і кожен з них має свої переваги та недоліки. Далі наведено декілька поширених архітектурних стилів:

1. Client-Server. Система розділена на дві частини: клієнтську (яка надає користувацький інтерфейс) і серверну (яка обробляє запити і зберігає дані).

2. Multitier Architecture. Система розділена на декілька шарів (презентаційний, бізнес-логіка, дані, тощо), кожен з яких виконує свою функцію.

3. Microservices. Система розділена на невеликі автономні сервіси, які виконують конкретні функції і комунікують між собою за допомогою API.

4. Event-Driven Architecture. Обробка подій сприймається як центральний елемент. Компоненти реагують на виникнення подій.

5. Pipeline. Вхідні дані обробляються через послідовність фільтрів. Кожен фільтр використовує конкретну функцію.

6. Service-Oriented Architecture. Система розділена на автономні сервіси, які взаємодіють між собою через стандартизовані інтерфейси.

7. Hierarchical Architecture. Система організована у вигляді ієрархії, де кожен рівень відповідає за визначений аспект функціональності.

8. Monolithic Architecture. Всі компоненти програми об'єднані в одному великому блоці.

Дані архітектурні стилі можуть використовуватися окремо чи в комбінації залежно від конкретних вимог та характеристик програмного продукту. Будь-який складний застосунок складається з багатьох класів, функцій та модулів. Дані елементи необхідно організувати у такий спосіб щоб система була масштабованою, гнучкою та відмовостійкою [12]. Якісна архітектура вимагає значних витрат ресурсів на початку проєктуванні, але економить зусилля та час у майбутньому.

Архітектуру системи зазвичай представляють за допомогою діаграми компонентів (рис. 2.8).

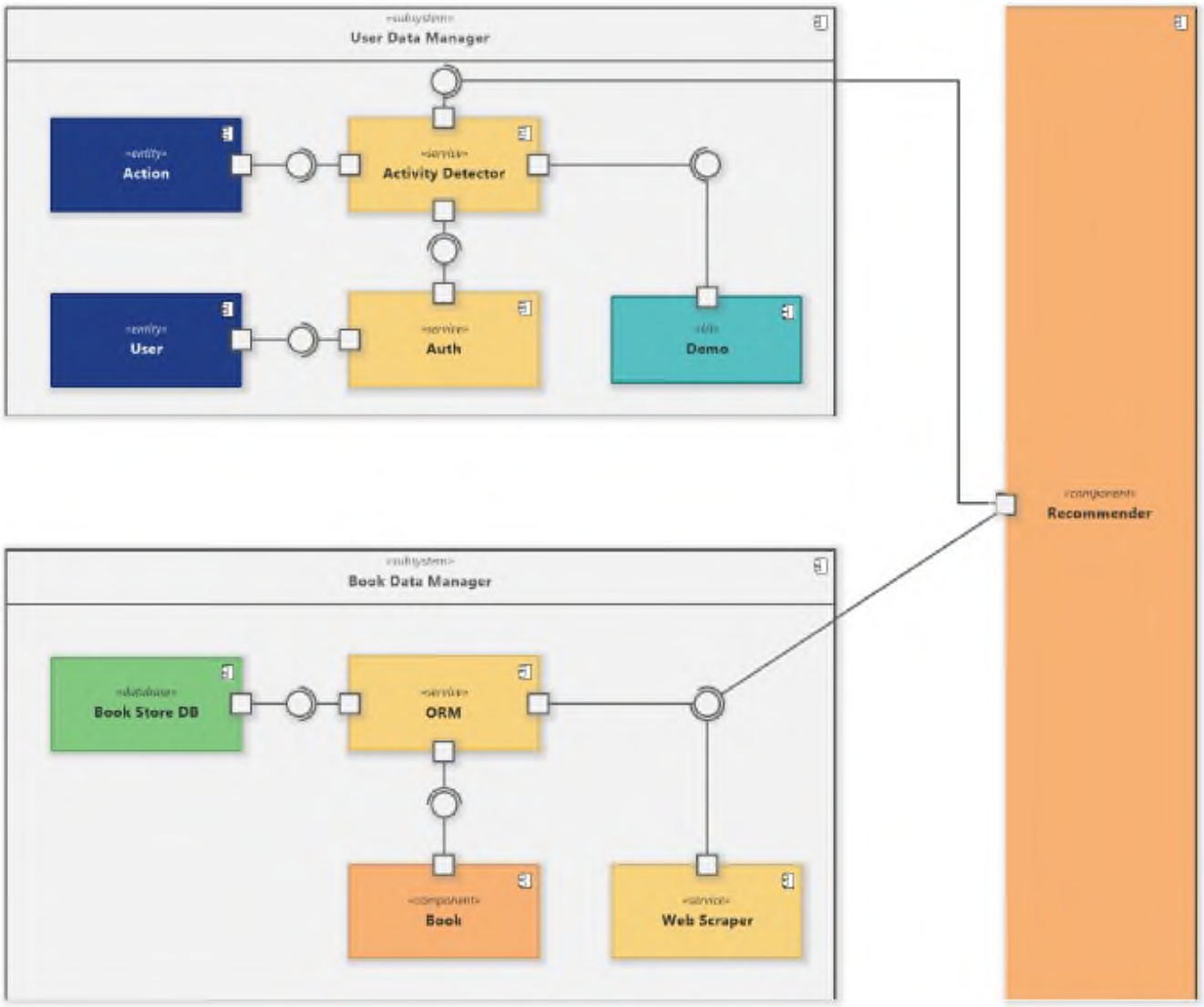

Рисунок 2.8 – Діаграма компонентів

Запропоноване рішення містить незалежні підсистеми та компоненти. В даному випадку обраний підхід до розробки програмного забезпечення дозволяє здійснити декомпозицію складної задачі, поступово нарощувати складність та тестувати систему на кожному етапі розробки [13].

ВИСНОВКИ ДО РОЗДІЛУ

В ході створення другого розділу кваліфікаційної роботи магістра виконано проєктування рекомендаційної системи. За допомогою ERD здійснено моделювання структури датасету, а DFD продемонструвала потоки даних в системі. Процеси змодельовано за допомогою IDEF0 та IDEF3. Визначено вимоги та архітектуру системи.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Етапи реалізації рекомендаційної системи

Машинне навчання дозволяє запрограмувати комп'ютер у такий спосіб аби він вчився на основі великих обсягів даних. Принцип роботи рекомендаційної системи у найбільш узагальненій формі представлено на рисунку 3.1.

Рисунок 3.1 – Принцип роботи рекомендаційної системи

При розробці рекомендаційної системи зазвичай виконується наступна послідовність дій:

1) Попередня обробка даних (pre-processing). Тут йдеться як про стандартні процедури (збір, очищення та трансформація даних), так і про застосування більш складних технік. Наприклад, якщо система будується на основі колаборативної фільтрації зазвичай застосовують факторизацію матриці (matrix factorization) або нормалізацію значень рейтингу об'єктів.

2) Налаштування гіперпараметрів (hyper-parameters tuning). Будь-яка модель машинного навчання передбачає можливість гнучкого налаштування гіперпараметрів для підвищення точності прогнозування. Гіперпараметри використовуються для керування процесом навчання.

3) Навчання моделі та прогнозування (model training & prediction). На даному етапі здійснюється використання моделі машинного навчання для фільтрації та складання рейтингового переліку об'єктів відповідно до даних про вподобання користувача та властивості рекомендованих об'єктів.

4) Обробка результатів (post-processing). Згенеровані рекомендації можуть містити надто багато рекомендованих елементів або пропонувати користувачу те, що вже було придбано і більше не актуально. Тому варто обрати лише перші N

елементів та виключити дублікати. Також можливі інші варіанти обробки прогнозу в залежності від поставленого завдання.

5) Оцінка точності прогнозування (evaluation). У випадку інших алгоритмів машинного навчання зазвичай йдеться про те що дані заздалегідь було розділено на навчальний та тестовий набір. На основі порівняння прогнозованих моделлю значень та фактичних даних з тестового набору можливо зробити висновки про точність. Але рекомендаційні алгоритми краще вчити та тестувати на тій самій вибірці, штучно створюючи дефіцит даних (рис. 3.2).

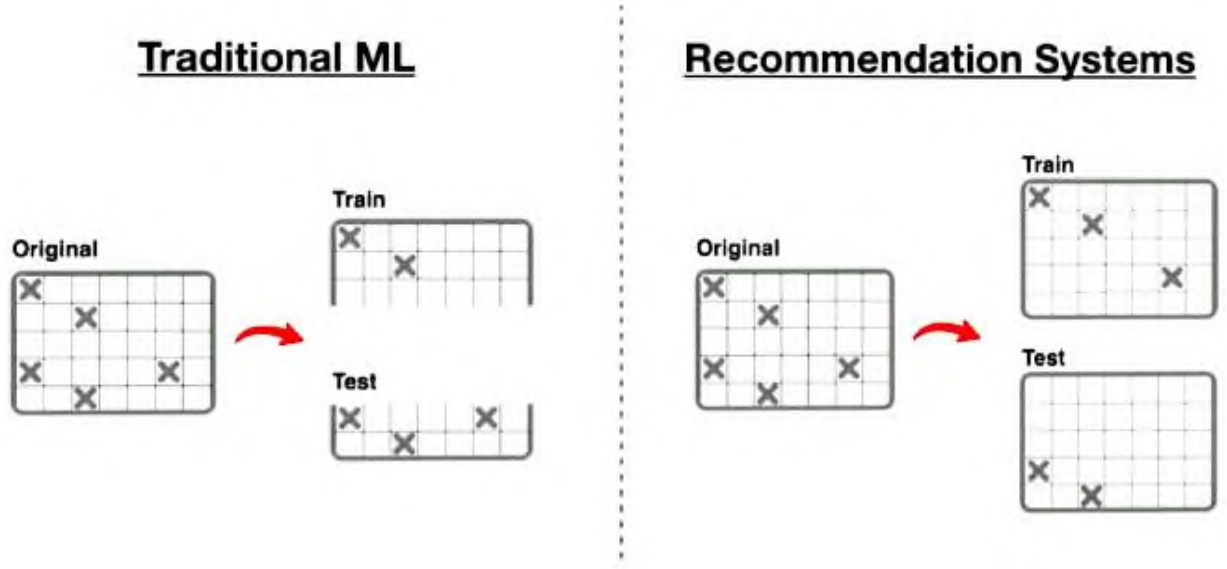

Рисунок 3.2 – Порівняння принципів розділення набору даних [20]

Дані про людські вподобання надто унікальні та різнопланові для того, щоб створити репрезентативні вибірки при розділенні на навчальний та тестовий набір. Рекомендаційні системи краще всього додатково випробовувати за допомогою A/B тестування та справжніх відгуків. Наприклад, можна запропонувати оцінювати релевантність наданих альтернатив під час взаємодії з системою.

Для оцінки точності рекомендацій використовуються такі метрики як RMSE, recall, precision та F1.

Метрика RMSE (Root Mean Square Error) є однією з найпопулярніших метрик оцінки точності прогнозування у моделях МН. Формула 3.1 демонструє принцип розрахунку даної метрики.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \qquad (3.1)$$

де: $\hat{y}_i$ - прогнозоване значення;

y_i - фактичне значення;

n – кількість порівнянь прогнозованих та фактичних значень.

Розраховане число показує середню різницю між фактичним та прогнозованим значенням. Оцінка того наскільки критичним є отриманий результат залежить від предметної сфери. Наприклад, при прогнозуванні ціни автомобіля похибка у 15 дол. США може вважатися незначною, але при прогнозуванні цін на книги це неприпустимо велике відхилення. На рисунку 3.2 представлено варіанти співвідношення прогнозів з фактичними значеннями.

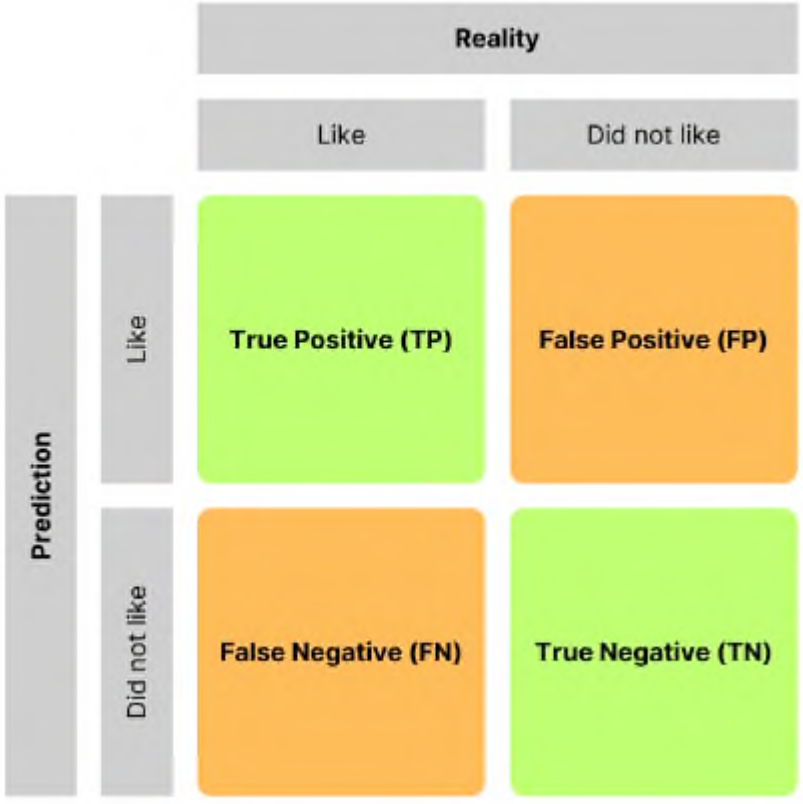

Рисунок 3.3 – Інтерпретація прогнозів

Формула 3.2 дозволяє розрахувати precision. Дана метрика показує яка частка рекомендацій може вважатися релевантною.

