

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» _____ 2025 р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Рекомендаційна система для e-commerce проектів»**

Виконала:

студентка IV курсу, групи КА-12

Фисак Софія Романівна

Керівник:

доцент, докт. філос., Гуськова Віра Геннадіївна

Консультант з економічного розділу:

доцент, к.е.н., Рощина Надія Василівна

Консультант з нормоконтролю:

к.ф.-м.н, Статкевич Віталій Михайлович

Рецензент:

асистент каф ІІІ, PhD,

Гаврилович Марія Павлівна

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» _____ 2025 р.

ЗАВДАННЯ
на дипломну роботу студентці
Фисак Софії Романівні

1. Тема роботи «Рекомендаційна система для e-commerce проєктів», керівник роботи Гуськова Віра Генадіївна доцент, PhD, затверджені наказом по університету від « 26 » травня 2025 р. № 1759-с
2. Термін подання студентом роботи _____
3. Вихідні дані до роботи: принципи побудови рекомендаційних систем, датасет Amazon Reviews 2023.
4. Зміст роботи: визначення ключових проблем сфери e-commerce; вибір та підготовка вхідних даних; опис використаних алгоритмів та моделей; розробка рекомендаційної системи для e-commerce проєкту; функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доцент, Рощина Надія Василівна		

7. Дата видачі завдання 14.04.2025.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Обробка літератури згідно теми	19.04.2025	Виконано
2.	Вибір та підготовка даних	24.04.2025	Виконано
3.	Написання першого розділу	29.04.2025	Виконано
4.	Написання другого розділу	10.05.2025	Виконано
5.	Розробка програмного продукту	17.05.2025	Виконано
6.	Написання третього розділу	23.05.2025	Виконано
7.	Підготовка економічної частини	25.05.2025	Виконано
8.	Проходження нормоконтролю	28.05.2025	Виконано
9.	Підготовка презентації	30.05.2025	Виконано

Студентка

Софія ФИСАК

Керівник

Віра ГУСЬКОВА

РЕФЕРАТ

Дипломна робота: 92 с., 16 рис., 8 табл., 25 джерел, 2 додатки.

ГІБРИДНА РЕКОМЕНДАЦІЙНА СИСТЕМА, ЕМОЦІЙНИЙ АНАЛІЗ, МАТРИЧНА ФАКТОРИЗАЦІЯ, E-COMMERCE, ОБРОБКА ПРИРОДНОЇ МОВИ.

Об'єкт дослідження – користувацькі оцінки та відгуки на товари з e-commerce платформи Amazon.

Предмет дослідження – алгоритми та методи побудови рекомендаційних систем, зокрема ті, що поєднують колаборативну фільтрацію та емоційний аналіз тексту.

Мета роботи – створити гібридну рекомендаційну систему для e-commerce, яка поєднує матричну факторизацію з емоційним аналізом відгуків користувачів, для надання персоналізованих та емоційно релевантних товарних рекомендацій.

Методи дослідження – методи машинного навчання, колаборативної фільтрації та емоційного аналізу з використанням сучасних трансформерів.

Актуальність – в умовах зростаючої конкуренції в e-commerce персоналізовані рекомендації з урахуванням емоційної складової здатні значно підвищити конверсію та задоволення користувачів

Результати роботи – реалізовано гібридну рекомендаційну систему для e-commerce-платформи. Система поєднує класичний підхід матричної факторизації з емоційним аналізом текстових відгуків, що дозволяє враховувати як кількісні, так і якісні аспекти користувацької взаємодії з товарами. Реалізація виконана мовою Python з використанням відповідних бібліотек для машинного навчання та обробки тексту.

Шляхи подальшого розвитку – масштабування системи на більші набори даних, удосконалення емоційної класифікації, а також адаптація системи до інших платформ e-commerce.

ABSTRACT

Diploma thesis: 92 p., 16 figs, 8 tables, 25 references, 2 appendices.

HYBRID RECOMMENDATION SYSTEM, EMOTIONAL ANALYSIS, MATRIX FACTORIZATION, TRANSFORMERS, E-COMMERCE, NATURAL LANGUAGE PROCESSING.

Object of research - user ratings and reviews of products from the Amazon e-commerce platform.

Subject of research - algorithms and methods for building recommender systems, in particular those that combine collaborative filtering and emotional text analysis.

Purpose of study - creating a hybrid recommender system for e-commerce that combines support vector machine factorisation with emotional analysis of user reviews to provide personalised and emotionally relevant product recommendations.

Research methods - methods of machine learning, collaborative filtering, natural and emotional analysis using modern transformers.

Relevance - in the context of growing competition in e-commerce, personalised recommendations with an emotional component can significantly increase conversion and user satisfaction

Results - a hybrid recommendation system for an e-commerce platform has been implemented. The system combines the classical approach of matrix factorisation with emotional analysis of text reviews, which allows taking into account both quantitative and qualitative aspects of user interaction with products. The implementation was done in Python with the use of appropriate libraries for machine learning and text processing.

Ways to further develop the subject of research - scale the system to larger data sets, improve emotional classification, and adapt the system to other e-commerce platforms.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	8
ВСТУП.....	9
РОЗДІЛ 1 АНАЛІЗ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДЛЯ РОБОТИ З ПРОДУКТОМ У СФЕРІ E-COMMERCE.....	10
1.1 Огляд рекомендаційних систем.....	10
1.2 Методи, що застосовуються в рекомендаційних системах для e-commerce.....	13
1.3 Аналіз існуючих рішень.....	16
1.4 Постановка проблеми, з якими стикаються e-commerce проекти у сфері персоналізації покупок.....	18
Висновки до розділу 1	21
РОЗДІЛ 2 ЗБІР ДАНИХ ТА ОПИС ЗАПРОПОНОВАНОГО ПІДХОДУ.....	22
2.1 Вимоги до вхідних даних для побудови ефективної рекомендаційної системи.....	22
2.2 Огляд обраних даних.....	22
2.2.1 Загальний опис.....	23
2.2.2 Підготовка даних.....	24
2.3 Опис використаних алгоритмів.....	24
2.3.1 Системи на основі вмісту (Content-based filtering)	24
2.3.2 Колаборативна фільтрація (Collaborative filtering).....	28
2.3.2.1 Види колаборативної фільтрації.....	29
2.3.3 Емоційний аналіз.....	30
Висновки до розділу 2.....	33
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТА ОЦІНКА РЕЗУЛЬТАТІВ.....	35
3.1 Опис запропонованого підходу.....	35
3.2 Побудова обраної рекомендаційної системи.....	35

3.3 Оцінювання та аналіз результатів	37
Висновки до розділу 3.....	41
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО	
ПРОДУКТУ.....	42
4.1 Постановка задачі проектування.....	42
4.2 Обґрунтування функцій програмного продукту.....	43
4.3 Обґрунтування системи параметрів програмного продукту.....	47
4.4 Аналіз експертного оцінювання параметрів.....	49
4.5 Аналіз рівня якості варіантів реалізації функцій.....	53
4.6 Економічний аналіз варіантів розробки ПП.....	54
4.7 Вибір кращого варіанту ПП техніко-економічного рівня.....	59
Висновки до розділу 4.....	59
ВИСНОВКИ.....	61
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	63
ДОДАТОК А ПРОГРАМНИЙ КОД.....	66
ДОДАТОК Б ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДО ДОПОВІДІ.....	85

ПЕРЕЛІК СКОРОЧЕНЬ

РС – рекомендаційна система

ФВА – функціонально-вартісний аналіз

ПП – Програмний продукт

ВСТУП

У сучасному житті вже неможливо уявити онлайн-магазин без персоналізованих рекомендацій. Кожного разу відкриваючи улюблений інтернет-сервіс, ми бачимо добірку товарів «для вас», «вам може сподобатись» або «разом купують». Саме тоді ми маємо справу з результатами роботи рекомендаційної системи (РС). Вона майже непомітна, але відіграє ключову роль у тому, що і як ми купуємо в цифровому середовищі.

У світі, де різноманітність і кількість товарів зростає швидше, ніж здатність людини їх охопити, РС стають містком між користувачем і продуктом. Вони допомагають не лише зекономити час, але й зробити вибір точнішим, комфортнішим, більш персоналізованим. Для e-commerce проєктів це означає не просто покращення сервісу — це інструмент, що прямо впливає на конверсії, прибуток і довіру клієнтів.

Кожна рекомендація передбачає складні алгоритми, аналіз поведінкових даних, створення профілів користувачів і продуктів, а також пошук прихованих шаблонів. Саме ці процеси складають основу інтелектуальних систем, які визначають конкурентоспроможність сучасних платформ.

Рекомендаційні системи — це не просто технологія, а один із проявів того, як штучний інтелект змінює повсякденну взаємодію з цифровим простором. У цій роботі розглядається, як ці системи працюють, які підходи використовуються в їхній розробці та чому саме для e-commerce є критично важливим впровадження таких технологій.

РОЗДІЛ 1 АНАЛІЗ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДЛЯ РОБОТИ З ПРОДУКТОМ У СФЕРІ E-COMMERCE

1.1 Огляд рекомендаційних систем

Рекомендаційні системи — це програмні інструменти та алгоритмічні рішення, призначені для надання персоналізованих пропозицій користувачам на основі їхніх інтересів, поведінки чи вподобань. Такі системи стали важливою частиною цифрового простору, де обсяги контенту й товарів давно перевищили можливості ручного пошуку та вибору. Вони орієнтують користувачів під час вибору бажаного товару серед величезної кількості варіантів, а бізнесам допомагають підвищити релевантність пропозицій, довіру клієнтів і, як наслідок, прибутковість.

Перші РС почали з'являтися ще у 90-х роках, які здебільшого базували на фільтрації інформації. У 1992 Девід Голдберг, Девід Ніколс, Браян М. Окі, Дуглас Террі запропонували систему під назвою Tapestry, яка вважається першою рекомендаційною системою на основі колаборативної фільтрації [1]. Саме з цього моменту почалися повноцінні дослідження рекомендаційних систем та розробка повноцінних рекомендаційних сервісів. Проте справжній прорив відбувся з розвитком великих даних (Big Data), доступу до поведінкової аналітики в реальному часі та методів машинного навчання, які дозволили автоматично виявляти складні патерни у взаємодії користувачів із системами.

Сьогодні рекомендаційні системи охоплюють майже всі сфери життя людини — від інтернет-магазинів, де вони допомагають покупцям знаходити бажаний товар, до стримінгових платформ, де контент підбирається на основі поведінки та вподобань користувачів. У соціальних мережах рекомендації впливають на стрічку новин, пропонуючи контент та можливих друзів. У

освітніх платформах — система може підібрати курси чи інший навчальний матеріал на основі потреб та наявних знань користувача.

Крім того, рекомендаційні системи усе активніше застосовуються в інших сферах. Наприклад в туризмі, при виборі маршрутів або готелів, у рекрутингу, допомагають знайти релевантні вакансії або працівників, в онлайн-банкінгу (персональні пропозиції фінансових продуктів) та навіть в інтернет-журналістиці, де вони формують новинні стрічки. Така масштабність стала можливою завдяки універсальності алгоритмів, які можуть бути адаптовані під будь-яку предметну галузь, де є дані про вибір, вподобання чи взаємодію користувачів.

Одним із найвідоміших прикладів ефективного використання рекомендаційних систем є Netflix. Використовуючи складні алгоритми, ця платформа формує персоналізовані добірки фільмів та серіалів, які максимально відповідають вимогам користувачів. Система аналізує історію переглядів, рейтинги, час взаємодії з контентом, а також схожість між користувачами, щоб запропонувати максимально релевантні варіанти.

Іншим прикладом є Coursera, освітня онлайн-платформа. Тут система рекомендацій допомагає користувачам знаходити нові курси, які відповідають їхнім інтересам, попереднім навчальним темам і прогресу курсу. Алгоритм також враховує популярність курсу, складність та відгуки інших студентів. У результаті користувачі отримують не тільки список доступних курсів, але й відповідний шлях навчання, щоб надихнути на залучення та успішне завершення курсу.

У сфері e-commerce рекомендовані системи стали одним із ключових елементів персоналізації. За дослідженнями McKinsey, близько 35% покупок на Amazon відбуваються завдяки роботі рекомендаційного механізму [2]. У Netflix завдяки рекомендаціям формується до 80% переглянутого контенту [3]. Це підтверджує їхню ефективність не лише з точки зору зручності, а й бізнес-цінності.

З технічного погляду, рекомендаційні системи можуть ґрунтуватися на різних підходах: простих евристиках, фільтрації за контентом, колаборативній фільтрації чи гібридних моделях. Їхнє впровадження можливе як у вигляді простого блоку на сайті (наприклад, «Схожі товари»), так і в рамках складної системи, яка постійно навчається на нових даних і оптимізує рекомендації в реальному часі.