Формула 3.3 призначена для розрахунку recall. В даному випадки йдеться про оцінку того наскільки поведінка користувача є непередбачуваною для системи (тобто враховується кількість випадків коли користувач вподобав нерекомендовані елементи).

$$precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$recall = \frac{TP}{TP + FN} \tag{3.3}$$

де: TP – кількість рекомендованих елементів, які користувач вподобав;

FP – кількість рекомендованих елементів, які користувач не вподобав;

FN – кількість нерекомендованих елементів, які користувач вподобав.

Метрика F1 поєднує precision та recall. Формула 3.4 демонструє принцип розрахунку даної метрики.

$$F1 = \frac{precision \cdot recall}{precision + recall} \tag{3.4}$$

Отже, наведені метрики у поєднанні з результатами тестування та відгуками від користувачів дозволяють оцінити якість рекомендаційної системи.

3.2 Фільтрація на основі контенту

Досить популярним алгоритмом для аналізу тексту та визначення ваги вектору інтересів користувача є TF-IDF (Term Frequency Inverse Document Frequency). Формула 3.5 використовується для визначення оцінки (*score*) певного ключового слова.

$$TF * log(\frac{Q_{docs}}{Q_{docswithterm}}) \tag{3.5}$$

де: TF – частота появи ключового слова в документі

Q_{docs} - загальна кількість документів

$Q_{docswithterm}$ - кількість документів, які містять даний термін

Отже, даний алгоритм оцінює документи (будь-які тексти або інші об'єкти, що характеризуються тегами або ключовими словами) на основі того як сильно цей документ пов'язаний з шуканим терміном. Чим частіше термін зустрічається у документі і чим більш рідкісним він є у всій колекції тим вища оцінка (релевантність). Тобто даний алгоритм виключає stopwords (артиклі, прийменники та інші поширені, але не навантажені унікальним змістом слова) з переліку тегів.

TF-IDF часто використовують для визначення ключових слів. Якщо співставити визначені даним алгоритмом оцінки документів та характер взаємодії користувачів з цими документами, то можна у такий спосіб розрахувати вектор інтересів.

При фільтрації на основі контенту з використанням ключових слів алгоритм визначення подібності працює на основі наступних принципів.

1. Дана множина ключових слів.

2. Об'єкт має певну позицію в заданій ключовими словами матриці.

3. Кожен користувач має вектор інтересів, який також заданий в межах множини всіх можливих ключових слів.

4. Ступінь подібності між вектором користувацьких інтересів та вектором тегів об'єкта визначається тим наскільки близько один до одного знаходяться ці вектори.

Для співставлення вектору інтересів та профілю об'єкта зазвичай використовується косинус подібність (cosine similarity). Формула 3.6 демонструє принцип розрахунку косинуса подібності.

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (3.6)$$

де: A_i - координати вектора A

B_i - координати вектора B

Косинус подібності допомагає порівняти не величини, а напрямки двох векторів. Якщо вектори подібні (співнаправлені) їх косинус подібності дорівнює 1. Якщо вектори перпендикулярні один до одного, то їх косинус подібності дорівнює 0. Також за допомогою косинуса подібності можливо порівнювати різні об’єкти між собою (наприклад, розрахувати подібність між книгами).

Існують різні підходи до представлення цих векторів. Вектори можуть бути зваженими, отже тоді деякі ключові слова важливіші за інші. Або ці вектори можуть містити лише значення 1 або 0, тобто вказувати на наявність певного ключового слова в описі об’єкта чи в векторі інтересів. В такому випадку невідомо якою мірою це ключове слово характеризує об’єкт чи інтереси користувача.

У спрощеній формі процес розрахунку рекомендацій представлено на рисунку 3.4. У наведеному прикладі використовується тестовий набір даних для демонстрації принципу роботи рекомендаційного алгоритму на основі фільтрації контенту.

Матриця, яка описує документи за допомогою тегів:											Кількість тегів	Взаємодія з документами		Прогноз	
	baseball	economics	politics	Europe	Asia	soccer	war	security	shopping	family		User 1	User 2	Pred1	Pred2
doc1	1	0	1	0	1	1	0	0	0	1	5	1	-1	4	-4
doc2	0	1	1	1	0	0	0	1	0	0	4	-1	1	-4	10
doc3	0	0	0	1	1	1	0	0	0	0	3			2	0
doc4	0	0	1	1	0	0	1	1	0	0	4		1	-3	8
doc5	0	1	0	0	0	0	0	0	1	1	3			-1	1
doc6	1	0	0	1	0	0	0	0	0	0	2	1		3	1
doc7	0	0	0	0	0	0	0	1	0	1	2			-1	2
doc8	0	0	1	1	0	0	1	0	0	1	4			-2	4
doc9	0	0	0	0	0	1	0	0	1	0	2			3	-2
doc10	0	1	0	0	1	0	1	0	0	0	3			-3	1
doc11	0	0	1	0	1	0	0	0	1	0	3			0	1
doc12	1	0	0	0	0	1	1	0	0	0	3		-1	4	-4
doc13	0	0	1	1	1	0	0	1	0	0	4			-2	7
doc14	0	1	1	1	0	0	0	0	1	0	4			-2	7
doc15	0	0	0	1	0	1	1	1	0	0	4			0	4
doc16	1	0	0	0	0	1	0	0	1	0	3	1		6	-4
doc17	0	1	1	1	0	0	0	1	0	0	4		1	-4	10
doc18	0	0	0	1	0	0	0	0	1	0	2			1	3
doc19	0	1	1	0	1	0	1	0	0	1	5	-1		-4	2
doc20	0	0	1	1	0	0	1	0	1	0	4			-1	5
Частота атрибутів	4	6	10	11	6	6	7	6	7	5					
Вектор інтересів															
User1	3	-2	-1	0	0	2	-1	-1	1	0					
User2	-2	2	2	3	-1	-2	0	3	0	-1					

Рисунок 3.4 – Спрощений приклад надання рекомендацій на основі тегів

Отже, процес надання рекомендацій відбувається у наступний спосіб.

1) Створення матриці тегів (визначає наявність або відсутність зв’язку між елементом та певним фактором).

2) Створення матриці взаємодії (відображає характер взаємодії користувача з певним елементом).

3) Розрахунок вектору інтересів на основі матриці тегів та матриці взаємодії.

4) Розрахунок ступеню відповідності кожного документа інтересам користувача на основі матриці тегів та вектору інтересів. Результат нормалізації розрахунків з рисунка 3.4 та застосування алгоритму TF-IDF представлено на рисунку 3.5.

NORMAL	baseball	economics	politics	Europe	Asia	soccer	war	security	shopping	family
doc1	0.447	0.000	0.447	0.000	0.447	0.447	0.000	0.000	0.000	0.447
doc2	0.000	0.500	0.500	0.500	0.000	0.000	0.000	0.500	0.000	0.000
doc3	0.000	0.000	0.000	0.577	0.577	0.577	0.000	0.000	0.000	0.000
doc4	0.000	0.000	0.500	0.500	0.000	0.000	0.500	0.500	0.000	0.000
doc5	0.000	0.577	0.000	0.000	0.000	0.000	0.000	0.000	0.577	0.577
doc6	0.707	0.000	0.000	0.707	0.000	0.000	0.000	0.000	0.000	0.000
doc7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.707
doc8	0.000	0.000	0.500	0.500	0.000	0.000	0.500	0.000	0.000	0.500
doc9	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.000	0.707	0.000
doc10	0.000	0.577	0.000	0.000	0.577	0.000	0.577	0.000	0.000	0.000
doc11	0.000	0.000	0.577	0.000	0.577	0.000	0.000	0.000	0.577	0.000
doc12	0.577	0.000	0.000	0.000	0.000	0.577	0.577	0.000	0.000	0.000
doc13	0.000	0.000	0.500	0.500	0.500	0.000	0.000	0.500	0.000	0.000
doc14	0.000	0.500	0.500	0.500	0.000	0.000	0.000	0.000	0.500	0.000
doc15	0.000	0.000	0.000	0.000	0.000	0.500	0.500	0.500	0.000	0.000
doc16	0.577	0.000	0.000	0.000	0.000	0.577	0.000	0.000	0.577	0.000
doc17	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.500	0.000	0.000
doc18	0.000	0.000	0.000	0.707	0.000	0.000	0.000	0.000	0.707	0.000
doc19	0.000	0.447	0.447	0.000	0.447	0.000	0.447	0.000	0.000	0.447
doc20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
IDF	0.260	0.167	0.100	0.091	0.167	0.167	0.143	0.167	0.143	0.250

Profiles										
User 1	1.732	-0.847	-0.900	0.207	0.000	1.025	-0.447	-0.900	0.577	0.000
User 2	-1.025	1.000	1.053	1.500	-0.447	-1.025	-0.077	1.500	0.000	-0.447

User 1	User 2
1	-1
-1	1
	1
1	
	-1
1	
	1
-1	

Prediction 1	Prediction 2
1.009	-0.848
-0.870	2.526
0.711	0.016
-0.820	1.988
-0.214	0.319
1.371	0.336
-0.358	0.744
-0.370	1.014
1.133	-0.724
-0.806	0.274
0.048	0.350
1.133	-1.218
-0.396	1.053
-0.331	1.776
0.142	0.049
1.535	-1.183
-0.870	2.526
0.695	1.061
-0.847	0.493
-0.581	1.238

Prediction 1	Prediction 2
0.248	-0.217
-0.136	0.329
0.109	-0.068
-0.089	0.240
-0.044	0.045
0.319	-0.085
-0.059	0.114
-0.048	0.071
0.179	-0.121
-0.128	0.047
0.019	0.018
0.313	-0.253
-0.087	0.319
-0.053	0.104
0.021	0.102
0.336	-0.240
-0.136	0.329
0.072	0.086
-0.122	0.043
-0.098	0.115

Рисунок 3.5 – Приклад застосування TF-IDF

Отже, алгоритм TF-IDF дозволяє надати цінність найбільш специфічним тегам та врахувати їх вагу при формуванні рекомендацій. На рисунку 3.6 візуалізовано принцип факторизації матриці. Даний алгоритм можна застосувати, наприклад, при аналізі матриці користувацьких вподобань.

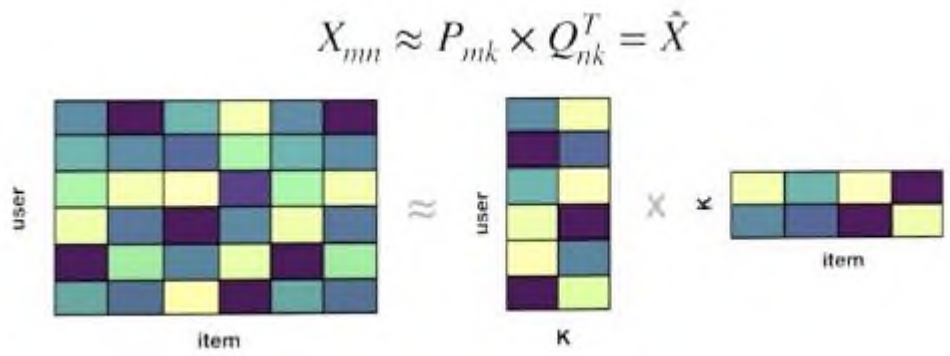

Рисунок 3.6 – Пояснення matrix factorization [20]

Отже, після дослідження способів реалізації рекомендаційних алгоритмів на основі фільтрації контенту та створення прикладу розрахунків можливо розробити покрокові алгоритми функціонування рекомендаційної системи онлайн-книгарні.

3.3. Розробка алгоритму функціонування рекомендаційної системи

Рекомендаційна система онлайн-книгарні реалізується у декілька кроків:

1) Збір даних про книги за допомогою сканування web-сторінок (рис. 3.7).

2) Трансформація, обробка та збереження даних про книги.

3) Розробка графічного інтерфейсу для збору даних про взаємодію користувачів з представленими книгами.

4) Надання рекомендацій на основі даних про книги та взаємодію користувачів з цими книгами.

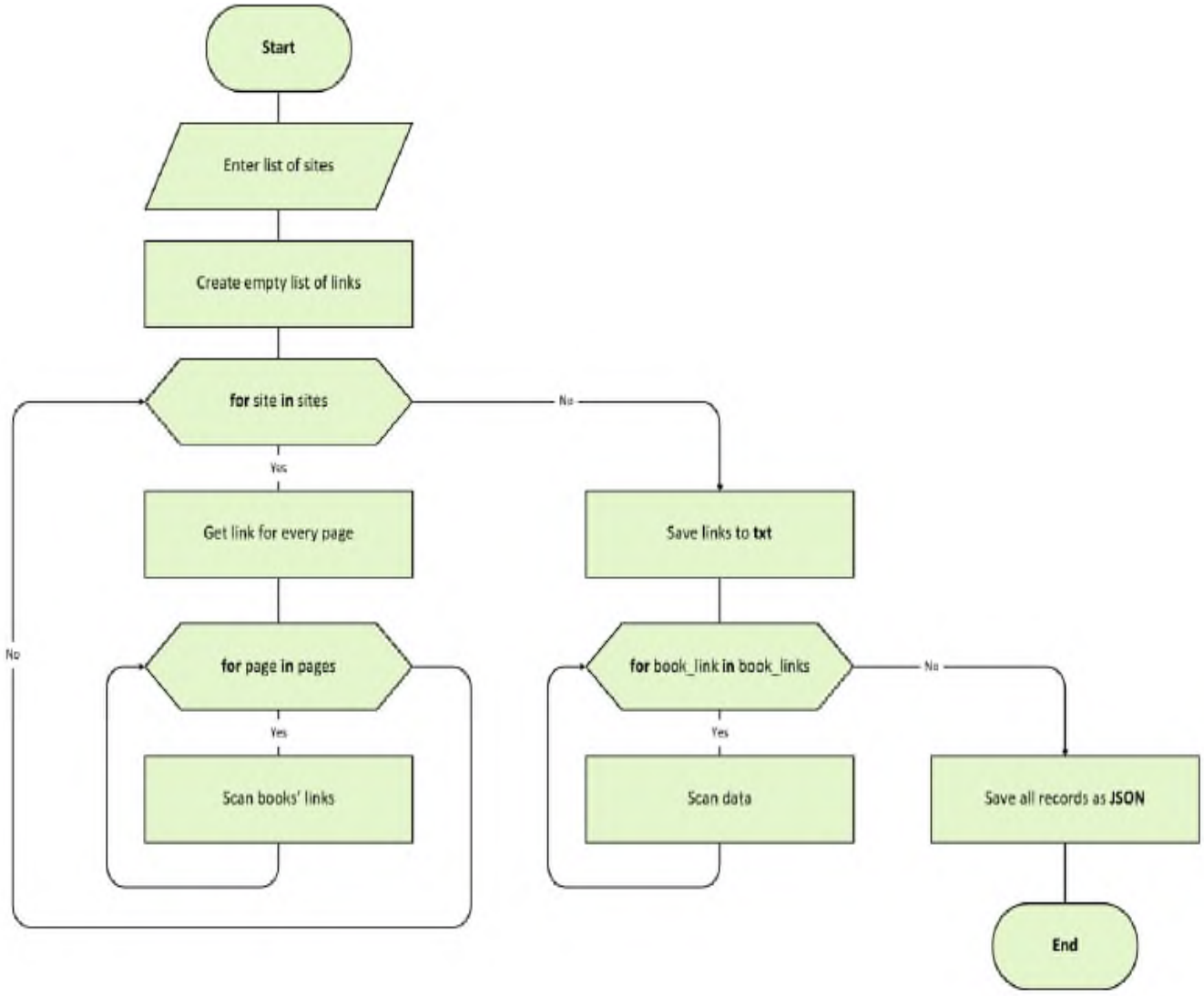

Рисунок 3.7 – Блок-схема алгоритму web scraping

Важливо розуміти що посилання на книги представлено на багатьох сторінках. Тому першим етапом є збір та збереження посилань. Після збору по силань на окремі книги можливо переходити до сканування відповідних сторінок. Не варто робити неперервний процес зі сканування web- сторінок, обробки даних та запису результатів в БД. Кожен з цих етапів вирізняється складністю та високою ймовірністю виникнення помилок. Такий підхід суттєво ускладнює відлагодження та знижує читабельність коду. На рисунку 3.8 представлена блок-схема алгоритму обробки, трансформації та збереження даних.

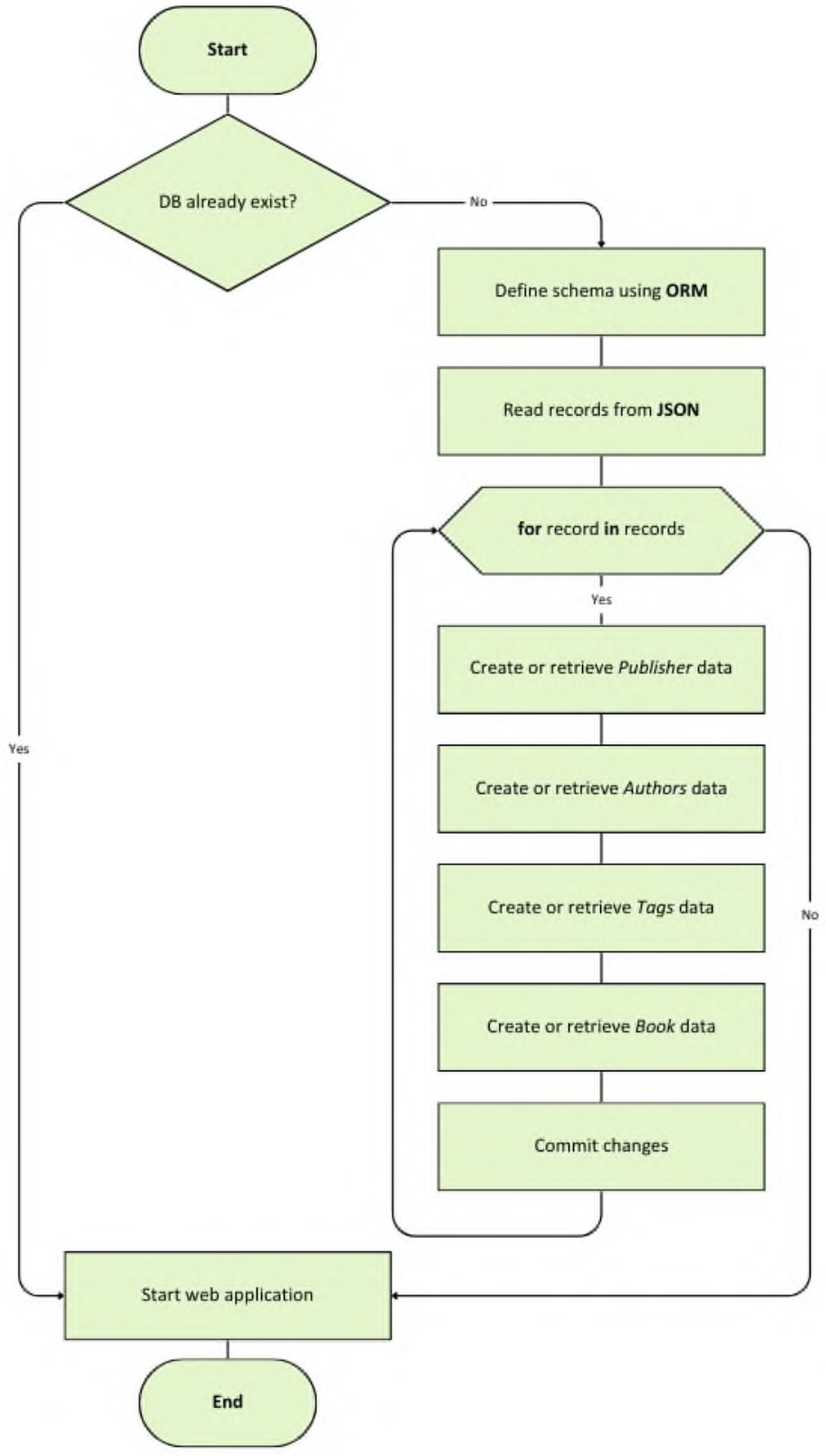

Рисунок 3.8 – Алгоритм обробки даних

Заповнення бази даних відбувається лише один раз при ініціалізації. Дані не зчитуються в режимі реального часу з сайтів книжкових магазинів, оскільки для

демонстрації принципу роботи рекомендаційної системи достатньо статичного датасету. Спочатку необроблені дані записуються у файл JSON, а вже наступним кроком записи з цього файлу читаються, обробляються та трансформуються відповідно до моделі даних. Варто враховувати що при існуванні зв'язків між сутностями та використанні зовнішніх ключів порядок додавання об'єктів (записів) в таблиці бази даних має значення.

Блок-схема алгоритму надання рекомендацій на основі історії взаємодії користувача з книгами представлена на рисунку 3.9. Константа N визначається в коді застосунку відповідно до вимог стосовно кількості рекомендованих книг.

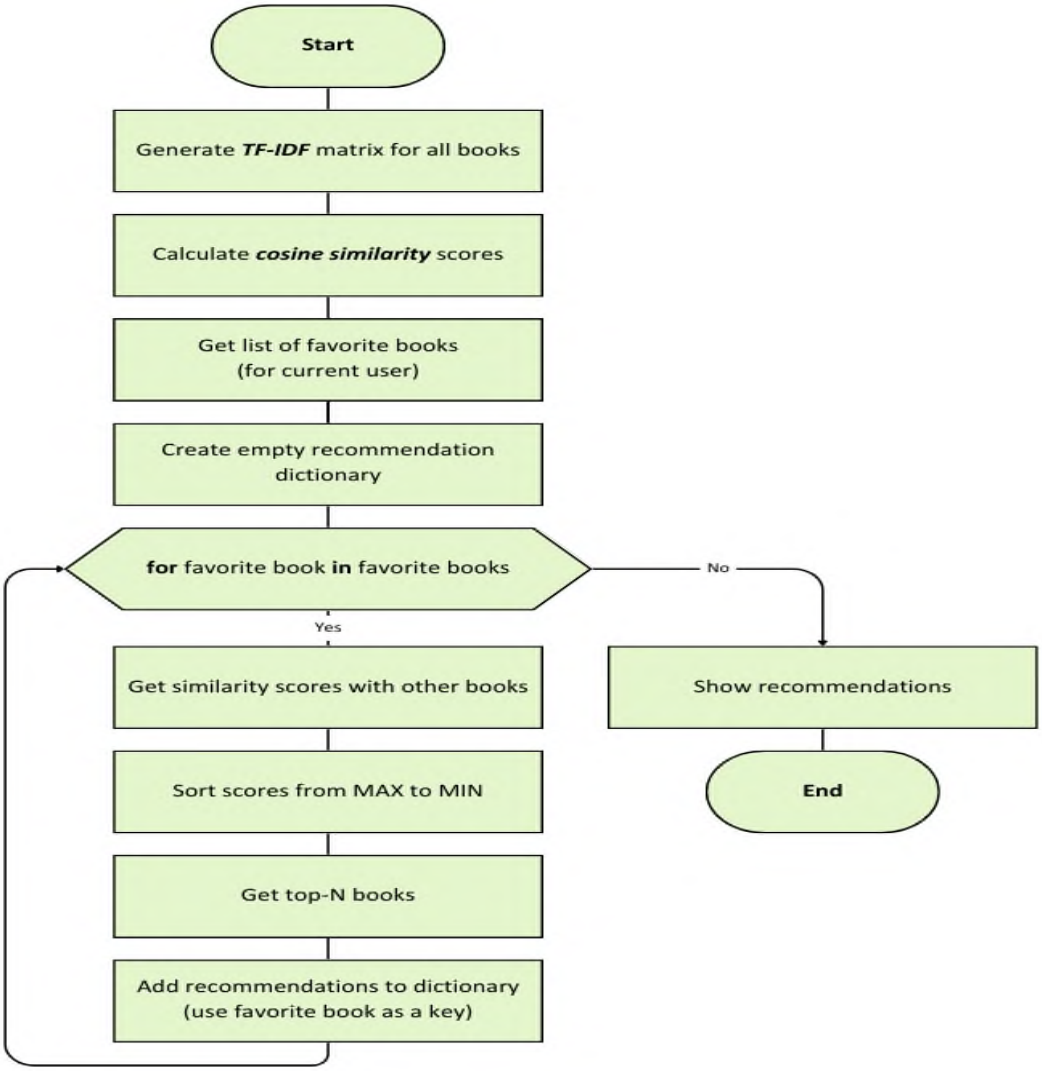

Рисунок 3.9 – Рекомендаційний алгоритм TF-IDF та cosine similarity застосовано у рекомендаційному алгоритмі.

3.4 Бінарні дерева

Бінарне дерево – це структура даних, кожен елемент якої окрім самих даних містить покажчики на два наступних елементи структури. Один з цих наступних елементів умовно називається лівим, а інший правим. Кожен елемент дерева називається вузлом або листом дерева. Перший вузол дерева (з якого дерево власне починається) називається коренем. Фрагмент дерева разом з вузлом, від якого він починається, називається піддеревом або віткою. Множина всіх вузлів, рівновіддалених від кореня, називається рівнем. Вузол, з якого не починається жодна вітка, називається кінцевим або термінальним вузлом. Оскільки дерево бінарне, кожен вузол може породжувати два вузли наступного рівня. Породжені вузли є дочірніми по відношенню до вузла, що їх породив. Породжуючий вузол є батьківським по відношенню до своїх дочірніх вузлів. Батьківський вузол разом із своїми дочірніми складає ланку. Сумарна кількість рівнів дерева називається висотою дерева.