Завдяки своїй гнучкості, масштабованості та здатності швидко вчитися на поведінці великої кількості користувачів, такі системи стали одним з основних інструментів сучасного цифрового маркетингу та підвищення якості взаємодії між платформою та користувачем. Вони дозволяють створювати персоналізований досвід, що значно підвищує залученість аудиторії та рівень задоволеності клієнтів. У довгостроковій перспективі це сприяє зростанню лояльності, повторних звернень і конкурентних переваг компанії на ринку.

Існує два найпоширеніші типи РС [4]:

1) Системи на основі вмісту (Content-based filtering).

Такі моделі будуються рекомендації на основі об'єктів, які схожі з тими, з яким користувач вже взаємодіяв. Головний принцип полягає в аналізі характеристик об'єктів (колір, категорія, жанр і тд.) та побудові профілю користувача на основі його вподобань. Найбільше ефективно такі системи працюють у онлайн-магазинах, стрімінгових-сервісах.

Серед переваг можна виділити, що дані моделю досить персоналізовані, дають релевантні рекомендації, а також добре працюють із нішовими продуктами, оскільки здатні ефективно рекомендувати не тільки популярні товари, а й маловідомий. Але системи на основі вмісту погано працюють в умовах браку інформації (наприклад, коли користувач є новим).

2) Система на основі колаборативної фільтрації (Collaborative filtering)

Такі рекомендаційні системи підбирають рекомендації на основі аналізу взаємодії користувачів із різними товарами. Рекомендації будуються враховуючи оцінку, відгуки, покупки чи іншу ключову дію.

Системи колаборативної фільтрації зазвичай мають дві складові:

- 1) Колаборативний фільтр Item-Item рекомендує товари, аналізуючи схожість з товарами, які цікавили користувача раніше. Тобто, якщо користувач позитивно оцінив один товар, то йому буде рекомендуватися схожий.
- 2) Колаборативний фільтр User-Item аналізує взаємодії між користувачами та товаром. У такому випадку, якщо два користувачі мають подібні вподобання, то і рекомендувати їм будуть подібні товари.

Колаборативна фільтрація найчастіше використовується у сферах інтернет товарів та соціальних мереж. Серед основних переваг можна виділи здатність рекомендувати нові або маловідомі товари в умовах обмеженості інформації та врахування соціального контексту.

1.2 Методи, що застосовуються в рекомендаційних системах для e-commerce

Рекомендаційні системи відіграють ключову роль в електронній комерції, забезпечуючи персоналізацію пропозицій, підвищення рівня продажів і покращення користувацького досвіду. Їхня основна мета — передбачити вподобання клієнтів і запропонувати товари чи послуги, які відповідають їхнім інтересам. Розвиток РС у сфері e-commerce пройшов кілька етапів, від простих підходів до складних систем, що використовують сучасні технології обробки даних.

На початкових етапах рекомендаційні системи базувалися на базових принципах аналізу даних. Найперші підходи ґрунтувалися на ідеї групування схожих користувачів або товарів за певними ознаками. Наприклад, клієнтам пропонувалися популярні продукти чи товари, які купували схожі за поведінкою користувачі. Такі методи були інтуїтивно зрозумілими, але мали

обмеження через низьку точність і неможливість врахувати індивідуальні вподобання.

З часом з'явилися підходи, які зосереджувалися на аналізі взаємодій між користувачами та товарами. Один із таких підходів передбачав використання даних про поведінку клієнтів (наприклад, покупки, перегляди сторінок) для знаходження закономірностей і формування рекомендацій на основі схожості між користувачами чи продуктами. Інший підхід фокусувався на характеристиках самих товарів, таких як їхня категорія, опис чи інші атрибути, щоб пропонувати продукти, подібні до тих, що вже цікавили користувача. Обидва методи мали свої сильні сторони, але стикалися з проблемами, такими як недостатня різноманітність рекомендацій або складність обробки нових користувачів чи товарів.

Подальший прогрес у сфері e-commerce був пов'язаний із комбінацією різних підходів. Такі гібридні системи намагалися об'єднати переваги аналізу поведінки користувачів і характеристик товарів, щоб підвищити точність і релевантність рекомендацій. Вони дозволили долати обмеження окремих методів, таких як проблема "холодного старту", коли система не має достатньо даних про нового користувача чи продукт.

З появою технологій обробки великих даних і штучного інтелекту рекомендаційні системи почали активно використовувати методи машинного навчання. Ці підходи дозволили моделювати складні взаємозв'язки між користувачами, товарами та контекстом (наприклад, часом або місцем покупки). Зокрема, системи почали враховувати динамічні фактори, такі як сезонність, географічне розташування чи тип пристрою, з якого здійснюється доступ до платформи. Крім того, з'явилися методи, які оптимізують рекомендації в реальному часі на основі зворотного зв'язку від користувачів, що значно підвищило їхню адаптивність.

Сучасні рекомендаційні системи в e-commerce також звертають увагу на різноманітність і новизну пропозицій, щоб уникнути так званої "бульбашки фільтрів", коли користувач отримує лише схожі товари, що обмежує його

вибір. Деякі підходи спрямовані на довгострокову взаємодію з клієнтами, пропонуючи рекомендації, які сприяють утриманню та підвищенню лояльності [5].

Цікавий підхід до моделювання РС пропонується в роботі [6]. У даній роботі автори розглядають застосування методів кластерного аналізу для формування рекомендацій. Розроблена модель групує схожі товари та клієнтів за їхніми характеристиками, наприклад категоріями чи схожою поведінкою. Для прикладу було розглянуто спортивний онлайн-магазин, де модель рекомендувала товар, а саме рюкзаки, на основі кластеризації за параметрами (розмір, бренд, ціна) та історії покупок клієнтів. За результатами, даний підхід підвищив точність рекомендацій на 12% в порівнянні з традиційною моделлю колаборативної фільтрації.

Також одним із провідних напрямів розробки рекомендаційних систем є впровадження нейронних мереж. Такий підхід допомагає знайти складні взаємозв'язки між товарами та роботи рекомендації більше різноманітними, не втрачаючи актуальності. Модель з використання нейронної мережі було описано в роботі [7]. У дослідженні пропонується модель глибокої нейронної мережі для рекомендації модних поєднань одягу в електронній комерції, яка інтегрує побічну інформацію, таку як зображення, текстові описи та категорії товарів. Модель поєднує візуальний модуль (з використанням конволюційних нейронних мереж для аналізу зображень), текстовий модуль (з обробкою природної мови для описів) і категорійний модуль (для кодування атрибутів), створюючи комплексне представлення предметів одягу. Ці дані об'єднуються через повнозв'язні шари для моделювання стилістичної сумісності, а спеціальна функція втрат оптимізує рекомендації на основі історичних даних. Підхід забезпечує персоналізовані пропозиції, враховуючи вподобання користувачів, і демонструє вищу точність порівняно з традиційними методами на даних реальних платформ.

1.3 Аналіз існуючих рішень

Електронна комерція стала значною частиною життя сучасної людини, де персоналізація відіграє ключову роль у залученні клієнтів і підвищенні продажів. Рекомендаційні системи, які лежать в основі багатьох успішних платформ, дозволяють пропонувати користувачам товари чи послуги, що відповідають їхнім інтересам, тим самим покращуючи досвід і сприяючи комерційному успіху. Далі розглядаються три відомі проєкти електронної комерції — Amazon, ASOS і Spotify, — кожен із яких представляє різні сегменти ринку, але всі вони демонструють потужність своїх рекомендаційних систем. Ці платформи, хоч і пропонують різні продукти, об'єднані спільною метою: зробити взаємодію з користувачем максимально релевантною та ефективною.

Amazon, гігант універсального ритейлу, пропонує все — від книг до електроніки. Його рекомендаційна система давно стала еталоном у галузі. Вона побудована на складному поєднанні кількох підходів, які дозволяють обробляти величезні обсяги даних. Основу становить аналіз поведінки користувачів: система відстежує, які товари клієнти купують, переглядають чи додають до кошика, і на основі цього пропонує схожі продукти. Наприклад, пропозиції на кшталт “інші клієнти також купили” з'являються завдяки порівнянню товарів, які цікавлять схожих користувачів. Окрім цього, Amazon враховує характеристики самих товарів, такі як категорія чи бренд, щоб рекомендувати схожі позиції. Усе це підсилюється глибокими нейронними мережами, які аналізують не лише дії користувачів, а й контекст — час доби, місце розташування чи навіть пристрій, з якого здійснюється доступ. Завдяки постійному тестуванню нових алгоритмів і врахуванню зворотного зв'язку, система адаптується до мінливих потреб. Такий підхід виявився надзвичайно успішним: дослідження показують, що третина всіх покупок на платформі генерується саме завдяки рекомендаціям. Це не лише сприяє імпульсивним

покупкам, але й скорочує час, який клієнт витрачає на пошук, що підвищує його задоволеність і лояльність. У 2023 році Amazon повідомила про значне зростання доходів, і значна частина цього успіху пов'язана з ефективністю її рекомендаційної системи [8].

Переходячи до світу моди, ASOS, британська платформа, що спеціалізується на одязі та аксесуарах, також покладається на персоналізацію, щоб завоювати серця молодих споживачів. Її рекомендаційна система вирізняється акцентом на візуальних даних, що є критично важливим для fashion-індустрії. Система аналізує зображення товарів, визначаючи стилі, кольори чи візерунки, щоб пропонувати клієнтам стилістично сумісні комбінації, наприклад, сукню разом із відповідними аксесуарами. Цей процес базується на технологіях глибокого навчання, які дозволяють обробляти візуальну інформацію з високою точністю. Водночас платформа відстежує послідовність дій користувачів — які товари вони переглядають чи додають до кошика, — щоб передбачити їхні вподобання. Текстові описи товарів і категорії також відіграють роль, допомагаючи системі знаходити схожі продукти. ASOS іде ще далі, враховуючи сезонність чи індивідуальні характеристики, такі як розмір або улюблені бренди, щоб зробити пропозиції максимально релевантними. Постійне тестування дозволяє платформі вдосконалювати алгоритми в реальному часі, що відображається у функціях на кшталт “вам також може сподобатися”. Успіх системи підтверджується цифрами: рекомендації генерують п'яту частину продажів платформи, а ймовірність покупки зростає завдяки точному підбору товарів. Крім того, персоналізовані пропозиції зменшують кількість повернень, оскільки клієнти отримують товари, які краще відповідають їхнім очікуванням [9].

Spotify, хоч і не є класичною платформою електронної комерції, пропонує цифровий контент і використовує рекомендаційні системи для продажу підписок і підвищення залучення користувачів. Її підхід до персоналізації став легендарним завдяки плейлистам на кшталт “Discover Weekly” чи “Daily Mix” [10]. Система Spotify спирається на аналіз поведінки

мільйонів користувачів, щоб знаходити схожі музичні вподобання та пропонувати треки, які можуть сподобатися. Але це лише частина картини. Платформа використовує технології обробки текстів для аналізу описів треків, жанрів і навіть відгуків у соціальних мережах, щоб краще зрозуміти контент. Глибоке навчання дозволяє обробляти послідовності прослуховувань і навіть аудіохарактеристики, такі як темп чи тональність, щоб створювати гармонійні плейлисти. Система також враховує контекст — час доби, місце чи настрій користувача, — щоб адаптувати пропозиції. Унікальним є використання методів, які оптимізують довгострокове залучення, адаптуючи рекомендації на основі того, як користувач реагує на запропонований контент. Цей підхід надзвичайно ефективний: плейлисти, створені системою, відповідають за значну частину прослуховувань на платформі, а зростання кількості преміум-підписок частково пояснюється персоналізацією. Користувачі повертаються до платформи, тому що відчують, що вона розуміє їхні смаки [11].

Ці три платформи — Amazon, ASOS і Spotify — демонструють, як рекомендаційні системи можуть трансформувати бізнес у різних сегментах електронної комерції. Усі вони використовують складні технології, що поєднують аналіз поведінки, характеристик продуктів і контексту, щоб створювати пропозиції, які не лише підвищують продажі, але й роблять взаємодію з платформою приємною та інтуїтивною. Їхній успіх підкреслює, що в сучасному світі персоналізація — це не просто додаткова функція, а необхідність для конкурентоспроможності.

1.4 Постановка проблеми, з якими стикаються e-commerce проекти у сфері персоналізації покупок

Персоналізація покупок стала основним викликом в роботі електронної комерції, адже здатність платформ пропонувати клієнтам товари, що

відповідають їхнім індивідуальним потребам, безпосередньо впливає на продажі, лояльність і конкурентоспроможність. Проте e-commerce проєкти стикаються з низкою складних проблем, які ускладнюють створення ефективних рекомендаційних систем. Ці виклики стосуються як технічних аспектів обробки даних, так і потреб користувачів, які постійно змінюються. Розуміння цих проблем є критично важливим для розробки рішень, здатних забезпечити високий рівень персоналізації.