Основними операціями при роботі з деревами є:

- додавання елемента до дерева;
- пошук елемента дерева, що відповідає заданому критерію пошуку;
- сортування елементів дерева;
- вилучення елемента дерева.

Процес доступу до елементів дерева називається проходженням дерева. Існує три способи проходження дерев: послідовний, низхідний та висхідний.

При послідовному методі доступу до елементів дерева порядок проходження дерева наступний: спочатку розглядається самий лівий відносно кореня елемент, потім його батьківський вузол, потім правий елемент даної ланки, потім переходять до елемента попереднього рівня, що є батьківським по відношенню до батьківського вузла даної ланки і т. д. Вверх до кореня, а потім від кореня вниз до самого правого елемента.

При низхідному проходженні дерева порядок обходу елементів зверху- вниз та зліва-направо. Тобто спочатку проходиться корінь, потім його лівий елемент, а потім правий і т.д.

При висхідному проходженні порядок проходження зліва-направо та знизу-вверх. Тобто спочатку проходиться лівий вузол найнижчої ланки, потім правий вузол цієї ланки, а потім їх батьківський вузол і т.д.

Як бачимо бінарне дерево – досить складна структура даних. Фактично це ускладнений варіант списку. Деякі операції реалізуються значно складніше ніж у списку. Однак операції пошуку у відсортованому бінарному дереві виконуються значно ефективніше ніж у списку, тому бінарні дерева використовуються на практиці досить часто.

3.5. Бінарні дерева пошуку

Бінарне дерево пошуку — це бінарне дерево, яке володіє додатковими властивостями: значення лівого нащадка менше значення батька, а значення правого нащадка більше значення батька для кожного вузла дерева. Тобто, дані в бінарному дереві пошуку зберігаються у відсортованому вигляді. При кожній операції вставки нового або видалення існуючого вузла відсортований порядок дерева зберігається. При пошуку елемента порівнюється шукане значення з коренем. Якщо шукане більше кореня, то пошук продовжується в правому нащадка кореня, якщо менше, то в лівому, якщо одно, то значення знайдено і пошук припиняється.

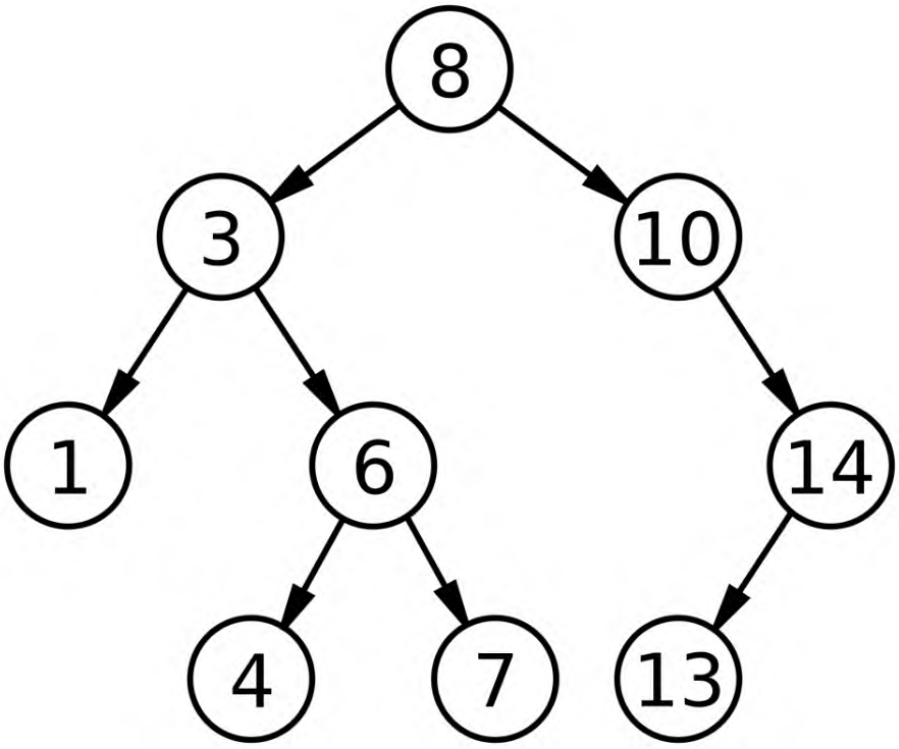

Рисунок 3.10 – Бінарне дерево пошуку

Збалансоване бінарне дерево пошуку — це бінарне дерево пошуку з логарифмічною заввишки. Дане визначення швидше ідейне, ніж суворе. Суворе визначення оперує різницею глибини самого глибокого і самого неглибокого аркуша (AVL-дерева) чи відношенням глибини самого глибокого і самого неглибокого листа (у червоно-чорних деревах). У збалансованому бінарному дереві пошуку операції пошуку, вставки і видалення виконуються за логарифмічний час (так як шлях до будь-якого листа від кореня не більше логарифма). В виродженому випадку незбалансованого бінарного дерева пошуку, наприклад, коли в порожнє дерево вставлялася відсортована послідовність, дерево перетвориться в лінійний список, і операції пошуку, вставки і видалення будуть виконуватися за лінійний час. Тому балансування дерева вкрай важлива. Технічно балансування здійснюється поворотами частин дерева при вставці нового елемента, якщо вставка елемента порушила умова збалансованості.

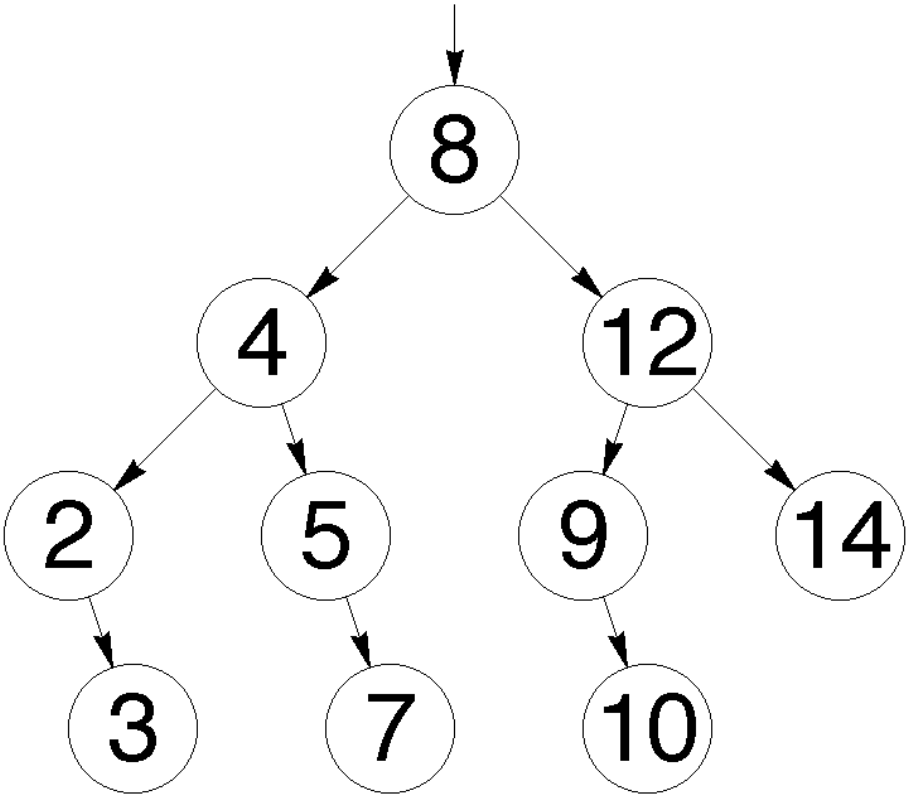

Рисунок 3.11 – Збалансоване бінарне дерево пошуку

Збалансоване бінарне дерево пошуку застосовується, коли необхідно здійснювати швидкий пошук елементів, що чергується зі вставками нових елементів і вилученнями існуючих. У разі, якщо набір елементів, що зберігається в структурі даних фіксований і немає нових вставок і вилучень, то масив краще. Тому що пошук можна здійснювати алгоритм бінарного пошуку за логарифмічний час, але відсутні додаткові витрати по зберіганню і використанню покажчиків.

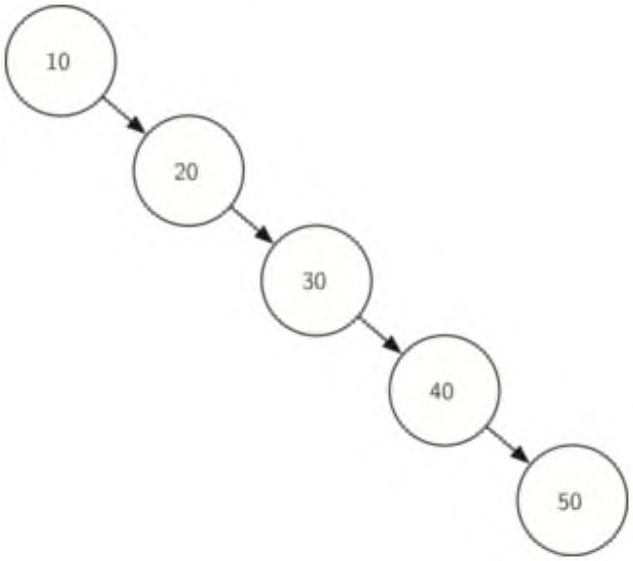

Рисунок 3.12 – Екстремально збалансоване бінарне дерево пошуку

Отже, бінарне дерево пошуку – це структура даних, яка призначена для того, щоб реалізувати швидкий пошук елементів, які зберігаються в структурі.

ВИСНОВКИ ДО РОЗДІЛУ

В третьому розділі розроблено блок-схеми алгоритмів збору даних, обробки даних та надання рекомендацій. Визначено математичні моделі для відповідних алгоритмів та наведено метрики оцінки точності наданих рекомендацій.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Обрання технологій для реалізації рекомендаційної системи

Рекомендаційна система реалізована у форматі вебзастосунка. В якості мови програмування для створення серверної частини обрано Python (версія 3.12.0). Дана високорівнева та платформно-незалежна мова програмування загального призначення дозволяє просто та ефективно впровадити алгоритми машинного навчання [16]. В даній роботі необхідно поєднати рекомендаційний алгоритм, web scraping, базу даних та вебзастосунок. Саме Python є універсальним та багатофункіональним рішенням, яке дозволяє об'єднати різноманітні функції в межах одного проєкту. В якості фреймворку обрано Flask. Фреймворк – це набір готових програмних рішень, які визначають архітектуру системи. У таблиці 4.1 наведено перелік використаних бібліотек.

Таблиця 4.1 – Використані бібліотеки (Python)

Назва	Призначення	Версія
urllib	Пакет, який використовується для отримання доступу до вебсторінок з коду мовою Python.	3.12.0
BeautifulSoup	Бібліотека для парсингу даних. В даному проєкті дозволяє аналізувати HTML.	4.12.2
SQLAlchemy	ORM для взаємодії з базою даних.	2.0.25
Flask	Фреймворк для розробки вебзастосунку.	3.0.1
scikit-learn	Бібліотека, яка містить реалізацію алгоритмів машинного навчання і ще деякі додаткові функції.	1.4.0

Клієнтська частина застосунка реалізована з використанням HTML та CSS. Макети сторінок оптимізовано та пов'язано з серверним кодом за допомогою шаблонізатора Jinja2. Для вдосконалення стилів використано Bootstrap. Шрифти урізноманітнено за допомогою Google Fonts.

В якості системи керування базами даних обрано SQLite. Взаємодія з таблицями у базі даних відбувається з використанням ORM (Object-Relational Mapping). ORM – це об'єктно-реляційне відображення, яке дозволяє пов'язати

об’єкти в коді з таблицями в базі даних. Отже, система може бути адаптована до будь-якої іншої СКБД в разі виникнення такої потреби.

4.2. Опис загальної структури проекту

Застосунок складається з багатьох елементів, кожен з яких відповідає за деякий аспект функціоналу. На рисунку 4.1 представлено принцип організації файлів в межах проєкту.

Рисунок 4.1 – Структура застосунку

Директорії та файли мають наступне призначення:

1) Директорія «web_scraping» містить код для сканування web-сторінок та запису зібраних даних у JSON-файли. У файлі web_scraper збережено базовий клас WebScraper. У routes додано URL-адреси та деякі індивідуальні налаштування для різних онлайн-книгарень. Файли komora_web_scraper та nash_format_web_scraper містять класи, які походять від WebScraper та дозволяють просканувати сайти онлайн-книгарень «Комора» та «Наш Формат». Останній є ретейлером, тому з його сайту можна отримати книги багатьох різних видавництв. В main відбувається запуск процесу збору даних.