Однією з основних перешкод є проблема недостатньої кількості даних, особливо для нових користувачів або товарів. Коли клієнт вперше відвідує платформу, система не має достатньої інформації про його вподобання, що ускладнює формування релевантних пропозицій. Аналогічно, нові продукти, які ще не мають історії переглядів чи покупок, важко інтегрувати в рекомендаційні алгоритми. Ця ситуація, відома як "холодний старт", знижує точність рекомендацій і може відштовхнути користувачів, які очікують миттєвої персоналізації. Різноманітні дослідження підкреслюють, що ця проблема особливо актуальна для платформ із широким асортиментом, де нові товари з'являються щодня [12].

Іншою значною перешкодою є обробка великих обсягів даних із високою розрідженістю. E-commerce платформи збирають величезну кількість інформації про поведінку користувачів — перегляди, кліки, покупки, — але більшість із цих даних є фрагментарними. Наприклад, клієнт може взаємодіяти лише з невеликою частиною каталогу, що ускладнює прогнозування його інтересів для інших категорій. Така розрідженість даних знижує ефективність традиційних алгоритмів, таких як колаборативна фільтрація, і вимагає застосування складніших моделей, які, своєю чергою, потребують значних обчислювальних ресурсів. Стаття в *Journal of Visual Communication and Image Representation* [13] зазначає, що ця проблема особливо відчутна в системах із мільйонами користувачів і товарів, де алгоритми повинні обробляти дані в реальному часі.

Ще однією складністю є забезпечення різноманітності рекомендацій. Часто системи, прагнучи максимізувати релевантність, пропонують користувачам товари, дуже схожі на ті, що вони вже переглядали чи купували. Це може призвести до так званої "бульбашки фільтрів", коли клієнт втрачає можливість відкривати нові продукти чи категорії. Такий підхід обмежує потенціал продажів і може викликати відчуття одноманітності [14].

Контекстна адаптація також становить проблему. Користувачі очікують, що рекомендації враховуватимуть не лише їхні минулі дії, а й поточні обставини — час, місце, пристрій чи навіть сезон. Наприклад, пропозиція зимового одягу влітку або вечірньої музики вранці може бути сприйнята як нерелевантна. Однак збір і обробка контекстних даних у реальному часі вимагають складної інфраструктури та швидкодії.

Нарешті, етичні та конфіденційні аспекти створюють додаткові виклики. Персоналізація вимагає збору великих обсягів особистих даних, що викликає занепокоєння щодо їхньої безпеки та відповідності регуляціям, таким як GDPR. Користувачі можуть негативно реагувати на надмірну персоналізацію, якщо відчують, що їхні дані використовуються без належної прозорості. Згідно з дослідженням Forbes [15], велика кількість споживачів готові припинити користуватися платформою, якщо відчують порушення конфіденційності. Це змушує e-commerce проекти балансувати між персоналізацією та захистом даних, що ускладнює розробку рекомендаційних систем.

Отже, персоналізація покупок у сфері електронної комерції стикається з низкою перешкод, від технічних обмежень, таких як "холодний старт" і розрідженість даних, до необхідності забезпечити різноманітність, контекстну релевантність і етичне використання інформації. Ці проблеми вимагають комплексного підходу, який поєднує передові технології, такі як глибоке навчання, із стратегіями, що враховують очікування користувачів і регуляторні вимоги. Розв'язання цих викликів є ключем до створення

рекомендаційних систем, які не лише підвищують продажі, а й зміцнюють довіру клієнтів.

Висновки до розділу 1

Рекомендаційні системи є ключовим елементом сучасної електронної комерції, сприяючи персоналізації пропозицій і підвищенню рівня продажів. Проведений аналіз показав, що РС еволюціонували від простих методів фільтрації до складних систем, які використовують машинне навчання та нейронні мережі. Основні підходи, такі як контентна та колаборативна фільтрація, а також гібридні моделі, дозволяють ефективно обробляти дані користувачів і товарів. Вивчення рішень платформ Amazon, ASOS і Spotify продемонструвало, як персоналізація адаптується до різних ринкових сегментів, підвищуючи залученість і лояльність клієнтів. Проте e-commerce проєкти стикаються з низкою проблем, зокрема "холодним стартом", розрідженістю даних, необхідністю забезпечення різноманітності та контекстної релевантності рекомендацій. Етичні аспекти та захист даних також ускладнюють розробку РС. Ці виклики вимагають комплексного підходу, що поєднує передові технології та стратегії, орієнтовані на користувача. Розуміння цих аспектів створює основу для розробки ефективних рекомендаційних систем, які зміцнюють конкурентоспроможність платформ.

РОЗДІЛ 2 ЗБІР ДАНИХ ТА ОПИС ЗАПРОПОНОВАНОГО ПІДХОДУ

2.1 Вимоги до вхідних даних для побудови ефективної рекомендаційної системи

Для ефективної роботи рекомендаційної системи в електронній комерції необхідно забезпечити якісний збір і обробку вхідних даних, які є основою для створення персоналізованих пропозицій. Вхідні дані мають включати інформацію про поведінку користувачів, таку як історія покупок, перегляди товарів, кліки, додавання до кошика та пошукові запити, що дозволяє прогнозувати їхні вподобання. Важливими є також характеристики товарів, зокрема категорії, бренди, ціни, описи та зображення, які забезпечують можливість порівняння продуктів і пропонування схожих. Контекстні дані, такі як час взаємодії, геолокація, тип пристрою чи сезонність, сприяють адаптації рекомендацій до поточних потреб клієнта. Для уникнення проблем із "холодним стартом" бажано мати додаткові дані, наприклад, демографічну інформацію користувачів або провести невелике опитування, щоб отримати інформацію про потреби та бажання клієнта. Обсяг даних повинен бути достатнім для навчання моделей, але водночас система має бути здатна обробляти їх у реальному часі для динамічної персоналізації.

2.2 Огляд обраних даних

Для побудови рекомендаційної системи в даній роботі було вирішено обрати дані Amazon Reviews 2023 [16]. Це великий масив даних, зібраний

командою McAuley Lab, який містить відгуки користувачів про продукти на платформі Amazon, а також супутні метадані. Використання цього набору даних дозволяє ефективно аналізувати вподобання користувачів та створювати персоналізовані рекомендації. Крім того, дані включають різноманітні категорії продуктів, що забезпечує широке охоплення та можливість тестування системи в різних контекстах.

2.2.1 Загальний опис

Набір даних Amazon Reviews 2023 є оновленою версією попередніх наборів даних Amazon. Він включає понад 500 мільйонів відгуків про 48 мільйонів товарів із 33 категорій, що охоплюють період із травня 1996 року до вересня 2023 року [16]. Дані містять багатий набір характеристик, які ідеально підходять для побудови рекомендаційних систем у сфері e-commerce.

Саме ці дані містять в собі таку важливу інформацію, як:

- 1) Відгуки користувачів (рейтинги, текст, голосування за корисність тощо),
- 2) Метадані товару (опис, ціна, необроблене зображення тощо).

Структура даних.

Частина з відгуками користувачів містить таку інформацію, як:

- `user_id`: Унікальний ідентифікатор користувача, який залишив відгук (анонімізований).
- `item_id` (або `asin`): Унікальний ідентифікатор товару (Amazon Standard Identification Number).
- `rating`: Оцінка товару (число від 1 до 5).
- `title`: Заголовок відгуку користувача
- `text`: Текстовий відгук користувача.
- `helpfulness`: Показник корисності відгуку (наприклад, відношення позитивних голосів до загальної кількості).

- **timestamp:** Дата та час написання відгуку.

Частина з метаданими містить в собі:

- **rating_number:** оцінка товару
- **parent_asin:** Унікальний ідентифікатор товару, що зв'язує метадані з відгуками.
- **title:** Назва товару.
- **description:** Текстовий опис товару.
- **price:** Ціна товару (може бути в числовому форматі або відсутньою).
- **details:** Містить в собі характеристики товару (колір, розмір, бренд, матеріал, з якого виготовлений товар)
- **images:** Посилання на зображення товару.
- **main_category:** Список категорій, до яких належить товар.
- **categories:** Ієрархічні категорії товару.
- **features:** Додаткові характеристики товару (наприклад, технічні специфікації).

2.2.2 Підготовка даних

Для роботи потрібно виконати попередню обробку даних.

Оскільки масив даних із відгуками є надзвичайно великим, було прийнято рішення скоротити його, обмеживши період. Для побудови моделі буде взято тільки найактуальніше дані за 2023 рік. Приклад таблиці із відгуками зображено на рисунку 2.1.

	rating	title	text	images	asin	parent_asin		user_id	timestamp	helpful_vote	verified_purchase	date
32	1.0	halo hair extensions	This halo hair extension is simply put, garbag...	[]	B0BFR5WF1R	B0BFR5WF1R	AFZUK3MTBIBEDQOPAK3OATUOUKLA	1675826333052		4	True	2023-02-08 03:18:53.052
379	1.0	Crap	These look and fit BAD. Fake looking.<br ...	[]	B0BL3HSBZB	B0BL3HSBZB	AF5PN3FPG5Z66P7Z7UWL56D6CGMA	1674411398983		0	True	2023-01-22 18:16:38.983
444	4.0	Good...But Not My Favorite	Despite having a thick-ish, creamy consistency...	[]	B0BSR6WK1Q	B0BXB4J297	AGAM2CCKV52H4YZU7ASZTSXA7YQ	1678571121367		0	False	2023-03-11 21:45:21.367
468	4.0	Nice Formula!	Very "rusty" color.	[]	B07TT8JK51	B07TT8JK51	AGAHANLSS7DG4ZHNP5S56W4SKHA	1672923265267		0	True	2023-01-05 12:54:25.267
549	1.0	over priced, over hyped, doesn't work	Chose warm vanilla and the fragrance is wonder...	[]	B09XBSDCXP	B09XBSDCXP	AFH7ZT2NJSJHDKJ43IODXCE6RHWVA	1676202555202		1	True	2023-02-12 11:49:15.202

Рисунок 2.1 – Частина таблиці із відгуками

Також, на рисунку 2.2 показано приклад таблиці із метаданими. Як сказано в попередньому пункті, ця таблиця містить інформацію про доступні товари.

	main_category	title	average_rating	rating_number	features	description	price	images	store	categories	details	parent_asin	bought_together
0	All Beauty	Howard LC0008 Leather Conditioner, 8-Ounce (4...	4.8	10	[]	[]	NaN	['thumb': 'https://m.media-amazon.com/images/...	Howard Products	[]	{'Package Dimensions': '7.1 x 5.5 x 3 inches; ...	B01CUPMQZE	None
1	All Beauty	Yes to Tomatoes Detoxifying Charcoal Cleanser ...	4.5	3	[]	[]	NaN	['thumb': 'https://m.media-amazon.com/images/...	Yes To	[]	{'Item Form': 'Powder', 'Skin Type': 'Acne Pro...	B076WQZGPM	None
2	All Beauty	Eye Patch Black Adult with Tie Band (6 Per Pack)	4.4	26	[]	[]	NaN	['thumb': 'https://m.media-amazon.com/images/...	Levine Health Products	[]	{'Manufacturer': 'Levine Health Products'}	B000B658RI	None
3	All Beauty	Tattoo Eyebrow Stickers, Waterproof Eyebrow, 4...	3.1	102	[]	[]	NaN	['thumb': 'https://m.media-amazon.com/images/...	Cherioll	[]	{'Brand': 'Cherioll', 'Item Form': 'Powder', '...	B088FKY3VD	None

Рисунок 2.2 – Частина таблиці із метаданими

2.3 Опис використаних алгоритмів

2.3.1 Системи на основі вмісту (Content-based filtering)

Контентна фільтрація є одним із ключових підходів до створення рекомендаційних систем, який використовується в електронній комерції для персоналізації пропозицій товарів. Цей метод базується на аналізі характеристик (атрибутів) товарів та профілів користувачів, щоб рекомендувати продукти, схожі на ті, що користувач уже оцінив позитивно. Наприклад, якщо користувач залишив високий рейтинг на товари з категорії "Електроніка", система запропонує інші товари з цієї категорії або з подібними характеристиками (наприклад, бренд, ціна, технічні специфікації).

Фільтрація на основі контенту ґрунтується на припущенні, що користувач надасть перевагу товарам, які мають схожі характеристики з тими, що він оцінив раніше [17]. Основна ідея полягає в тому, щоб:

1. Сформулювати профіль користувача на основі його попередніх взаємодій (оцінок, відгуків);
2. Порівняти профіль користувача з атрибутами товарів, щоб знайти найбільш схожі;
3. Порекомендувати товари з найвищим показником схожості.

Ідея даного підходу зображена на рисунку 2.3 [18].