2) У директорії «clean_json_data» розміщено код для очищення даних, які зібрано в результаті сканування web-сторінок. Необроблені дані знаходяться у спеціальній директорії «raw_json_data». Попередній аналіз показав що різні книгарні надають різну кількість даних про окремі книги та різними способами підписують однакові атрибути (наприклад, «ISBN», «ISBN:» та навіть «Виробник» позначають той самий атрибут). Саме тому записи про книги потрібно узгодити, очистити та підготувати до наступних етапів обробки.

3) Шаблони різних сторінок застосунку розміщено в «templates». Базовим шаблоном є «base.html», а всі інші наслідують частину макету цієї сторінки.

4) У «static» містяться користувацькі CSS-стилі та деякі зображення (графіка для сторінки авторизації та для пустої сторінки з рекомендаціями на випадок відсутності інформації про вподобання користувача).

5) В «instance» знаходиться БД SQLite.

6) Файл «stop_words_ua.txt» містить слова, які часто трапляються у текстах, але не представляють жодної цінності для розуміння унікальності тексту чи його теми. Для англійської мови цей список вбудовано у алгоритм за замовчуванням, але для української мови довелося його завантажити вручну [21].

7) В app міститься код моделей для ORM, форми реєстрації та авторизації, маршрутизація та функція для перенесення даних з файлу JSON до бази даних. Можна сказати що app є основою даного Flask-застосунка.

Отже, при розгортанні системи файли виконуються у наступному порядку: main з «web_scraping», data_cleaning з «clean_json_data» та нарешті app. Якщо база даних вже існує, то застосунок працює як рекомендаційна система без повторного сканування вебсторінок та оновлення БД.

4.3. Реалізація алгоритму сканування вебсторінок

Сканування web-сторінок відбувається за допомогою бібліотеки BeautifulSoup. Не залежно від адреси сайту існує перелік стандартних кроків для парсингу HTML-сторінки, які об'єднано в межах одного базового класу. Отже, сканування сторінки зазвичай починається з наступного коду:

```
# PARSING
def __parse_web_data(self, url):
    ctx = ssl.create_default_context()
    ctx.check_hostname = False
    ctx.verify_mode = ssl.CERT_NONE

    html = urlopen(url, context=ctx).read()

    return BeautifulSoup(html, «html.parser")
```

В даному лістингу відбувається підключення до сайту за допомогою urllib з наступним перетворення вмісту на об'єкт BeautifulSoup. Вказаний об'єкт має набір методів та властивостей для вилучення необхідних фрагментів інформації з усього масиву різноманітних HTML-тегів та атрибутів. Повний код базового класу для всіх «сканерів» наведений у додатку А. Приклад отримання вмісту тегу наведено нижче.

```
soup.select("#annotation")[0].text.strip()
```

Наприклад, для отримання опису книги необхідно взяти перший елемент з ідентифікатором «annotation» у наданій HTML-сторінці та отримати його текст. Зручно відразу здійснювати обробку рядків та зберігати лише необхідну інформацію. Також інколи доводиться використовувати регулярні вирази, наприклад, для розбиття рядка з тегами на окремі теги.

Перш за все необхідно зібрати посилання на всі доступні сторінки з книгами. На цих сторінках потрібно просканувати посилання на окремі книги. Для цього завдання розроблено функції відповідно до структури вебсторінок певної онлайн-книгарні. Приклад такої функції та алгоритму сканування даних про книгу наведено у додатку Б.

Отже, в коді застосунку відбувається послідовна обробка сторінок про книги з усіх сайтів, адреси яких збережено у routes. Фрагмент коду для запуску процесу сканування даних:

```
    web_scrapers = []

    # Komora
    routes = ROUTES["komora"]
    komora_ws = KomoraWebScraper(routes["initial"], routes["book"],
routes["file"])
    web_scrapers.append(komora_ws)

    # Publishers: Laboratoria, Old Lion Publishing House, Nash Format,
Fabula, Vivat
    publishers = ["laboratoria", "olph", "nf", "fabula", "vivat"]
    for publisher in publishers:
        routes = ROUTES[publisher]
        ws = NashFormatWebScraper(routes["initial"], routes["book"],
routes["file"], routes["base"])
        web_scrapers.append(ws)

    # PARSING
    total = 0
    for web_scraper in web_scrapers:
        web_scraper.parse()
        total = total + web_scraper.books_count
```

Результати сканування даних з заданих сайтів представлено на рисунку 4.2. Отже, код для web scraping успішно виконано. Отримані дані збережено до файлів JSON з метою подальшої обробки та трансформації.

```
Збір даних з сайту: https://komorabooks.com/shop/

Пошук посилань на книги на сторінці 10 з 10 (https://komorabooks.com/shop/?product-page=10)
Просканована 100% сторінок про окремі книги
JSON-файл успішно збережено. Кількість книг: 121

Збір даних з сайту: https://laboratoria.pro/catalog/books

Пошук посилань на книги на сторінці 48 з 48 (https://laboratoria.pro/catalog/books/page-48)
Просканована 100% сторінок про окремі книги
JSON-файл успішно збережено. Кількість книг: 952

Збір даних з сайту: https://starylev.com.ua/bookstore

Пошук посилань на книги на сторінці 57 з 57 (https://starylev.com.ua/bookstore/page--57)
Просканована 100% сторінок про окремі книги
JSON-файл успішно збережено. Кількість книг: 1635

Збір даних з сайту: https://nashformat.ua/publishers/nash-format-books

Пошук посилань на книги на сторінці 34 з 34 (https://nashformat.ua/publishers/nash-format-books/page-33)
Просканована 100% сторінок про окремі книги
JSON-файл успішно збережено. Кількість книг: 670

Збір даних з сайту: https://nashformat.ua/publishers/fabula-books

Пошук посилань на книги на сторінці 19 з 19 (https://nashformat.ua/publishers/fabula-books/page-18)
Просканована 100% сторінок про окремі книги
JSON-файл успішно збережено. Кількість книг: 371

Збір даних з сайту: https://nashformat.ua/publishers/vivat-books

Пошук посилань на книги на сторінці 70 з 70 (https://nashformat.ua/publishers/vivat-books/page-27)
Просканована 100% сторінок про окремі книги
JSON-файл успішно збережено. Кількість книг: 1395
Отже, загальна кількість книг: 5142
```

Рисунок 4.2 – Результати сканування вебсторінок

Наступним кроком варто розробити базу даних.

4.4. Розробка бази даних

Повний текст програми для створення схеми бази даних та її заповнення зібраними даними надано у додатку В. Після заповнення бази даних можна робити запити за допомогою SQL. Завдяки запитам з'ясовано що всього 360 книг з 3922 мають теги, визначені менеджерами інтернет-магазину. Саме тому важко використати лише ці теги для надання рекомендацій. Отже, варто формувати рекомендації на основі більш інформативного опису книги. На рисунку 4.3 представлено зразок даних, які отримано за допомогою SQL-запиту.

```
1  SELECT book_title, author_name, language, year, publisher_name, pages, price
2  FROM book
3  JOIN publisher ON publisher.id = book.id
4  JOIN authors ON authors.book_id = book.id
5  JOIN author ON author.id = authors.author_id
```

	book_title	author_name	language	year	publisher_name	pages	price
1	Книга «Наше Спільне. Як зберегти в собі ...	Таня Касьян	українська	2023	Лабораторія	216	269.0
2	Книга «Як насправді влаштований світ. Минуле	Вацлав Сміл	українська	2022	Віват	216	329.0
3	Книга «Нові стоїки. 52 уроки для ...	Ґреґорі Лопіз	українська	2021	Комора	216	449.0
4	Книга «Нові стоїки. 52 уроки для ...	Максімос Пільончі	українська	2021	Комора	216	449.0
5	Книга «Вища вірність. Правда, брехня і ...	Джеймс Комі	українська	2022	Фабула	288	399.0
6	Книга «Напокірні»	Емілія Гарт	українська	2023	Видавництво Старого Лева	320	399.0
7	Книга «Чого я не навчився у школі. ...	Ерлінг Кагге	українська	2023	Наш Формат	208	299.0
8	Книга «Канцелярка. Дивовижна одіссея Антоніни	Кеті Мартон	українська	2021	Наш Формат , в книгогав	288	299.0
9	Книга «Фізична (не)активність. Що насправді ...	Даніел Ліберман	українська	2021	Наш Формат , Дитяча редакція видавництва	432	349.0
10	Книга «Розгадка геніальності. Як працює ...	Рон Фрідман	українська	2021	Наш Формат , Центр Досліджень Визволення	256	299.0

Рисунок 4.3 – Результати запиту до бази даних

У дані про книги зміни не вносяться. Моделі «User» та «Action» змінюються в ході роботи застосунку, оскільки реєструються нові користувачі та ставляться нові оцінки «Подобається». Нижче наведено код для збереження оцінки.

```
@app.route('/like-<int:book_id>')
@login_required
def like(book_id):
    like = Action.query.filter(Action.book_id == book_id, Action.user_id ==
current_user.id).first()
    if not like:
        like = Action(book_id = book_id, user_id = current_user.id, type =
"like")

    db.session.add(like)
    db.session.commit()

    return redirect(url_for("book", book_id=book_id))
```

Якщо користувач хоче прибрати вподобання, то запис про лайк просто видаляється з бази даних. Система придатна до розширення функціональності за допомогою реакцій інших типів. Наприклад, можна додати можливість ставити оцінку книгам або негативну реакцію. Код для видалення лайку наведено нижче:

```
@app.route('/dislike-<int:book_id>')
@login_required
def dislike(book_id):
    like = Action.query.filter(Action.book_id == book_id, Action.user_id ==
current_user.id).first()
    if like:
        db.session.delete(like)
    db.session.commit()

    return redirect(url_for("book", book_id=book_id))
```

На рисунку 4.4 продемонстровано зміну інтерфейсу при натисканні кнопки «Подобається».

Рисунок 4.4 – Кнопки для надання реакції на книги

У наведених функцій є декоратори, які вказують що неавторизований користувач не може надавати реакції на книги та створювати вектор інтересів. Також додано декоратор для маршрутизації. Декоратори дозволяють модифікувати поведінку функції чи класу без зміни їх коду [17].

4.5. Реалізація рекомендаційного алгоритму

Рекомендаційний алгоритм реалізовано на основі класу TfidfVectorizer та методу cosine_similarity з scikit-learn [15]. Scikit-learn – це бібліотека для Python, яка використовується для створення та тренування різноманітних моделей машинного навчання на основі класифікації, регресії та кластерізації [14]. У розробленій функції для пошуку подібних книг додано сортування коефіцієнту подібності (score) від максимального до мінімального значення.

```
def similar_books(book_id, n):
    global COS_SIM

    # Similarity score for all books
    if COS_SIM is None:
        stop_words_ua = read_stop_words("stopwords_ua.txt")
        descriptions = [book.description for book in Book.query.all()]

        vectorizer = TfidfVectorizer(stop_words=stop_words_ua)
        books_tfidf_matrix = vectorizer.fit_transform(descriptions)

        COS_SIM = cosine_similarity(books_tfidf_matrix, books_tfidf_matrix)

    # Similarity score for selected book and all other books
    ids = [book.id for book in Book.query.all()]
    scores = sorted(zip(ids, COS_SIM[book_id - 1]), key=lambda x: x[1],
reverse=True)[1:n + 1]

    # Best match book's ids
    similar_books_ids = [score[0] for score in scores]

    return Book.query.filter(Book.id.in_(similar_books_ids)).all()
```

Очевидно що книга має найбільший коефіцієнт подібності (рівний 1) з самою собою. Тому в отриманому рейтинговому переліку книг варто виключити перший елемент для уникнення дублювання. Також важливо обмежити кількість рекомендованих варіантів за допомогою параметра n. Надто велика кількість пропозицій обтяжує користувача і призводить до того що до рекомендацій можуть потрапити книги з мінімальним коефіцієнтом подібності (наближається до 0). Система не має достатньої кількості даних про поведінку різних користувачів, тому створювати тестовий та навчальний набір даних для оцінки точності рекомендацій неможливо. Саме тому проведено тестування вручну, результати якого свідчать про достатню точність надання рекомендацій.

4.6. Програмна реалізація

У розробленому застосунку заборонено доступ неавторизованих користувачів. Тому сторінки авторизації та реєстрації є найпершим елементом системи, з яким користувач повинен взаємодіяти для отримання доступу до всіх наявних функцій. На рисунку 4.5 представлено форму реєстрації.