CONTENT-BASED FILTERING

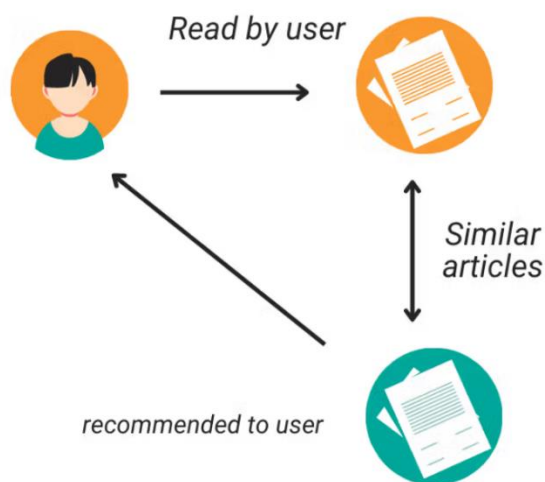


Рисунок 2.3 - Схема алгоритму роботи контентної фільтрації

Важливим частною роботи цієї системи є знаходження схожості. Спочатку кожен елемент подається у вигляді вектору його ознак [19].

$$\vec{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]$$

x_{ij} — значення j -ої ознаки товару i ,

n — кількість ознак (характеристик)

Потім знаходить профіль користувача на основі його вподобань або попередніх дій.

$$P_u = \frac{1}{|I_u^+|} \sum_{i \in I_u^+} \vec{x}_i$$

I_u^+ — множина товарів, які користувач оцінив позитивно,

$\vec{x}_i$ — вектор товару

Далі вираховується схожість між користувачем та елементом. Чи більше значення схожості, тим з більшою ймовірністю цей елемент буде рекомендуватися користувачеві. Є різні алгоритми пошуку схожості :

1. Косинусна схожість

У такому підході шукається косинус кута між векторами [20]:

$$s(q, x) = \cos(q, x)$$

2. Скалярний добуток

Схожість шукаємо як скалярний добуток векторів

$$s(q, x) = \langle q, x \rangle = \sum_{i=1}^d q_i x_i$$

3. Евклідова відстань

У цьому підході схожість обернено пропорційна евклідовій відстані: чим мені відстань, тим більша схожість:

$$s(q, x) = \|q - x\| = \left[\sum_{i=1}^d (q_i - x_i)^2 \right]^{\frac{1}{2}}$$

Вже після знаходження схожості, будуть рейтинг елементів для рекомендації. Часто ще додається етап перенаджування, щоб врахувати особливості окремих елементів чи користувачів. Після чого видається остаточний результат. На цьому етапі також можуть застосовуватись евристики або бізнес-правила, наприклад, пріоритезація нових або акційних товарів. У деяких системах враховується контекст, як-от час доби, сезонність або місцезнаходження користувача. Крім того, для підвищення різноманітності результатів іноді додають елементи, що не повністю відповідають профілю, але можуть бути цікавими. У результаті система надає персоналізовані рекомендації, що поєднують релевантність, новизну та індивідуальні вподобання. Наприклад, якщо двоє користувачів мають схожі

інтереси, але один з них нещодавно здійснив покупку, система може знизити рейтинг вже придбаних товарів для цього користувача. Або, якщо певний товар має високий рейтинг, але закінчується на складі, він може бути переміщений нижче у списку. Таке перенаджування дозволяє адаптувати рекомендації до актуальних умов і дій користувача в реальному часі.

2.3.2 Колаборативна фільтрація (Collaborative filtering)

Колаборативна фільтрація є одним з найдавніших та найпоширеніших методів у системах рекомендацій. Цей підхід базується на припущенні, що користувачі, які мали схожі інтереси або уподобання в минулому, матимуть їх і в майбутньому. На відміну від контентно-орієнтованих методів, колаборативна фільтрація не вимагає знань про самі об'єкти, а використовує лише поведінкові дані користувачів: оцінки, перегляди, покупки. Загальна схема роботи колаборативного фільтру зображена на рисунку 2.4 [21].

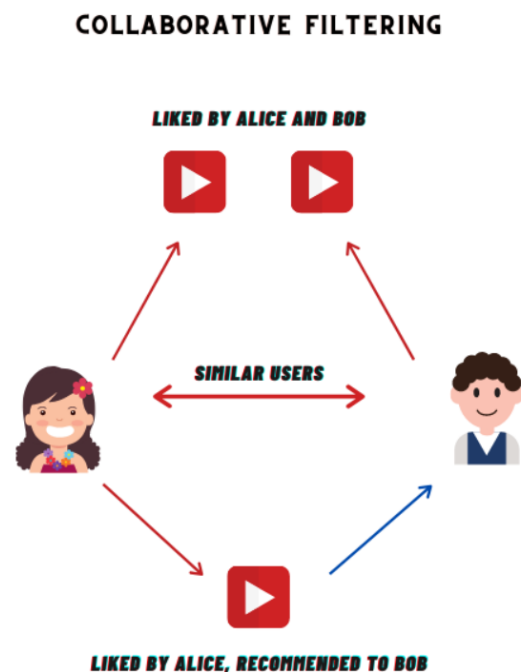


Рисунок 2.4 - Схема алгоритму роботи колаборативної фільтрації

Колаборативна фільтрація передбачає наявність великої матриці взаємодій $R \in \mathbb{R}^{m \times n}$, де m — кількість користувачів, а n — кількість об'єктів (фільмів, товарів тощо) [22]. Елемент r_{ui} у цій матриці — це рейтинг, який користувач u надав об'єкту i , або індикатор взаємодії (наприклад, клік або покупка). Ідея методу полягає в тому, щоб передбачити невідомі елементи матриці R , використовуючи відомі оцінки та подібності між користувачами або об'єктами.

2.3.2.1 Види колаборативної фільтрації

Першим видом колаборативної фільтрації є User-User Collaborative Filtering. У цьому методі для кожного користувача u знаходяться k найближчих сусідів (інших користувачів), які мають схожі вподобання. Схожість між користувачами визначається, наприклад, за косинусною метрикою або коефіцієнтом кореляції Пірсона:

$$\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

де I_{uv} — множина об'єктів, оцінених обома користувачами, а $\bar{r}_u$ — середній рейтинг користувача u . Оцінка невідомого рейтингу:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N_k(u)} \text{sim}(u, v) \cdot (r_{vi} - \bar{r}_v)}{\sum_{v \in N_k(u)} |\text{sim}(u, v)|}$$

Другим видом є Item-Item Collaborative Filtering. Метод працює на оцінюванні схожості між об'єктами враховуючи оцінки користувачів. Наприклад, якщо користувач раніше оцінив фільм "Інтерстеллар", система

може запропонувати "Початок", якщо багато інших користувачів поставили високі оцінки обом фільмам. Прогноз для рейтингу користувача u об'єкта i :

$$\hat{r}_{ui} = \frac{\sum_{j \in N_k(i)} \text{sim}(i, j) \cdot r_{uj}}{\sum_{j \in N_k(i)} |\text{sim}(i, j)|}$$

Цей метод ефективно використовується в системі Amazon для рекомендацій товарів. Такий підхід добре масштабується при великій кількості користувачів, оскільки обчислення подібності між об'єктами є менш чутливим до змін у поведінці окремих користувачів. Крім того, подібність між елементами змінюється рідше, що дозволяє кешувати результати та пришвидшити генерацію рекомендацій.

2.3.3 Емоційний аналіз

Емоційний аналіз, або аналіз настрою (Sentiment Analysis), є однією з найцікавіших і найдинамічніших областей обробки природної мови (NLP). Його основна мета полягає у визначенні емоційного тону, ставлення або суб'єктивної інформації, що міститься в тексті. Найпоширенішим використанням емоційного аналізу є визначення, чи є висловлювання позитивним, негативним чи нейтральним. Однак сучасний емоційний аналіз виходить далеко за межі цієї тривіальної класифікації. Він може також виявляти конкретні емоції, такі як радість, гнів, смуток, страх, здивування або відраза.

Методи емоційного аналізу дозволяють комп'ютерам "розуміти" людські почуття, виражені у тексті, що раніше було виключно прерогативою людського інтелекту [23]. Ця здатність має величезне значення у світі, де

щодня генеруються величезні обсяги текстових даних: від твітів і коментарів у соціальних мережах до оглядів продуктів та новинних статей. Розуміння громадської думки, настроїв клієнтів або емоційного фону дискусій стає критично важливим для багатьох галузей.

Процес здійснення емоційного аналізу зазвичай включає кілька етапів. По-перше, відбувається збір та попередня обробка текстових даних. Це може включати очищення тексту від шумів, таких як HTML-теги або спеціальні символи, приведення до нижнього регістру, лематизацію або стемінг для зведення слів до їхньої базової форми. По-друге, необхідне представлення тексту у форматі, зрозумілому для машин, тобто векторизація. Історично для цього використовувалися такі методи, як "мішок слів" (Bag-of-Words) [23] або TF-IDF, які перетворювали текст на числові вектори, що відображали частоту або важливість слів.

З розвитком глибокого навчання, особливо архітектур трансформерів, підходи до представлення тексту значно покращилися. Тепер використовуються вбудовування слів (Word Embeddings), такі як Word2Vec, GloVe або FastText [24], які дозволяють захопити семантичні та синтаксичні зв'язки між словами. Ще більш потужними є контекстуальні вбудовування від таких моделей, як BERT, RoBERTa або GPT. Ці моделі можуть генерувати різні векторні представлення для одного й того ж слова залежно від його контексту в реченні, що є революційним для розуміння тонких смислових відтінків.

Після векторизації використовуються різні моделі машинного навчання або глибокого навчання для класифікації тексту за настроєм. Простіші моделі включають наївний Баєс, SVM (метод опорних векторів) або логістичну регресію. Більш складні та потужні рішення ґрунтуються на нейронних мережах, таких як рекурентні нейронні мережі (RNN), LSTM (довга короткочасна пам'ять) або згорткові нейронні мережі (CNN). Однак, як було зазначено вище, домінуючими сьогодні є моделі на основі архітектури Transformer.

Transformer-моделі, такі як BERT (Bidirectional Encoder Representations from Transformers) і його оптимізовані версії, як-от RoBERTa (Robustly Optimized BERT Pretraining Approach), використовують механізм Self-Attention. Цей механізм дозволяє моделі одночасно враховувати всі слова в реченні при обробці кожного окремого слова, захоплюючи складні взаємозв'язки та залежності на великих відстанях у тексті. Це робить їх надзвичайно ефективними для завдань, що вимагають глибокого розуміння контексту, таких як емоційний аналіз, де тонкі відтінки мови можуть повністю змінити значення висловлювання (наприклад, іронія) [25].

Приклади використання емоційного аналізу в рекомендаційних системах є численними та надзвичайно корисними. Наприклад, він дозволяє покращити кількісні рейтинги якісним розумінням. Традиційні рекомендаційні системи часто покладаються виключно на числові оцінки (наприклад, від 1 до 5 зірок). Однак рейтинг "3 зірки" може означати "непогано, але є недоліки" або "добре, але очікування були вищими". Аналіз настрою тексту відгуку може розкрити ці нюанси. Якщо користувач дав 3 зірки, але написав дуже позитивний відгук, система може інтерпретувати це як більш позитивний сигнал, ніж просто середню оцінку.

Також, емоційний аналіз допомагає виявляти причини низьких або високих оцінок. Аналізуючи негативні відгуки, система може виявити конкретні аспекти продукту чи послуги, які викликають незадоволення (наприклад, "камера погана", "доставка повільна", "сервіс недбалый"). Ця інформація є безцінною не тільки для покращення продукту, але й для більш точних рекомендацій. Наприклад, якщо користувач постійно висловлює негатив щодо часу автономної роботи телефону, система може рекомендувати моделі з вищою ємністю батареї, навіть якщо загальний рейтинг попередніх телефонів був середнім.

Емоційний аналіз сприяє персоналізації рекомендацій на глибшому рівні. Знаючи емоційні реакції користувача на різні атрибути товарів (наприклад, "обожнюю яскраві кольори", "ненавиджу надмірну складність інтерфейсу"),

система може формувати рекомендації, що краще відповідають не лише функціональним, а й емоційним потребам. Це особливо актуально для товарів, де емоційний компонент відіграє велику роль, таких як одяг, мистецтво, розваги або косметика.

Ще емоційний аналіз дозволяє вчасно відслідковувати зміни в настроях користувачів з часом. Це допомагає адаптувати рекомендації до змінюваних вподобань користувача або до зміни репутації товару чи бренду. Якщо раніше користувач мав нейтральний настрій щодо певного жанру фільмів, але останні його відгуки стають дуже позитивними, система може активніше рекомендувати фільми цього жанру.

Нарешті, емоційний аналіз є потужним інструментом для раннього виявлення проблем з товарами чи послугами. Якщо велика кількість користувачів починає висловлювати негативний настрій щодо конкретного продукту, навіть до того, як це відобразиться на середньому рейтингу, система може попередити розробників або менеджерів про потенційну проблему. Це дозволяє оперативно реагувати та запобігати подальшому зниженню задоволеності клієнтів.