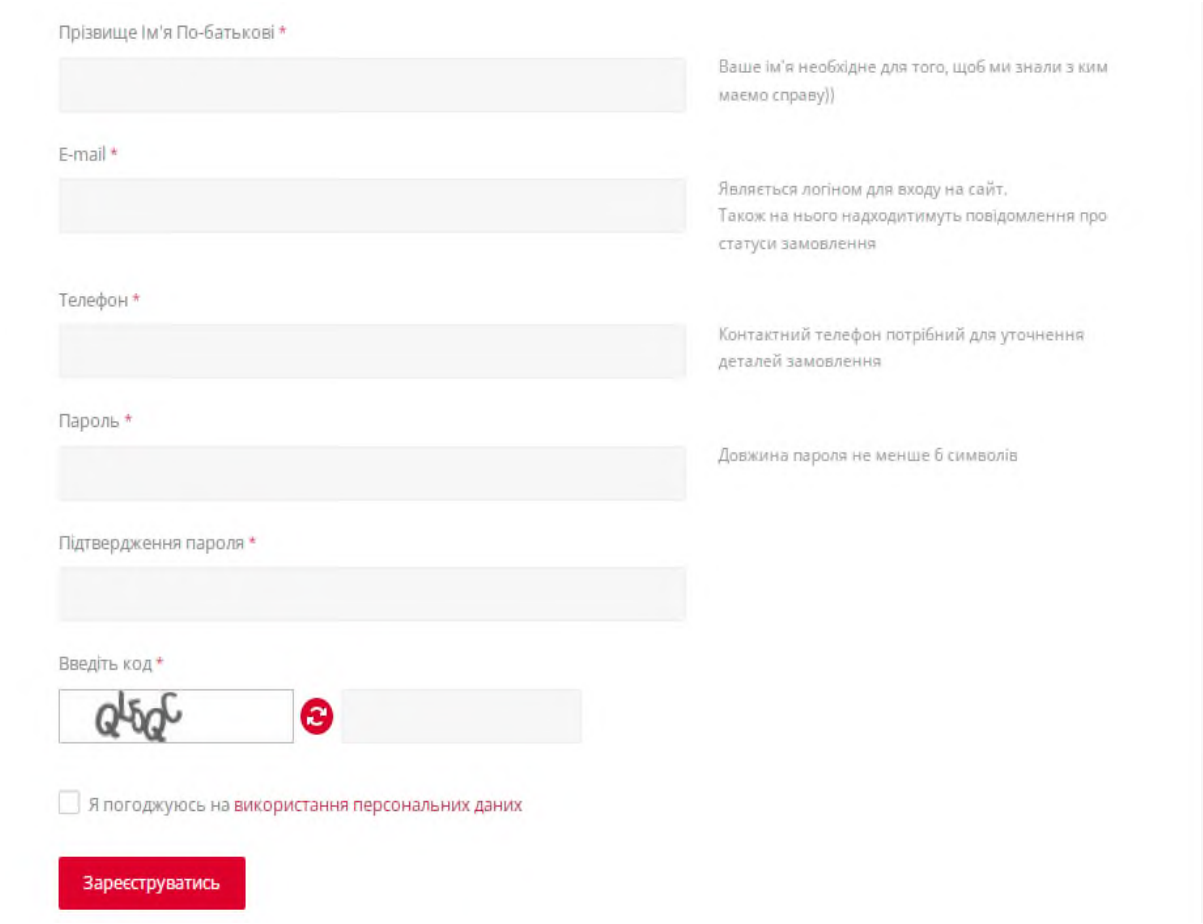

Рисунок 4.5 – Форма реєстрації

Форма авторизації відображається щоразу при спробі неавторизованого користувача отримати доступ до сторінок сайту за посиланням (рис. 4.6).

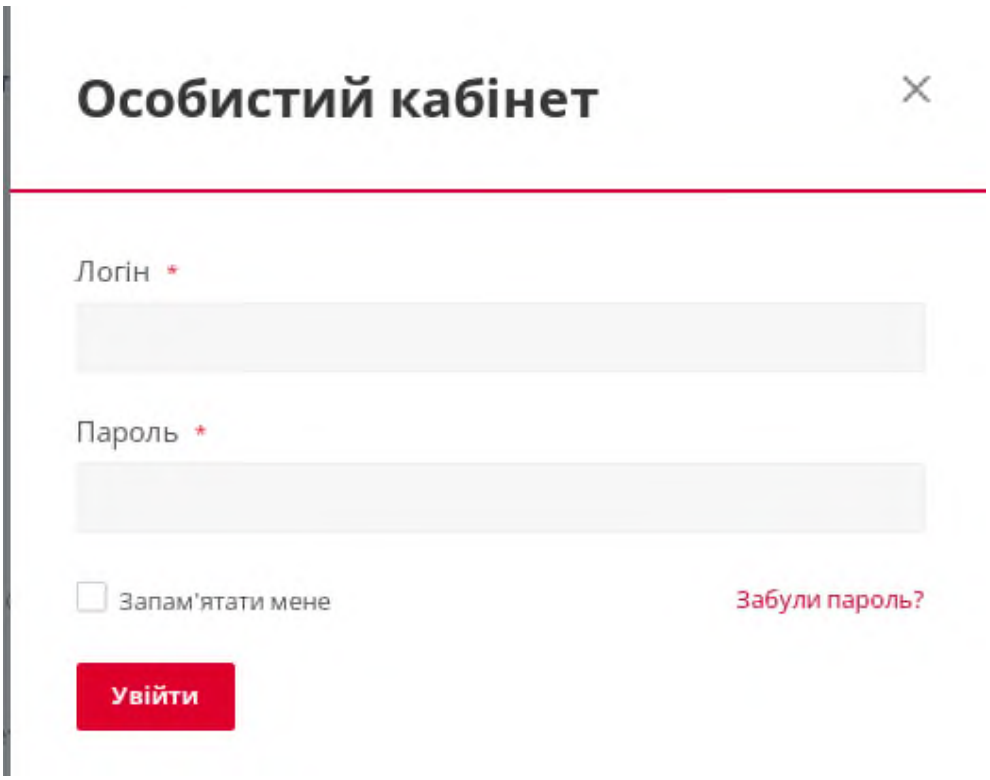

Рисунок 4.6 – Форма авторизації

Після авторизації та реєстрації можливо перейти до головної сторінки сайту, на якій наведено перелік книг, форму пошуку книги за назвою та кнопки для отримання рекомендацій чи для виходу з системи.

На рисунку 4.7 зображено скріншот головної сторінки застосунка. У системі реалізовано розбиття на окремі сторінки (pagination) через велику кількість книг, які необхідно відобразити.

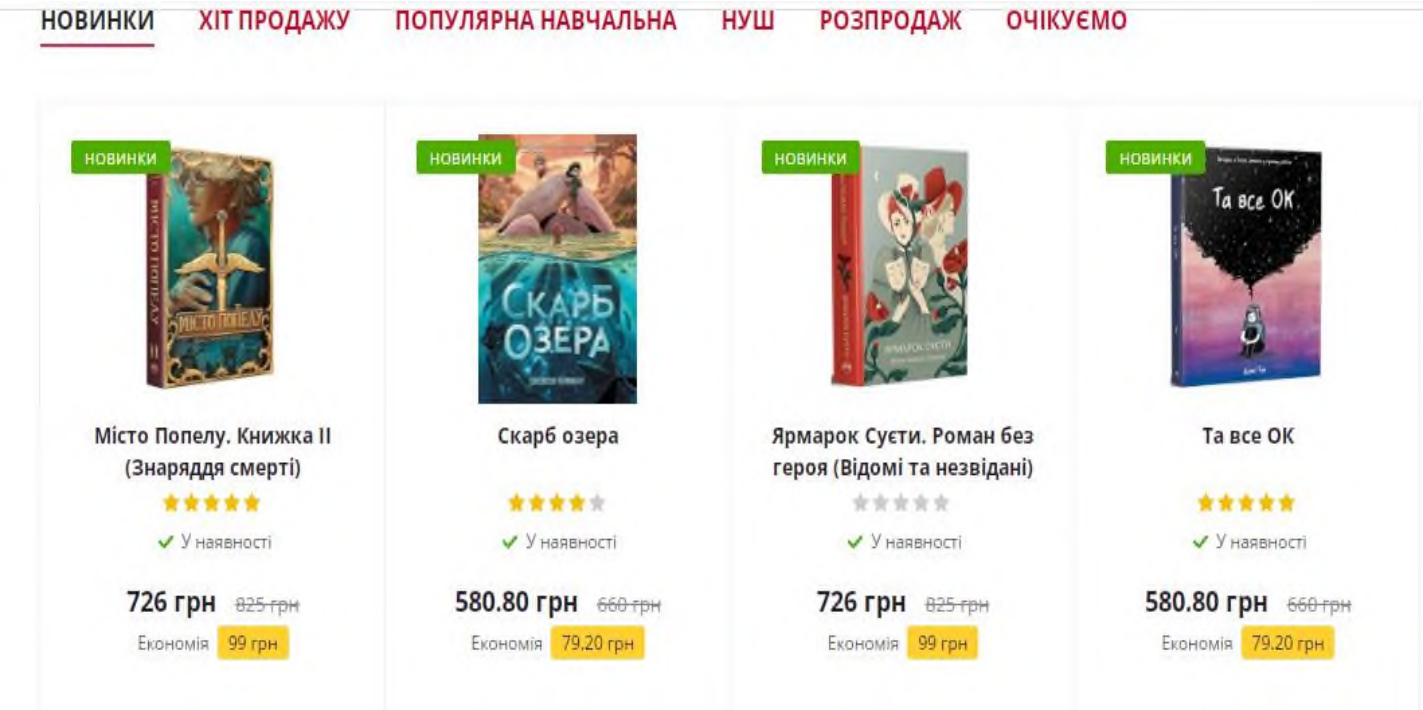

Рисунок 4.7 – Головна сторінка застосунка

Взагалі доступно більше 300 сторінок, на кожній з яких представлено по 12 книг. Передивлятися всю цю кількість заради пошуку певного твору не надто доцільно. Саме тому за допомогою форми пошуку за назвою можна швидко знайти книги, назва яких містить певне слово або фразу.

Пошук регістронезалежний та надає результати навіть у випадку часткового співпадіння. Результати пошуку за словом «інтелект» представлено на рисунку 4.8.

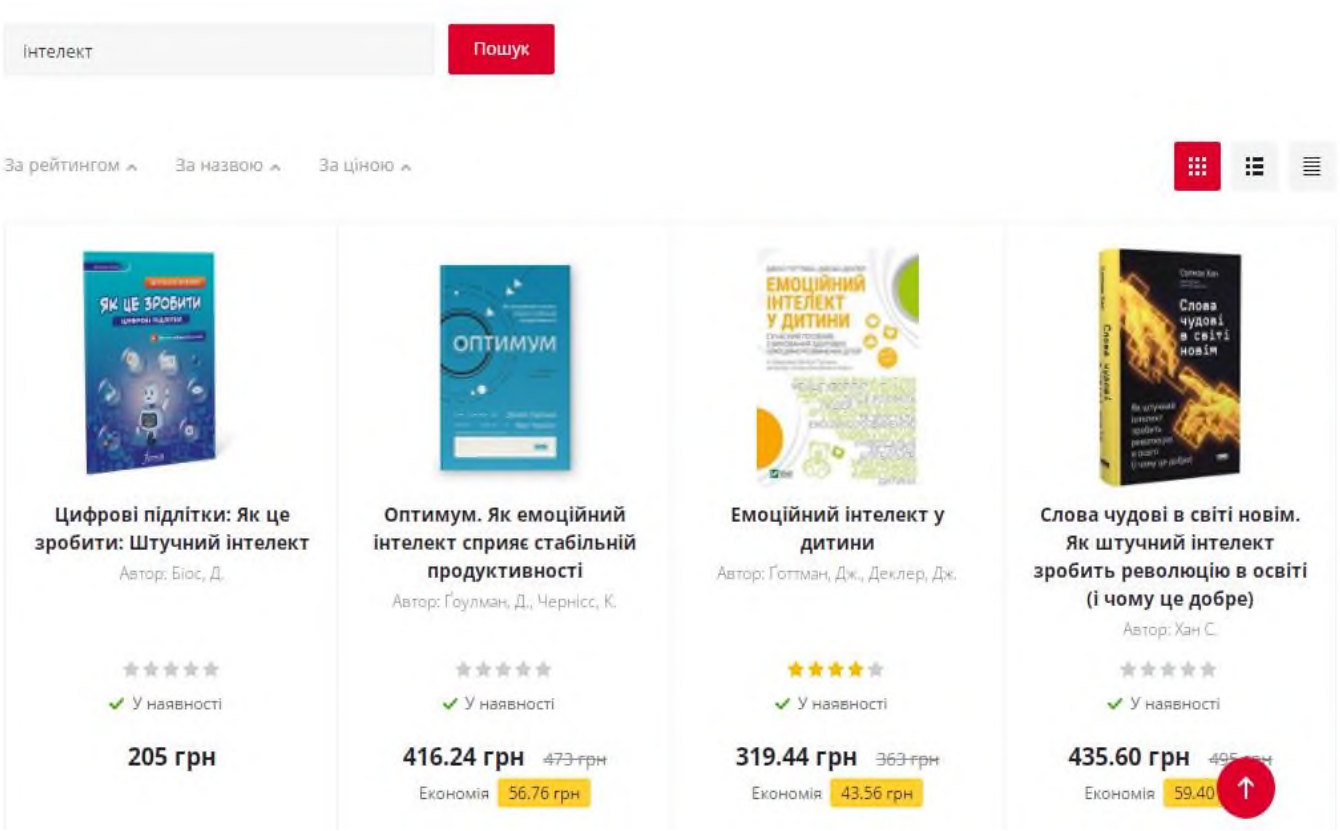

Рисунок 4.8 – Результати пошуку за назвою

Отже, користувач знаходить необхідну книгу та може перейти на сторінку з детальною інформацією про обраний твір. Вказана сторінка містить кнопку «Подобається». За допомогою цієї реакції користувачі формують свій вектор інтересів. Пісня того як книга вподобана з’являється кнопка «Не подобається», яка дозволяє відкоригувати вподобання за потреби.