Таким чином, інтеграція емоційного аналізу в рекомендаційні системи перетворює їх з простих "машинок для співставлення" на більш "інтелектуальних агентів", здатних розуміти та реагувати на складні аспекти людської поведінки та емоцій. Це відкриває нові можливості для створення дійсно персоналізованих та ефективних рекомендацій, що підвищують задоволеність користувачів і сприяють успіху бізнесу.

Висновки до розділу 2

У розділі два було розглянуто загальні вимоги до даних, які використовуються для побудови рекомендаційних систем. Будо описано

вибрані для цієї роботи дані, а саме Amazon Reviews 2023. Розглянуті особливості та структуру цих даних.

Також було розглянуто наявні рекомендаційні системи. Описано найпопулярніші РС, а саме: контентна фільтрація та колаборативне фільтрування. Розкрито загальні принципи цих методів, зокрема, як вони формують рекомендації на основі схожості між елементами або користувачами. Контентна фільтрація дозволяє пропонувати нові товари на основі атрибутів, які вже сподобались користувачу. Натомість колаборативне фільтрування використовує досвід інших користувачів і ефективно виявляє приховані патерни у взаємодіях. Окрему увагу приділено item-based підходу, який добре масштабується та застосовується у промислових рішеннях, таких як Amazon.

Таким чином, у розділі було висвітлено основні типи рекомендаційних систем, що слугують базисом для побудови гібридної моделі, запропонованої в наступних розділах. Результати аналізу дозволяють перейти до етапу реалізації моделі та тестування її ефективності на реальних даних.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТА ОЦІНКА РЕЗУЛЬТАТІВ

3.1 Опис запропонованого підходу

Основною ідеєю запропонованої роботи є прогнозування рейтингу користувача враховуючи не лише його історичні вподобання та характеристики товару, але й емоційний контекст, виражений в його відгуках та емоційних реакціях схожих користувачів.

Система реалізує гібридний підхід до рекомендацій, який поєднує традиційну колаборативну фільтрацію (матричну факторизацію) з сучасним емоційним інтелектом (трансформаторний аналіз настроїв). Основна гіпотеза для створення саме такої моделі полягає в тому, що вподобання користувачів базуються не лише на історичних взаємодіях, але й на них суттєво впливають емоційні реакції на продукти, які можна зрозуміти з відгуків.

3.2 Побудова обраної рекомендаційної системи

Вибір нейронної матричної факторизації як основного алгоритму колаборативної фільтрації обумовлений кількома ключовими факторами. По-перше, класична матрична факторизація, незважаючи на свою елегантність та ефективність, обмежена лінійністю взаємодій між латентними факторами. Нейронні мережі дозволяють моделювати складні нелінійні залежності, що особливо важливо для розуміння багатовимірної природи людських вподобань. По-друге, архітектура на основі PyTorch забезпечує гнучкість у

налаштуванні моделі та можливість ефективного навчання на графічних процесорах, що критично важливо для обробки великих обсягів даних.

Архітектура нейронної мережі спроектована таким чином, щоб максимально ефективно кодувати інформацію про користувачів та товари. Ембедінг-шари перетворюють дискретні ідентифікатори у високорозмірні векторні представлення, де кожен вимір може інтерпретуватися як прихований фактор впливу на вподобання. Розмірність ембедінгів була обрана з урахування таких чинників, як виразність моделі та обчислювальна ефективність. Додавання bias-термів для кожного користувача та товару дозволяє врахувати індивідуальні особливості оцінювання, такі як схильність конкретного користувача до завищених або занижених оцінок, або загальну популярність певних товарів.

Другим стовпом системи є модуль емоційного аналізу, побудований на основі моделі RoBERTa від Cardiff University. Вибір саме цієї моделі зумовлений її спеціалізацією на аналізі настрою в соціальних медіа, що робить її особливо придатною для обробки користувацьких відгуків. RoBERTa представляє собою удосконалену версію архітектури BERT, де застосовано оптимізовану стратегію навчання та видалено завдання прогнозування наступного речення, що дозволяє моделі краще фокусуватися на розумінні контексту в межах одного документа.

Архітектура Transformer, що лежить в основі RoBERTa, використовує механізм self-attention для створення контекстуально-залежних представлень слів. Це означає, що значення кожного слова в тексті визначається не лише його власною семантикою, але й взаємодією з усіма іншими словами в реченні. Така особливість робить модель надзвичайно ефективною для розуміння тонких нюансів мови, включаючи іронію, сарказм та складні емоційні конструкції, що часто зустрічаються в користувацьких відгуках.

Оцінка настрою відгуку інтегрується в модель матричної факторизації як додатковий лінійний фактор, що дозволяє скоригувати прогнозований рейтинг. Наприклад, позитивний настрій може трохи підвищити

прогнозований рейтинг, а негативний – знизити, відображаючи не лише те, що користувач "подобається" товар, а й те, наскільки сильно виражені ці емоції в його відгуку.

На рисунку 3.1 зображено схему запропонованої моделі.



Рисунок 3.1 – Схема запропонованої моделі

3.3 Оцінювання та аналіз результатів

Для оцінювання моделі було вибрано декілька метрик:

- 1) RMSE: 0.7193
- 2) MAE: 0.5103
- 3) Accuracy: 0.8286
- 4) Precision: 0.9500
- 5) Recall: 0.7917
- 6) F1 Score: 0.8636

Отримані результати демонструють досить високу ефективність розробленої рекомендаційної системи, особливо враховуючи складність задачі передбачення користувацьких уподобань.

RMSE у 0.7193 та MAE у 0.5103 свідчать про досить точні передбачення рейтингів. Враховуючи, що рейтинги варіюються від 1 до 5, середня помилка в половину бала (MAE = 0.51) є цілком прийнятною для практичного

застосування. Це означає, що якщо користувач поставив би товару 4 зірки, модель в середньому передбачить від 3.5 до 4.5 зірок, що є досить точним результатом. Особливо вражаючими є результати бінарної класифікації позитивних та негативних відгуків. Ассигасу у 82.86% означає, що більше ніж у 4 з 5 випадків модель правильно визначає, чи сподобається користувачу товар.

Precision у 95% є виключно високим показником - це означає, що коли модель передбачає позитивний відгук, вона помиляється лише в 5% випадків. Для бізнес-застосування це критично важливо, оскільки невдалі рекомендації можуть призвести до незадоволення користувачів.

Recall у 79.17% показує, що модель знаходить приблизно 4 з 5 товарів, які дійсно сподобаються користувачу. Хоча цей показник дещо нижчий за precision, він все ще залишається на високому рівні.

F1-score у 86.36% демонструє гарний баланс між precision та recall. Це особливо важливо для рекомендаційних систем, де потрібно одночасно мінімізувати невдалі рекомендації (високий precision) та максимізувати кількість знайдених релевантних товарів (високий recall).

Також проведемо якісний аналіз результатів роботи розробленої моделі. Розглянемо юзера із таким id: AFZ54WN7RT3ZYJQV3CPA5AIUTACA.

На рисунках 3.2 – 3.3 показано покупки, які цей користувач вже здійснив у минулому.

```
all_beauty[all_beauty['user_id']=='AFZ54WN7RT3ZYJQV3CPA5AIUTACA']#.iloc[0]
```

	rating	title	text	images	asin	parent_asin	user_id	timestamp	helpful_vote	verified_purchase	date
415046	4.0	All in one	This product is an all in one, you wash and sh...		B0BQWTVXV2Q	B0BQWTVXV2Q	AFZ54WN7RT3ZYJQV3CPA5AIUTACA	1677891706547	0	False	2023-03-04 01:01:46.547
415047	4.0	Does the job	This scrub does the job well, it's saturated w...		B0BBH2RLDX	B0BBH2RLDX	AFZ54WN7RT3ZYJQV3CPA5AIUTACA	1673476540892	0	False	2023-01-11 22:35:40.892
415048	4.0	Jade stays cool even without ice	I like this roller, I was impressed when I tri...		B0BJJRV3M3X	B0BJJRV3M3X	AFZ54WN7RT3ZYJQV3CPA5AIUTACA	1673471484424	1	False	2023-01-11 21:11:24.424

Рисунок 3.2 – реальні покупки користувача

AFZ54WN7RT3ZYJQV3CPA5AIUTACA

```
meta_all_beauty[meta_all_beauty['parent_asin']=='B08QWTV2Q'].iloc[0, 1]
'Hanz de Fuko Everything Wash for Hair, Face & Body - All-in-One Soap & Shampoo for Men, Women & Kids, Safe for Sensitive Skin - Minty Rum & Sandalwood, 8 oz. Bottle'

meta_all_beauty[meta_all_beauty['parent_asin']=='B08BH2RLDX'].iloc[0, 1]
'Sia Botanics Revive Facial Scrub Gentle Antioxidant Scrub for Brighter Skin and Decongested pores - 3 Ounce'

meta_all_beauty[meta_all_beauty['parent_asin']=='B08J3RVW3X'].iloc[0, 1]
'2 In 1 Essential Oil Roller Bottles Jade Roller for Face Facial Massage Roller Skin Care Beauty Tool, Green'
```

Рисунок 3.3 – Назви продуктів, які вже придбав користувач AFZ54WN7RT3ZYJQV3CPA5AIUTACA

Як бачимо з історичних даних, вибраного користувача цікавлять продукти догляду за тілом та обличчям.

У ході роботи нашої системи, даному користувачу було запропоновано такі товари (рисунок 3.4):

```
Top-5 recommendations for user AFZ54WN7RT3ZYJQV3CPA5AIUTACA:
1. Pastel Cream Blush 41 | Pink | Fresh | Healthy & Natural Loo...
   Item ID: B08LYD2FMX
   Predicted Rating: 4.67

2. UCANBE Fruit Pie Filling 30 Shade Eyeshadow Palette...
   Item ID: B08CVW29PX
   Predicted Rating: 4.54

3. Saltair Body Wash (Exotic Pulp)...
   Item ID: B09V1TNP24
   Predicted Rating: 4.52

4. Automatic Curling Iron, Cordless Auto Hair Curler with 6 Tem...
   Item ID: B0B49VKP9J
   Predicted Rating: 4.47

5. Ice Roller For Face | Ice Roller, Face Roller, Ice Face Roll...
   Item ID: B0BRYZWF7R
   Predicted Rating: 4.45
```

Рисунок 3.4 – Рекомендації для користувача AFZ54WN7RT3ZYJQV3CPA5AIUTACA

Бачимо, що рекомендації також містять засоби для догляду за обличчям та тіло. Очевидно, що такі товари цікаві обраному користувачу. Тому модель добре підбирає схожі товари. Також варто помітити, що модель порекомендувала товар із розділу декоративної косметики (UCANBE Fruit Pie Filling 30 Shade Eyeshadow Palette). Людину, яка доглядає за собою та своїм зовнішнім виглядом високо ймовірно може цікавити і декоративна косметика. Але варто зауважити, що серед історичних покупок декоративної косметики

не було. Тому можемо зробити висновок, що наша модель не тільки підбирає рекомендації по схожим купленим товарам, а й може рекомендувати нові для користувача товари, але не менш цікаві і релевантні.

Розглянемо юзера із таким id: AFQVCWWD7BEUS6YTHTDLCTLWFIBA.

На рисунках 3.5 – 3.6 показано покупки, які цей користувач вже здійснив у минулому.

all_beauty[all_beauty['user_id']=='AFQVCWWD7BEUS6YTHTDLCTLWFIBA']												
	rating	title	text	images	asin	parent_asin		user_id	timestamp	helpful_vote	verified_purchase	date
113587	4.0	Heavy	Nice color and it lasts, but it will feel heav...	[]	B089189NC2	B0BVQQ58G8	AFQVCWWD7BEUS6YTHTDLCTLWFIBA	1678715373341		0	False	2023-03-13 13:49:33.341
113588	4.0	Lovely quality	I highly recommend for blow drying.	[]	B0B4VHNL46	B0B4VHNL46	AFQVCWWD7BEUS6YTHTDLCTLWFIBA	1678714187405		0	False	2023-03-13 13:29:47.405
113589	4.0	Lovely	I like this little mask a lot. Definitely exfo...	[]	B09PBTFFV4	B09PBTFFV4	AFQVCWWD7BEUS6YTHTDLCTLWFIBA	1678472479465		0	False	2023-03-10 18:21:19.465

Рисунок 3.5 – реальні покупки користувача
AFQVCWWD7BEUS6YTHTDLCTLWFIBA

[14]	meta_all_beauty[meta_all_beauty['parent_asin']=='B0BVQQ58G8'].iloc[0, 1]
	Beauty Bakerie Lip Whip Liquid Metallic Lipstick, Long Lasting Lip Color, Smudge Proof Makeup, Berried, 3.5 mL
[15]	meta_all_beauty[meta_all_beauty['parent_asin']=='B0B4VHNL46'].iloc[0, 1]
	L'ange Argan-Infused Round Brush Lightweight & Ergonomic Design 43MM Tourmaline Ceramic Barrel
[16]	meta_all_beauty[meta_all_beauty['parent_asin']=='B09PBTFFV4'].iloc[0, 1]
	La Maxime chemical free, natural, organic vegan best brightening hydrating facial exfoliating scrub & mask (2 in 1) pack for wrinkle repair & glowing skin for women men (2.47 ounce)

Рисунок 3.6 – Назви продуктів, які вже придбав користувач
AFQVCWWD7BEUS6YTHTDLCTLWFIBA

Як бачимо з історичних даних, даний користувач віддавав перевагу декоративній косметичці та догляду за шкірою. У ході роботи нашої системи, даному користувачу було запропоновано такі товари (рисунок 3.7):

- Top-5 recommendations for user AFQVCWWD7BEUS6YTHTDLCTLWFIBA:
1. Ice Roller For Face | Ice Roller, Face Roller, Ice Face Roll...
Item ID: B0BRYZW7R
Predicted Rating: 4.91
 2. THE ORDINARY Niacinamide 10% + Zinc 1% 30ml, 1.01 Fl Oz (Pac...
Item ID: B01MDTVZTZ
Predicted Rating: 4.90
 3. UCANBE Fruit Pie Filling 30 Shade Eyeshadow Palette...
Item ID: B08CVW29PX
Predicted Rating: 4.85
 4. TEK - Medium Rectangular Ash Wood Brush with Short Tooth Han...
Item ID: B004J4K6T8
Predicted Rating: 4.83
 5. [SKIN&LAB] Barrierderm Milky Serum | Infused with Milk Ceram...
Item ID: B0B6DPN1HQ
Predicted Rating: 4.81

Рисунок 3.7 – Рекомендації для користувача
AFQVCWWD7BEUS6YTHTDLCTLWFIBA

Рекомендації системи загалом добре відповідають профілю даного користувача, особливо в категорії догляду за шкірою обличчя - THE ORDINARY Niacinamide та SKIN&LAB сироватка логічно продовжують лінію інтересу до продуктів догляду за обличчям. Палетка тіней UCANBE є релевантна, оскільки користувач раніше купував декоративну косметику. Але бачимо, що система рекомендує продукти із іншої ніші також. Наприклад, Ice Roller для обличчя та дерев'яна щітка ТЕК. Ці продукти відносяться до догляду, але є досить специфічним. Проте для нас це є вказівкою на те, що система все таки підбирає різноманітні продукти.

Висновки до розділу 3

У рамках даного розділу була успішно розроблена та реалізована інноваційна система рекомендацій, що поєднує в собі переваги традиційних підходів із сучасними можливостями глибокого навчання. Головною метою було створення механізму, здатного не просто пропонувати товари на основі попередніх взаємодій, а й враховувати тонкі нюанси користувацьких вподобань та емоційного забарвлення відгуків. Для досягнення цієї мети було обрано алгоритм нейронної матричної факторизації, який дозволяє моделювати складні, нелінійні залежності між користувачами та товарами.

Цей вибір є суттєвим кроком вперед порівняно з лінійними моделями, адже він відкриває шлях до більш глибокого розуміння прихованих факторів, що впливають на рішення користувача.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі здійснюється функціонально-вартісний аналіз (ФВА) програмного продукту з метою оцінки його функціональних можливостей у поєднанні з витратами на розробку. Метод ФВА дозволяє дослідити співвідношення між цінністю функцій для кінцевого користувача та витратами на їх реалізацію, що, своєю чергою, сприяє вибору оптимального варіанту реалізації продукту як з технічної, так і з економічної точки зору.

ФВА охоплює визначення необхідних функцій програмного продукту, їхню класифікацію за впливом на загальну вартість, оцінку повних витрат, включаючи кількість робочих годин, а також ідентифікацію джерел витрат. Результати аналізу дають змогу виявити надлишкові або неефективні елементи, оптимізувати структуру продукту та знизити витрати без шкоди для його якості чи функціональності.

Таким чином, функціонально-вартісний аналіз виступає важливим інструментом для прийняття обґрунтованих рішень щодо доцільності та ефективності впровадження конкретних функцій у програмному продукті.

4.1 Постановка задачі проектування

Основною метою даної роботи є проектування архітектури гібридної рекомендаційної системи, яка поєднує декілька підходів до генерації персоналізованих рекомендацій. Оскільки вибір архітектурних рішень та реалізаційних компонентів безпосередньо впливає на ефективність і якість

функціонування всієї системи, особливу увагу було приділено аналізу функцій кожного окремого модуля. Таким чином, функціонально-вартісний аналіз спрямовано на дослідження доцільності впровадження конкретних функцій у загальну структуру системи.

У процесі проєктування було також сформовано перелік технічних вимог до програмного продукту, які повинні бути враховані для забезпечення його надійності та зручності у використанні. Основні технічні вимоги включають:

- коректне функціонування на сучасних персональних комп'ютерах із типовим програмним забезпеченням;
- зручний та інтуїтивно зрозумілий інтерфейс для кінцевого користувача;
- високу швидкість обробки запитів та генерації рекомендацій у реальному часі;
- можливість масштабування та подальшої підтримки системи;
- оптимізацію витрат на впровадження і супровід системи.

Для реалізації архітектури системи та її окремих модулів було використано мову програмування Python, яка забезпечує гнучкість, широку підтримку бібліотек для роботи з даними та машинного навчання, а також високу швидкість розробки.

4.2 Обґрунтування функцій програмного продукту

У даній роботі головною функцією є F_0 – розробка програмного продукту, який створює рекомендації для користувачів e-commerce проєкту. Таким чином, було виділено наступні основні функції програмного продукту:

F_1 – вибір мови програмування

F_2 – вибір середовища програмування

F_3 – вибір алгоритму для формування рекомендацій

Кожна з основних функції може мати декілька варіантів реалізації:

Функція F_1 :

- а) мова програмування Python;
- б) мова програмування R;

Функція F_2 :

- а) «Google Colab»;
- б) «PyCharm»;

Функція F_3 :

- а) алгоритм матричної факторизації;
- б) методи глибокого навчання (deep learning);

Варіанти реалізації основних функцій наведені на рис. 4.1.

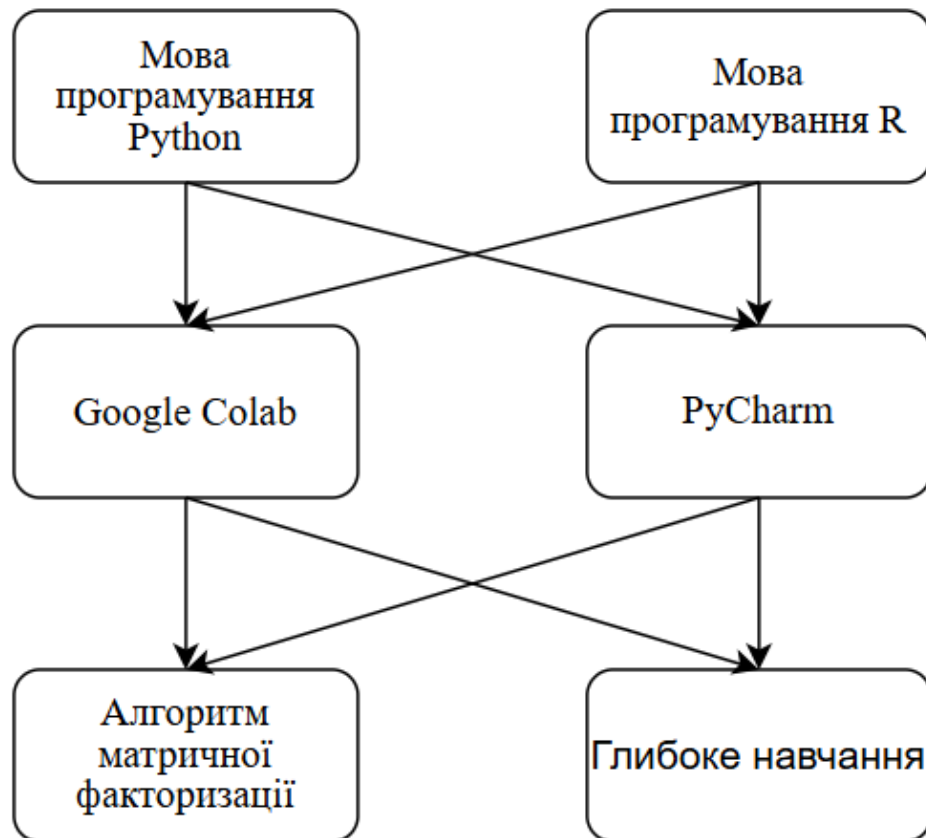


Рисунок 4.1 Морфологічна карта

Морфологічна карта відображає всі доступні комбінації варіантів функцій. На основі цієї карти було побудовано позитивно-негативну матрицю основних функцій, яка наведена в таблиці 4.1

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F ₁	А	Має широку екосистему бібліотек для машинного навчання та аналізу даних.	Менш ефективний при складній статистичній обробці
	Б	Чудово підходить для статистичного аналізу та візуалізації даних.	Має крутішу криву навчання та меншу гнучкість для загального програмування.
F ₂	А	Дозволяє безкоштовно запускати код на хмарних GPU/TPU без налаштувань.	Потребує постійного підключення до інтернету та має обмеження за ресурсами.
	Б	Потужне середовище з підтримкою автодоповнення, дебагінгу та інтеграції з Git.	Важке та ресурсомістке, особливо у безкоштовній версії (Community).

Продовження таблиці 4.1

F ₃	А	Ефективно виявляє приховані зв'язки між користувачами та об'єктами.	Погано працює з новими користувачами або товарами (проблема cold start).
	Б	Можуть враховувати складні нелінійні залежності та додаткові дані.	Потребують багато даних та обчислювальних ресурсів для навчання.

Після аналізу позитивно-негативної матриці можна зробити висновок, що при розробці програми деякі варіанти реалізації функції слід не розглядати, тому що вони не відповідають поставленим задачам до програмного продукту.

Функція F1:

Оскільки такі фактори, як: час та обсяг написаного програмного коду, наявність всіх потрібних бібліотек для розробки ПП є важливішими, ніж точність статистичних вимірювань, то варіант б) можна не розглядати.

Функція F2:

Так, як наведені варіанти реалізації мають свої значні переваги один над одним, варто розглянути та проаналізувати обидва.

Функція F3:

Оскільки зібрані нами дані є обмеженими, та не завжди коректними, то було прийнято рішення не розглядати варіант б).

Таким чином будемо використовувати наступні варіанти реалізації програмного продукту:

- 1) F1a–F2a–F3a
- 2) F1a–F2б–F3a

4.3 Обґрунтування системи параметрів програмного продукту

Щоб охарактеризувати програмний продукт, буде розглянуто наступні параметри:

- X_1 – швидкість виконання програмного коду
- X_2 – об’єм написаного коду
- X_3 – час, необхідний для обробки даних
- X_4 – об’єм оперативної пам’яті

Гірші, середні та кращі значення параметрів визначаються відповідно до вимог замовника та умов, що описують експлуатацію програмного продукту. Значення основних параметрів показано в таблиці 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			Гірші	Середні	Кращі
Швидкість виконання програмного коду	X_1	с	10	5	2
Об’єм написаного коду	X_2	кількість рядків коду	300	200	100
Час, необхідний для обробки даних	X_3	мс	1500	1000	500

Продовження таблиці 4.2

Об'єм оперативної пам'яті	X ₄	Мб	1024	512	256
---------------------------------	----------------	----	------	-----	-----

За даними таблиці 4.2 було побудовано графічні характеристики параметрів. Графіки наведено на рисунках 4.2 – 4.5.



Рисунок 4.2 – X₁, швидкість виконання коду



Рисунок 4.3 – X₂, об'єм написаного коду

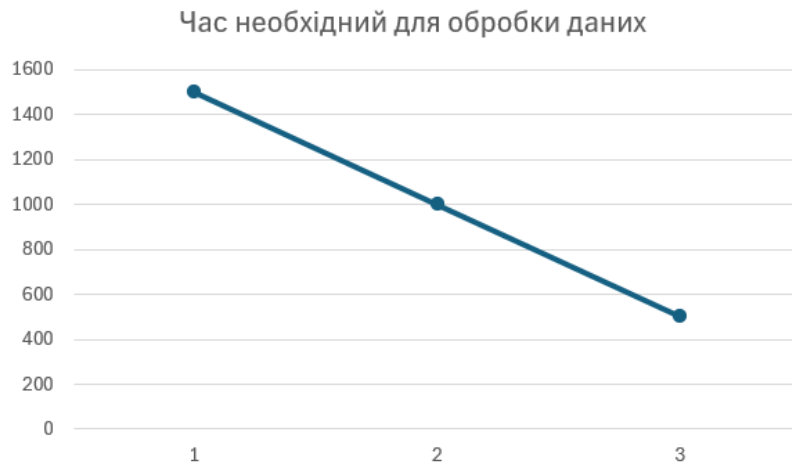


Рисунок 4.4 – X_3 , час, необхідний для обробки даних



Рисунок 4.4 – X_4 , об'єм оперативної пам'яті

4.4 Аналіз експертного оцінювання параметрів

Значущість кожного з параметрів визначається за допомогою методу попарного порівняння. Оцінювання здійснюється експертною групою, що складається з 7 учасників. Після ґрунтовного обговорення та аналізу кожен експерт визначає рівень важливості кожного параметра відповідно до

Для перевірки степені достовірності експертних оцінок, було визначено наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів

$$T = \frac{1}{n} R_{ij} = 17,5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів

$$\Delta_i = R_i - T$$

Сума відхилень по всіх параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилень

$$S = \sum_{i=1}^N \Delta_i^2 = 177$$

Таким чином, за таблицею 4.3 було обраховано коефіцієнт узгодженості

$$W = \frac{12 * S}{N^2(n^3 - n)} = \frac{12 * 179}{7^2(4^3 - 4)} \approx 0.72$$

Отриманий коефіцієнт є більшим за нормативний, який складає 0.67, тому ранжування можна вважати достовірним.