Рисунок 4.9 є скріншотом сторінки про окрему книгу.

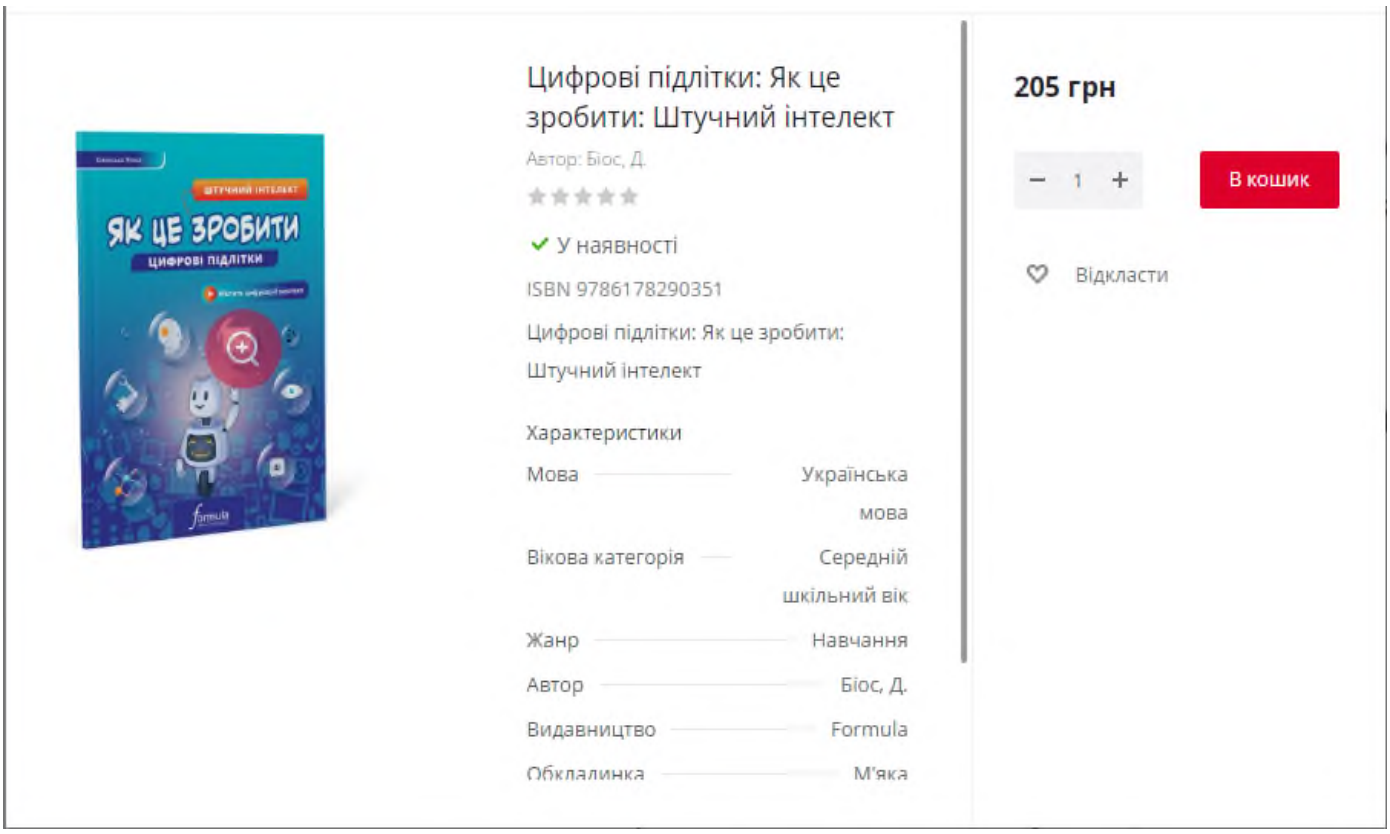

Рисунок 4.9 – Сторінка про окрему книгу

Система надає рекомендації на основі переліку вподобаних книг (рис. 4.10).

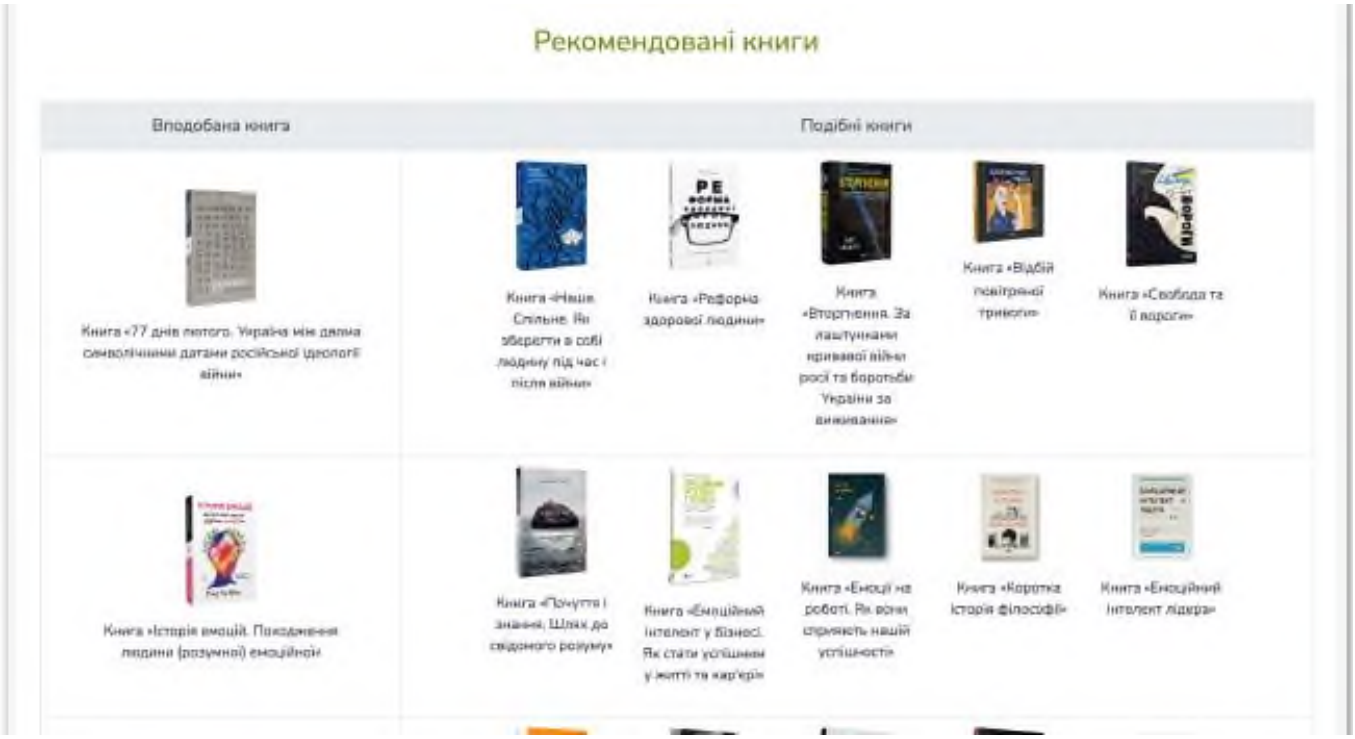

Рисунок 4.10 – Приклад надання рекомендацій

Якщо жодних вподобань немає, то система пропонує повернутися на головну сторінку та продовжити перегляд всього доступного асортименту. Отже, графічний інтерфейс рекомендаційної системи надає користувачу всі необхідні інструменти для отримання релевантних рекомендацій.

ВИСНОВКИ ДО РОЗДІЛУ

В ході розробки четвертого розділу здійснено реалізацію та тестування рекомендаційної системи онлайн-книгарні на основі засобів машинного навчання. Всі високорівневі рішення з попередніх розділів успішно імплементовано у розробленому Flask – застосунку.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП – ПРОЄКТУ

5.1. Опис ідеї проекту

Таблиця 5.1. Карта проєкту

Назва номінації	Рекомендаційна система
Назва проєкту	РС онлайн – книгарні
Назва ВНЗ,спеціальності	НЛТУ, кафедра комп'ютерних наук, "Комп'ютерні науки"
Прізвище,ім'я, по-батькові автора	Годованець Адам Данилович
Цілі і задачі проєкту	Розробити рекомендаційну систему онлайн – книгарні на основі засобів машинного навчання
Короткий зміст проєкту	Реалізовано рекомендаційну систему онлайн – книгарні на основі засобів машинного навчання. Для демонстрації створеного алгоритму створено вебзастосунок. Набір даних зібрано за допомогою web scraping та збережено як базу даних SQLite. Рекомендаційний алгоритм фільтрує контент та забезпечує користувача списком книг, який відповідає інтересам та зацікавленням (які визначено на основі попередньої активності на сайті).
Цільова аудиторія	Власники бізнесу та підприємці, які працюють в сфері діяльності, де робота виконується в командах, є чіткі завдання та цілі.
Терміни виконання проєкту	3 місяці
Бюджет проєкту	16000 грн.

Суть проекту – підвищення коефіцієнта конверсії онлайн – книгарні за рахунок надання списку книг, який відповідає інтересам користувача.

Метою даного проєкту є отримання такого програмного продукту.

5.2. Розроблення ринкової стратегії проєкту

Створення стратегії для ринку включає в себе визначення плану щодо охоплення цього ринку (табл. 5.2).

Таблиця 5.2.Опис цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції на ринку	Простота виходу на ринок
1.	Власники малих підприємств	Висока	Високий	На ринку спостерігається постійна конкуренція, однак введення унікальних рішень стане ключовим фактором для утвердження на ринку та забезпечення конкурентних переваг.	Розробка рекомендаційної системи не вимагає значних початкових інвестицій для запуску базової версії продукту, при цьому існує високий потенційний попит.
2.	Власники бізнесу	Середня	Середній		
3.	Розробники інформаційних проєктів	Середня	Середній		
4.	Керівники проєктів з найманими працівниками	Середня	Високий		

5.3. Розробка маркетингової програми стартап – проєкту

Таблиця 5.3. Визначення ключових переваг концепції потенційного продукту.

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Зручний та легкий в користуванні інтерфейс	Система підтримує логічну розділеність і повторювану структуру,що дозволяє користувачу зберігати один алгоритм для досягнення різних цілей.	Можливість використання системи користувачам без досвіду роботи з подібними інструментами.
2	Продуктивність програмного продукту	Структуроване зберігання даних дозволяє програмі працювати ефективно,оскільки інформація,яка рідко змінюється,може бути кешована для прискорення доступу.	Масштаб невиконання робіт

Використання користувачами спільних ресурсів, які збережені без повторень дозволяє зменшити витрати хмарної пам’яті де ціна напряму залежить від цього.

Таблиця 5.4. Встановлення цінових меж

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Невідомо	Невідомо	700$+	50–250$

ВИСНОВКИ ДО РОЗДІЛУ

Спроєктована рекомендаційна система онлайн – книгарні має перспективи для успішного розвитку на ринку. Вона призначена для різноманітних користувачів і не обмежується специфічними галузями чи типами діяльності.

ВИСНОВКИ

В магістерській дипломній роботі було розроблено рекомендаційну систему онлайн – книгарні на основі засобів машинного навчання. Для демонстрації створеного алгоритму створено вебзастосунок. Набір даних зібрано за допомогою web scraping та збережено як базу даних SQLite. Рекомендаційний алгоритм фільтрує контент та забезпечує користувача списком книг, який відповідає інтересам та зацікавленням (які визначено на основі попередньої активності на сайті). Взаємодія з обмеженою кількістю релевантних пропозицій має сприяти підвищенню конверсії онлайн-книгарні. Отже, поставлена мета досягнута. Для її досягнення виконано наступні завдання:

1) аналіз предметної сфери, існуючих аналогів та джерел даних;

2) визначення логічної та фізичної моделі даних;

3) аналіз бізнес-процесів у системі;

4) специфікація вимог;

5) проєктування архітектури програмного забезпечення;

6) розробка алгоритму надання рекомендацій;

7) реалізація та тестування рекомендаційної системи.

Завдяки роботі над проєктом вдалось вдосконалити знання та навички, які є актуальними та затребуваними на ринку праці. На основі набутого досвіду зроблено висновок стосовно того, що найважливішим етапом роботи над застосунком є проєктування. Далі рішення на етапі розробки архітектури та алгоритмів полегшують всю подальшу роботи, економлять час та сприяють підвищенню якості програмного забезпечення.