Скориставшись результатами ранжирування, було проведено попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	>	>	>	>	>	>	=	>	1.5
X_1 і X_3	<	>	<	<	>	>	=	<	0.5
X_1 і X_4	>	>	>	>	>	>	>	>	1.5
X_2 і X_3	<	<	<	<	<	<	=	<	0.5
X_2 і X_4	>	>	<	>	<	>	>	>	1.5
X_3 і X_4	>	>	>	>	>	>	>	>	1.5

Числове значення , що визначає ступінь переваги параметра над , визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок було побудовано матрицю $A = \|a\|$ (таблиця 4.5)

Для кожного параметра зробимо розрахунок вагомості K_{bi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i}$$

$$b_i = \sum_{j=1}^N a_{ij}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх. На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{bi} = \frac{b'_i}{\sum_{i=1}^n b'_i},$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Матриця А та розрахунок вагомості параметрів

Параметри X_i	Параметри X_i				Перший крок		Другий крок		Третій крок	
	X_1	X_2	X_3	X_4	b_i	K_{bi}	b_i	K_{bi}	b_i	K_{bi}
X_1	1	1,5	0,5	1,5	4,5	0,28	16,25	0,28	59,13	0,27
X_2	0,5	1	0,5	1,5	3,5	0,22	12,25	0,21	44,88	0,21
X_3	1,5	1,5	1	1,5	5,5	0,34	21,25	0,36	77,88	0,36
X_4	0,5	0,5	0,5	1	2,5	0,16	9,25	0,16	34,13	0,16
Всього:					16	1	59	1	216	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Було обрано абсолютні значення параметрів X_1 , X_2 , X_3 , X_4 , відповідно технічним умовам функціонування програмного продукту.

Використовуючи формулу, рахуємо показники якості ПП

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j}$$

де n – кількість параметрів;

K_{ei} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Значення розрахованих показників якості різних варіантів реалізації функції ПП занесено до таблиці 4.6.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Осно- вні функ- ції	Варіант реалізації функції	Парамет- ри	Абсолют- не значення парамет- ра	Бальна оцінка парамет- ра	Коефіці- єнт вагомості параметра	Коефіцієнт рівня якості
F ₁	А	X ₂	200	7	0.22	1.54
F ₂	А	X ₁	5	4	0.28	1.12
	Б	X ₄	512	5	0.16	0.8
F ₃	А	X ₃	1000	7	0.34	2.38

За даними з таблиці 4.6 та за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \dots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1.54 + 1.12 + 2.38 = 5.04$$

$$K_{K2} = 1.54 + 0.8 + 2.38 = 4.72$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки програмного продукту спочатку було вирішено провести розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи Б, завдання 2 до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_p \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М},$$

де T_p – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_p = 64$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{\Pi} = 1.021$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.6$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 64 \cdot 1.021 \cdot 1 \cdot 0.6 = 65.94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_p = 27$ людино-днів, $K_{\Pi} = 1.08$, $K_{СК} = 1$, $K_{СТ} = 0.7$:

$$T_2 = 27 \cdot 1.08 \cdot 1 \cdot 0.7 = 20,41 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_0 = (65.94 + 20.41) \cdot 8 = 690.8 \text{ людино-годин.}$$

В розробці беруть участь один програміст з окладом 36000 грн., один спеціаліст з медичним досвідом з окладом 28000. Визначимо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн.,}$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{36000 + 28000}{2 \cdot 18 \cdot 8} = 222,22 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{\text{ЗП}} = C_{\text{ч}} \cdot T_i \cdot K_{\text{Д}},$$

де $C_{\text{ч}}$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$K_{\text{Д}}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$C_{\text{ЗП}} = 222.22 \cdot 690.8 \cdot 1.125 = 172698,27 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0.22 = 172698,27 \cdot 0.22 = 37993,6 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (C_M)

Так як одна ЕОМ обслуговує одного програміста з окладом 36000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_G = 12 \cdot M \cdot K_3 = 12 \cdot 36000 \cdot 0,2 = 86400 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{\text{ЗП}} = C_G \cdot (1 + K_3) = 86400 \cdot (1 + 0.2) = 103680 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0.22 = 103680 \cdot 0,22 = 22809,6 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 41500 грн.

$$C_A = K_{\text{ТМ}} \cdot K_A \cdot C_{\text{ПР}} = 1.23 \cdot 0.25 \cdot 41500 = 12761,25 \text{ грн.,}$$

де $K_{\text{ТМ}}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$C_{\text{ПР}}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_R = K_{\text{ТМ}} \cdot C_{\text{ПР}} \cdot K_R = 1.23 \cdot 41500 \cdot 0.03 = 1531,35 \text{ грн.,}$$

де K_R – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{\text{ЕФ}} = (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 105 - 21 - 16) \cdot 8 \cdot 0.87 =$$

$$= 1552,08 \text{ години,}$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_P – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_3 \cdot C_{\text{ЕН}} = 1552,08 \cdot 0,3 \cdot 0,5 \cdot 9,43 = 4390,83 \text{ грн.},$$

де $N_{\text{С}}$ – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$C_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_{\text{Н}} = C_{\text{ПР}} \cdot 0,67 = 20928,45 \cdot 0,67 = 14022,06 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}},$$

$$C_{\text{ЕКС}} = 172698,27 + 22809,6 + 12761,25 + 1531,35 + 4390,83 + 14022,06 = 228213,36 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 228213,36 / 1552,08 = 147,03 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_{\text{М}} = C_{\text{М-Г}} \cdot T,$$

$$C_{\text{М}} = 147,03 \cdot 690,8 = 101867,3 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0,67,$$

$$\text{І. } C_{\text{Н}} = 172698,27 \cdot 0,67 = 115707,84 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}},$$

$$\text{І. } C_{\text{ПП}} = 172698,27 + 22809,6 + 101867,3 + 115707,84 = 413083,01 \text{ грн.}$$

4.7 Вибір кращого варіанту ІІІ техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{Kj} / C_{\Phi j},$$

$$K_{\text{ТЕР}1} = 19,5 / 414697,55 = 4,7 \cdot 10^{-5},$$

За результатами аналізу вартості розробки програмного продукту було визначено трудомісткість для двох основних етапів: створення проєкту програмного забезпечення та розробки його програмної оболонки. Для першого етапу трудомісткість становить 65,94 людино-днів, що враховує використання довідкових матеріалів та стандартних модулів, з поправочним коефіцієнтом 1,021 і коефіцієнтом стандартних модулів 0,6. Другий етап передбачає застосування алгоритмів третьої групи складності з коефіцієнтом 1,08 і має трудомісткість 20,41 людино-днів. Загальна трудомісткість обох завдань у перерахунку на людино-години становить 690,8, що дозволяє оцінити необхідні ресурси та строки реалізації проєкту. Загальні витрати на розробку програмного продукту складають 41 469 755 гривень, включаючи заробітну плату, витрати на машинний час, соціальні внески та накладні витрати. Коефіцієнт техніко-економічного рівня ($4,7 \times 10^{-5}$) свідчить про високу вартість проєкту відносно його технічних характеристик і ринкової цінності, що потребує додаткового аналізу ефективності інвестицій і оптимізації витрат.

Висновки до розділу 4

У цьому розділі здійснено всебічний функціонально-вартісний аналіз програмного забезпечення. Проведено оцінку ключових функцій продукту та

сформульовано висновки щодо доцільності використаних технологій для його реалізації.

Під час аналізу були визначені та оцінені основні функціональні можливості програмного продукту, а також встановлено його характерні параметри. На основі отриманих результатів обрано оптимальний варіант реалізації програмного забезпечення.

ВИСНОВКИ

У рамках даної дипломної роботи було проведено глибоке дослідження та розроблено рекомендаційну систему, спрямовану на оптимізацію роботи e-commerce проектів. Ключовим результатом є створення функціональної моделі, здатної ефективно пропонувати користувачам товари, що відповідають їхнім інтересам та вподобанням. Система ґрунтується на сучасному підході до аналізу даних про поведінку користувачів, враховуючи їхні перегляди, покупки, оцінки та інші взаємодії з продуктами. Особливу увагу було приділено розробці та впровадженню алгоритмів, що забезпечують високу точність прогнозування та релевантність рекомендацій. Це досягається завдяки використанню методів машинного навчання, зокрема алгоритмів колаборативної фільтрації, які дозволяють ідентифікувати схожість між користувачами та продуктами.

Метою роботи було створення ефективної рекомендаційної системи для e-commerce проектів, яка б сприяла підвищенню конверсії та задоволеності клієнтів. Ця мета була успішно досягнута завдяки послідовному вирішенню поставлених завдань. По-перше, було проведено детальний аналіз існуючих підходів до рекомендаційних систем, що дозволило обрати найбільш оптимальні методи для реалізації. По-друге, було розроблено архітектуру системи, що включає модулі збору та обробки даних, формування рекомендацій та їх подальшого представлення користувачеві. По-третє, була реалізована програмна частина системи, що пройшла етап тестування та демонструє високу працездатність. Зокрема, було успішно впроваджено функціонал прогнозування рейтингу товарів та формування персоналізованих рекомендацій для окремого користувача.

Практична цінність даної дипломної роботи є значною і багатогранною. Створена рекомендаційна система може бути безпосередньо впроваджена в

існуючі e-commerce платформи, що дозволить їм значно покращити користувацький досвід та підвищити комерційні показники. Завдяки персоналізованим рекомендаціям, користувачі отримуватимуть пропозиції, що дійсно їх цікавлять, що, в свою чергу, призведе до збільшення кількості покупок та середнього чека. Крім того, система може допомогти в оптимізації маркетингових стратегій, надаючи цінні інсайти про переваги клієнтів. Результати дослідження також можуть бути використані для подальшого розвитку та вдосконалення рекомендаційних технологій у галузі електронної комерції, слугуючи основою для майбутніх наукових та практичних розробок. Таким чином, дана робота не тільки вирішує актуальну проблему бізнесу, а й робить внесок у розвиток сучасної інформаційної технології.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zhenhua Dong, Zhe Wang, Jun Xu, Ruiming Tang, Jirong Wen. A Brief History of Recommender Systems. In Proceedings of ACM Conference(Conference'17). ACM, New York, NY, USA. 2022. 9 pa.
2. McKinsey. How retailers can keep up with consumers. URL: <https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers>
3. Jiangwei Pan, Gary Tang, Henry Wang, and Justin Basilico. Recommending for Long-Term Member Satisfaction at Netflix, 2024. URL: <https://netflixtechblog.com/recommending-for-long-term-member-satisfaction-at-netflix-ac15cada49ef>
4. Dmytro Poluriezov. Від концепції до реалізації: як побудувати ефективну рекомендаційну систему на основі машинного навчання за два дні, 2024. URL: <https://dou.ua/forums/topic/47504/>
5. Shah Khusro, Zafar Ali and Irfan Ullah. Recommender Systems: Challenges, Frontiers and Research Opportunities. Information Science and Applications. 2016, P. 1179-1189. URL: https://www.researchgate.net/publication/294860665_Recommender_Systems_Issues_Challenges_and_Research_Opportunities
6. Чередніченко О.Ю., Янголенко О.В., Іващенко О.В., Матвєєв О.М. Моделі формування рекомендацій у інтелектуальних системах електронної комерції. Системи обробки інформації. 2020. № 1(160). С. 32-39. URL: <https://doi.org/10.30748/soi.2020.160.04>.
7. Wang S., Qiu J. A deep neural network model for fashion collocation recommendation using side information in e-commerce, 2021. Vol. 110, Article. 107753. URL: <https://www.sciencedirect.com/science/article/abs/pii/S1568494621006748?via%3Dihub>