Отже, створена рекомендаційна система відповідає всім вимогам та своєму призначенню. За потреби вона може бути вдосконалена та розширена. Сформований набір даних може бути використаний для інших проєктів та разом з іншими алгоритмами машинного навчання. Розроблений рекомендаційний алгоритм можна деталізувати та ускладнити за рахунок рейтингових балів для книг, а не лише бінарної оцінки у форматі «подобається» та «не подобається».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to Recommender Systems: Non-Personalized and Content-Based: University of Minnesota. URL: https://www.coursera.org/learn/recommender-systems-introduction.
2. Web data: Amazon reviews URL: https://snap.stanford.edu/data/web Amazon.html.
3. GroupLens Book-Crossing Dataset URL: https://grouplens.org/datasets/book-crossing/.
4. Goodbooks-10k: a new dataset for book recommendations URL: https:// fastml.com/goodbooks-10k-a-new-dataset-for-book-recommendations/.
5. Beer and Diapers: The Impossible Correlation: Steve Swoyer. URL: https:// tdwi.org/articles/2016/11/15/beer-and-diapers-impossible-correlation.aspx.
6. Implicit Feedback Recommendation System (I) — Intro and datasets EDA URL: https://medium.com/@teddywang0202/implicit-feedback-recommendation-system-i-intro-and-datasets-eda-eda16764602a.
7. How retailers can keep up with consumers: Ian MacKenzie. URL: https:// www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.
8. Google Data Analytics Professional Certificate: Google Career Certificates. URL: https://www.coursera.org/professional-certificates/google-data-analytics.
9. Learn SQL Basics for Data Science Specialization: University of California. URL: https://www.coursera.org/specializations/learn-sql-basics-data-science.
10. Про авторське право і суміжні права: Закон України від від 20.03.2023 № 2974-IX. URL: https://zakon.rada.gov.ua/laws/show/2811-20#Text.
11. Про захист персональних даних: Закон України від 29.07.2022 № 2494-IX URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text.
12. Martin R. C. Clean Architecture. Hoboken : Prentice Hall, 2018. 351 p.
13. Freeman Er., Freeman El., Sierra K. et al. Head First Design Patterns. Sebastopol : O'Reilly Media, Inc., 2004. 680 p.
14. Géron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow. Sebastopol : O'Reilly Media, Inc., 2017. 563 p.

15. Machine Learning Specialization: University of Washington. URL: https://www.coursera.org/specializations/machine-learning.
16. Python for Everybody Specialization: Charles Russell Severance. URL: https://www.coursera.org/specializations/python.
17. McKinney W. Python for Data Analysis. Sebastopol : O'Reilly Media, Inc., 2018. 540 p.
18. Sheena S. Iyengar, Mark R. Lepper. When Choice is Demotivating: Can One Desire Too Much of a Good Thing? Journal of Personality and Social Psychology. 2000. Vol. 79, no. 6. P. 995–1006.
19. Why Everyone's Worried About Their Attention Span—and How to Improve Yours: Time Magazine URL: https://time.com/6302294/why-you-cant-focus-anymore-and-what-to-do-about-it/.
20. Building a Recommender System from Scratch: Jil Cates URL: https://www.jillcates.com/pydata-workshop/html/tutorial.html.
21. Ukrainian Stopwords: Serhii Kupriienko URL: https://github.com/skupriienko/Ukrainian-Stopwords.

ДОДАТКИ

ДОДАТОК А

Базовий клас для сканерів web – сторінок:

```
class WebScraper(ABC):

    def __init__(self, initial_url, book_basic_url, result_file_path,
page_base_url = None):
        self.INITIAL_URL = initial_url
        self.BASIC_URL = book_basic_url
        self.result_file_path = result_file_path
        self.page_base_url = page_base_url

        self.soup = self.__parse_web_data(self.INITIAL_URL)


    # PARSING
    def __parse_web_data(self, url):
        ctx = ssl.create_default_context()
        ctx.check_hostname = False
        ctx.verify_mode = ssl.CERT_NONE

        html = urlopen(url, context=ctx).read()

        return BeautifulSoup(html, "html.parser")


    # PAGE LINKS
    @abstractmethod
    def generate_page_links(self, soup):
        pass

# BOOK LINKS
@abstractmethod
def generate_book_links(self, soup):
    pass


# BOOK DATA
@abstractmethod
def parse_book_data(self, book_link):
    pass


# PARSE
def parse(self):
    print(f"\nЗбір даних з сайту: {self.INITIAL_URL}\n")

    page_links = self.generate_page_links(self.soup)

    # ЗБІР ПОСИЛАНЬ НА КНИГИ
    book_links = []

    i = 1
    pages_count = len(page_links)

    for page_link in page_links:
```

```
            print(f"\rПошук посилань на книги на сторінці {i} з
{pages_count} ({page_link})\t\t\t", end = "\t")

            soup = self.__parse_web_data(page_link)

            page_book_links = self.generate_book_links(soup)
            book_links.extend(page_book_links)

            i = i + 1

        print()

        # СКАНУВАННЯ СТОРІНОК ПРО ОКРЕМІ КНИГИ
        books = []

        i = 1
        books_count = len(book_links)
        self.books_count = books_count


        for book_link in book_links:
            print(f"\rПросканована {i / books_count * 100:.0f}% сторінок
про окремі книги", end = "\t")

            book = self.parse_book_data(book_link)
            books.append(book)

            i = i + 1

        # Збереження результатів парсингу
        with open(self.result_file_path, "w") as f:
            json.dump(books, f, ensure_ascii = False)
            print(f"\nJSON-файл успішно збережено. Кількість книг:
{books_count}")
```

ДОДАТОК Б

Сканування веб – сторінок:

```
class NashFormatWebScraper(WebScraper):


    # PAGE LINKS
    def generate_page_links(self, soup):
        page_links = set()

        # Всі сторінки
        pages = [page.text for page in soup.select(".page-link")]

        # Остання сторінка - максимальна за номером
        last = int(pages[-1])

        # Генерація посилань за шаблоном
        for page_number in range(1, last + 1):
            url = self.page_base_url + str(page_number)
            page_links.add(url)

        return page_links


    # BOOK LINKS
    def generate_book_links(self, soup):
        products = soup.select(".product-list")

        return set([self.BASIC_URL + product.find_all("a")[0].get("href")
for product in products])

    # BOOK DATA
    def parse_book_data(self, book_link):
        soup = self._WebScraper__parse_web_data(book_link)

        book = dict()

        # Назва
        book["title"] = soup.find("h1").text

        # Характеристики
        keys = [key.text.strip() for key in soup.select(".attr")]
        values = [" ".join(value.text.strip().split()).strip() for value in
soup.select(".value")]

        book.update(zip(keys, values))

        # Ціна
        book["price"] = dict()

        sources = [".tabs-btn_paper", ".tabs-btn_electronic", ".tabs-
btn_audio"]

        for source in sources:
            data = soup.select(source)
            if data:
```

```
                data = data[0].text.split()
                book["price"][data[0]] = "".join(data[1:])

        # Тема
        tags = soup.select(".product_shelve")

        if tags:
            raw_meta = [elem.text for elem in tags[0].find_all("a")]
            meta = [re.findall(r"(.+)\([0-9]+\)", tag)[0].strip() for tag
in raw_meta]
            book["meta"] = ";".join(meta)


        book["description"] = soup.select("#annotation")[0].text.strip()

        # Посилання
        book["url"] = book_link

        # Обкладинка
        book["cover_img"] = soup.select(".fn-img")[0]["data-src"]

        return book
```

ДОДАТОК В

Запис даних в БД:

```
# MODELS

# Tables (many-to-many)
theme = db.Table("theme",
    db.Column("book_id", db.Integer, db.ForeignKey("book.id"), primary_key
= True, nullable=False),
    db.Column("tag_id", db.Integer, db.ForeignKey("tag.id"), primary_key =
True, nullable=False)
)

authors = db.Table("authors",
    db.Column("book_id", db.Integer, db.ForeignKey("book.id"), primary_key
= True, nullable=False),
    db.Column("author_id", db.Integer, db.ForeignKey("author.id"),
primary_key = True, nullable=False)
)

class Tag(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    tag = db.Column(db.String(255), unique=True, nullable=False)

class Author(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    author_name = db.Column(db.String(255), unique=True, nullable=False)
    books = db.relationship("Book", secondary=authors, backref="authors")

class Publisher(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    publisher_name = db.Column(db.String(255), unique=True, nullable=False)
    books = db.relationship("Book", backref="publisher")


class Book(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    publisher_id = db.Column(db.Integer, db.ForeignKey('publisher.id',
ondelete='CASCADE'), nullable=False)
    isbn = db.Column(db.String(255), nullable=False)
    book_title = db.Column(db.String(255), unique=True, nullable=False)
    age_limit = db.Column(db.String(255), nullable=False)
    is_for_school = db.Column(db.Boolean, nullable=False)
    language = db.Column(db.String(255), nullable=False)
    pages = db.Column(db.String(255), nullable=False)
    price = db.Column(db.Float, nullable=True)
    original_book_title = db.Column(db.String(255), nullable=True)
    size = db.Column(db.String(255), nullable=True)
    weight = db.Column(db.String(255), nullable=True)
    cover = db.Column(db.String(255), nullable=True)
    year = db.Column(db.Integer, nullable=True)
    description = db.Column(db.String(5000), nullable=False)
    url = db.Column(db.String(255), nullable=False)
    cover_url = db.Column(db.String(255), nullable=False)
    tags = db.relationship("Tag", secondary=theme, backref="books")
    actions = db.relationship("Action", backref="book")

class User(db.Model, UserMixin):
```

```
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    username = db.Column(db.String(255), nullable=False, unique=True)
    password = db.Column(db.String(255), nullable=False)
    actions = db.relationship("Action", backref="user")

class Action(db.Model):
    user_id = db.Column(db.Integer, db.ForeignKey('user.id',
ondelete='CASCADE'), primary_key=True, nullable=False)
    book_id = db.Column(db.Integer, db.ForeignKey('book.id',
ondelete='CASCADE'), primary_key=True, nullable=False)
    type = db.Column(db.String(255), nullable=False)


# DATA

def add_data_to_db():

    # context
    app.app_context().push()

    # schema
    db.create_all()

    with open("clean_json_data/books.json", "r") as file:
        books_json = json.load(file)

        for book_json in books_json:
            # Publisher
            publisher_name = book_json["Видавництво"]


            publisher =
Publisher.query.filter_by(publisher_name=publisher_name).first()
            if not publisher:
                publisher = Publisher(publisher_name = publisher_name)
                db.session.add(publisher)

            # Authors
            authors = list()
            authors_names = book_json["Автори"]
            for author_name in authors_names:
                author =
Author.query.filter_by(author_name=author_name).first()
                if not author:
                    author = Author(author_name = author_name)
                    db.session.add(author)
                    authors.append(author)

            # Tags
            tags = list()
            tags_names = book_json["Теги"]
            for tag_name in tags_names:
                tag = Tag.query.filter_by(tag=tag_name).first()
                if not tag:
                    tag = Tag(tag = tag_name)
                    db.session.add(tag)
                    tags.append(tag)

            # Book
            isbn_text = book_json["ISBN"]
```

```
            book_title = book_json["title"]

            age_limit = "універсальна"
            if "Вік" in book_json.keys():
                age_limit = book_json["Вік"]

            is_for_school = False
            if "Шкільна програма" in book_json.keys():
                is_for_school = True

            language = "українська"
            if "Мова" in book_json.keys():
                language = book_json["Мова"]

            pages_number = book_json["Сторінки"]

            price = None
            if "Паперова" in book_json["price"]:
                price = book_json["price"]["Паперова"]
                if price != "":
                    price = float(price.replace("грн", "").strip())
                else:
                    price = None
            elif book_json["price"] != "":
                price = float(book_json["price"].split(' ')
[0].replace("грн", "").strip().replace(",", ""))
            else:
                price = None


        original_book_title = None
        if "Оригінальна назва" in book_json.keys():
            original_book_title = book_json["Оригінальна назва"]

        size = book_json["Розмір"]

        weight = None
        if "Вага" in book_json.keys():
            weight = book_json["Вага"]

        cover = None
        if "Палітурка" in book_json.keys():
            cover = book_json["Палітурка"]

        year = None
        if "Рік видання" in book_json.keys():
            try:
                year = int(book_json["Рік видання"])
            except:
                year = None

        description = book_json["description"]

        url = book_json["url"]

        cover_url = book_json["cover_img"]

        book = Book.query.filter_by(book_title=book_title).first()
        if not book:
```

```
                try:
                    book = Book(isbn = isbn_text,
                                book_title = book_title,
                                age_limit = age_limit,
                                is_for_school = is_for_school,
                                language = language,
                                pages = pages_number,
                                price = price,
                                original_book_title = original_book_title,
                                size = size,
                                weight = weight,
                                cover = cover,
                                year = year,
                                description = description,
                                url = url,
                                cover_url = cover_url)
                    book.tags = tags
                    book.authors = authors
                    book.publisher = publisher

                    db.session.add(book)
                except:
                    print(price)

            db.session.commit()


if __name__ == '__main__':
    if not os.path.isfile("instance/books.db"):
        print("Initialize DB")
        add_data_to_db()

app.run(port=8000, debug=True)
```