8. Sadq Z. M., Sabir H. N. Saeed V. S. H. Analyzing the Amazon success strategies. *Journal of Process Management and New Technologies*. 2018, 6(4), P. 65–69.
9. Cherry Adam. How ASOS utilises AI to suggest your next purchase, 2020. URL: <https://recommend.pro/how-asos-utilises-ai-to-suggest-your-next-purchase/>
10. ByBernard Marr, The Amazing Ways Spotify Uses Big Data, AI And Machine Learning To Drive Business Success, 2017. URL: <https://www.forbes.com/sites/bernardmarr/2017/10/30/the-amazing-ways-spotify-uses-big-data-ai-and-machine-learning-to-drive-business-success/>
11. Spotify, Safety and Privacy center. Understanding recommendations on Spotify. URL: <https://www.spotify.com/ua-en/safetyandprivacy/understanding-recommen>
12. Lucas Bernardi, Jaap Kamps, Julia Kiseleva, Melanie JI Müller. The Continuous Cold Start Problem in e-Commerce Recommender Systems, 2015. URL: <https://arxiv.org/abs/1508.01177>
13. Wang K., Zhang T., Xue T., Lu Y., Na S.G. E-commerce personalized recommendation analysis by deeply-learned clustering. *Journal of Visual Communication and Image Representation*, 2020. Vol. 71, Article 102735. URL: <https://doi.org/10.1016/j.jvcir.2019.102735>
14. Qazi Mohammad Areeb, Mohammad Nadeem, Shahab Saquib Sohail, Raza Imam, Faiyaz Doctor, Yassine Himeur, Amir Hussain, Abbas Amira. Filter Bubbles in Recommender Systems: Fact or Fallacy – A Systematic Review, 2023. URL: <https://arxiv.org/abs/2307.01221>
15. Forbes. Personalization Vs. Privacy: Marketing's Ultimate Heavyweight Match, 2022. URL: <https://www.forbes.com/councils/forbesbusinesscouncil/2022/12/28/personalization-vs-privacy-marketings-ultimate-heavyweight-match/>
16. Amazon Reviews 2023 Dataset, 2023. URL: <https://amazon-reviews-2023.github.io/>

17. Aggarwal Charu C. Recommender Systems: The Textbook. Cham, Springer, 2016. 498 p.
18. Stratascratch. Step-by-Step Guide to Building Content-Based Filtering, 2023. URL: <https://www.stratascratch.com/blog/step-by-step-guide-to-building-content-based-filtering/>
19. Analytics Vidhya. Beginners Guide to Content Based Recommender Systems, 2024. URL: <https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/#h-case-study-1-how-to-calculate-tf-idf>
20. Similarity Measures used in Recommender Systems: A Study URL: <https://www.semanticscholar.org/paper/Similarity-Measures-used-in-Recommender-Systems-%3A-A-Agarwal-Chauhan/943ae455fafc3d36ae4ce68f1a60ae4f85623e2a?p2df>
21. Analytics Vidhya. Introduction to Collaborative Filtering, 2024. URL: <https://www.analyticsvidhya.com/blog/2022/02/introduction-to-collaborative-filtering/>
22. Google Developers – Recommender Systems Introduction. URL: <https://developers.google.com/machine-learning/recommendation>
23. Bo Pang and Lillian Lee. Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval, 2008. Vol. 2, No. 1–2, P. 1–135.
24. Jurafsky D., Martin J. H. Speech and Language Processing (3rd Edition, Draft). Pearson, 2020. URL: <https://web.stanford.edu/~jurafsky/slp3/>
25. RoBERTa model documentation. URL: https://huggingface.co/docs/transformers/model_doc/roberta

ДОДАТОК А ПРОГРАМНИЙ КОД

```

import pandas as pd
import numpy as np
import json
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import matplotlib.pyplot as plt
import warnings

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, accuracy_score, precision_score,
recall_score, f1_score
from transformers import pipeline

warnings.filterwarnings('ignore')
from google.colab import drive

drive.mount('/content/drive')

class EnhancedRecommendationSystem:
    """
    Enhanced recommendation system with improved matrix factorization and sentiment analysis
    """

    def __init__(self, n_factors=100, dropout_rate=0.3, use_deep_model=True):
        self.n_factors = n_factors
        self.dropout_rate = dropout_rate
        self.use_deep_model = use_deep_model
        self.user_encoder = LabelEncoder()

```

```

self.item_encoder = LabelEncoder()
self.rating_scaler = StandardScaler()
self.model = None
self.device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
self.item_titles = {}
self.user_stats = {}
self.item_stats = {}

print(f'Using device: {self.device}')

try:
    self.sentiment_analyzer = pipeline(
        "sentiment-analysis",
        model="cardiffnlp/twitter-roberta-base-sentiment-latest",
        device=0 if torch.cuda.is_available() else -1
    )
except:
    print("Using basic sentiment analyzer")
    self.sentiment_analyzer = pipeline("sentiment-analysis")

def get_sentiment_score(self, text):
    """Get sentiment score from text"""
    if not text or pd.isna(text):
        return 0.5

    try:
        result = self.sentiment_analyzer(str(text)[:512])
        if isinstance(result, list):
            result = result[0]

        if result['label'] in ['POSITIVE', 'LABEL_2']:
            return result['score']

```

```

        elif result['label'] in ['NEGATIVE', 'LABEL_0']:
            return 1 - result['score']
        else:
            return 0.5
    except:
        return 0.5

def prepare_data(self, reviews_df, metadata_df):
    """Enhanced data preparation with filtering and feature engineering"""
    print("Preparing data...")

    data = reviews_df.merge(metadata_df, on='parent_asin', how='inner')

    data = data.dropna(subset=['user_id', 'parent_asin', 'rating'])
    data['text'] = data['text'].fillna("")
    data['title_x'] = data['title_x'].fillna("")

    print("Filtering sparse users and items...")
    user_counts = data['user_id'].value_counts()
    item_counts = data['parent_asin'].value_counts()

    min_user_reviews = 3
    min_item_reviews = 3

    valid_users = user_counts[user_counts >= min_user_reviews].index
    valid_items = item_counts[item_counts >= min_item_reviews].index

    data = data[data['user_id'].isin(valid_users) & data['parent_asin'].isin(valid_items)]

    print(f'After filtering: {len(data)} reviews, {data["user_id"].nunique()} users,
    {data["parent_asin"].nunique()} items')

    for _, row in data.iterrows():

```

```

if row['parent_asin'] not in self.item_titles:
    title = row.get('title_y', "") or row.get('title_x', "") or row['parent_asin']
    self.item_titles[row['parent_asin']] = title

print("Calculating user and item statistics...")

user_stats = data.groupby('user_id')['rating'].agg(['mean', 'std', 'count']).reset_index()
user_stats['std'] = user_stats['std'].fillna(0)
self.user_stats = dict(zip(user_stats['user_id'],
                           zip(user_stats['mean'], user_stats['std'], user_stats['count']))))

item_stats = data.groupby('parent_asin')['rating'].agg(['mean', 'std', 'count']).reset_index()
item_stats['std'] = item_stats['std'].fillna(0)
self.item_stats = dict(zip(item_stats['parent_asin'],
                           zip(item_stats['mean'], item_stats['std'], item_stats['count']))))

data['user_avg_rating'] = data['user_id'].map(lambda x: self.user_stats[x][0])
data['item_avg_rating'] = data['parent_asin'].map(lambda x: self.item_stats[x][0])
data['user_rating_count'] = data['user_id'].map(lambda x: self.user_stats[x][2])
data['item_rating_count'] = data['parent_asin'].map(lambda x: self.item_stats[x][2])

data['review_text'] = data['title_x'] + ' ' + data['text']

# Add sentiment score
print("Analyzing review sentiment...")
if len(data) > 10000:
    sample_data = data.sample(n=10000, random_state=42)
    sample_sentiments = sample_data['review_text'].apply(self.get_sentiment_score)
    avg_sentiment = sample_sentiments.mean()
    data['sentiment'] = data['review_text'].apply(
        lambda x: self.get_sentiment_score(x) if len(str(x)) > 20 else avg_sentiment
    )
else:

```

```

data['sentiment'] = data['review_text'].apply(self.get_sentiment_score)

data['rating_normalized'] = self.rating_scaler.fit_transform(data[['rating']])[ :, 0]

return data

def create_enhanced_model(self, n_users, n_items):
    """Create enhanced model with deep architecture"""

    if self.use_deep_model:
        class DeepMatrixFactorization(nn.Module):
            def __init__(self, n_users, n_items, n_factors, dropout_rate):
                super().__init__()

                self.user_embedding = nn.Embedding(n_users, n_factors)
                self.item_embedding = nn.Embedding(n_items, n_factors)
                self.user_bias = nn.Embedding(n_users, 1)
                self.item_bias = nn.Embedding(n_items, 1)
                self.global_bias = nn.Parameter(torch.tensor(0.0))

                self.fc1 = nn.Linear(n_factors * 2 + 1, n_factors) # +1 for sentiment
                self.fc2 = nn.Linear(n_factors, n_factors // 2)
                self.fc3 = nn.Linear(n_factors // 2, n_factors // 4)
                self.output = nn.Linear(n_factors // 4, 1)

                self.dropout = nn.Dropout(dropout_rate)
                self.bn1 = nn.BatchNorm1d(n_factors)
                self.bn2 = nn.BatchNorm1d(n_factors // 2)
                self.bn3 = nn.BatchNorm1d(n_factors // 4)

                nn.init.normal_(self.user_embedding.weight, std=0.05)
                nn.init.normal_(self.item_embedding.weight, std=0.05)

```

```

nn.init.zeros_(self.user_bias.weight)
nn.init.zeros_(self.item_bias.weight)

for layer in [self.fc1, self.fc2, self.fc3, self.output]:
    nn.init.xavier_uniform_(layer.weight)
    nn.init.zeros_(layer.bias)

def forward(self, user_ids, item_ids, sentiments=None):
    user_emb = self.user_embedding(user_ids)
    item_emb = self.item_embedding(item_ids)

    if sentiments is not None:
        features = torch.cat([user_emb, item_emb, sentiments.unsqueeze(1)], dim=1)
    else:
        features = torch.cat([user_emb, item_emb, torch.zeros(user_emb.size(0), 1,
device=user_emb.device)], dim=1)

    x = F.relu(self.bn1(self.fc1(features)))
    x = self.dropout(x)
    x = F.relu(self.bn2(self.fc2(x)))
    x = self.dropout(x)
    x = F.relu(self.bn3(self.fc3(x)))
    x = self.dropout(x)
    output = self.output(x).squeeze()

    user_bias = self.user_bias(user_ids).squeeze()
    item_bias = self.item_bias(item_ids).squeeze()

    result = output + user_bias + item_bias + self.global_bias

    return result

```

```
return DeepMatrixFactorization(n_users, n_items, self.n_factors, self.dropout_rate)
```

```
else:
```

```
# Simple model with improvements
```

```
class ImprovedMatrixFactorization(nn.Module):
```

```
    def __init__(self, n_users, n_items, n_factors):
```

```
        super().__init__()
```

```
        self.user_embedding = nn.Embedding(n_users, n_factors)
```

```
        self.item_embedding = nn.Embedding(n_items, n_factors)
```

```
        self.user_bias = nn.Embedding(n_users, 1)
```

```
        self.item_bias = nn.Embedding(n_items, 1)
```

```
        self.global_bias = nn.Parameter(torch.tensor(0.0))
```

```
        nn.init.normal_(self.user_embedding.weight, std=0.05)
```

```
        nn.init.normal_(self.item_embedding.weight, std=0.05)
```

```
        nn.init.zeros_(self.user_bias.weight)
```

```
        nn.init.zeros_(self.item_bias.weight)
```

```
    def forward(self, user_ids, item_ids, sentiments=None):
```

```
        user_emb = self.user_embedding(user_ids)
```

```
        item_emb = self.item_embedding(item_ids)
```

```
        interaction = (user_emb * item_emb).sum(dim=1)
```

```
        user_bias = self.user_bias(user_ids).squeeze()
```

```
        item_bias = self.item_bias(item_ids).squeeze()
```

```
        result = interaction + user_bias + item_bias + self.global_bias
```

```
        if sentiments is not None:
```

```
            result = result + sentiments * 0.3
```

```

        return result

    return ImprovedMatrixFactorization(n_users, n_items, self.n_factors)

def evaluate_model(self, test_data):
    """Evaluate model performance"""
    print("\nEvaluating model performance...")

    user_ids = torch.LongTensor(test_data['user_encoded'].values).to(self.device)
    item_ids = torch.LongTensor(test_data['item_encoded'].values).to(self.device)
    true_ratings = test_data['rating'].values
    sentiments = torch.FloatTensor(test_data['sentiment'].values).to(self.device)

    self.model.eval()
    with torch.no_grad():
        predictions = self.model(user_ids, item_ids, sentiments).cpu().numpy()

        predictions_denorm = self.rating_scaler.inverse_transform(predictions.reshape(-1, 1))[:, 0]
        predictions_denorm = np.clip(predictions_denorm, 1.0, 5.0)

    rmse = np.sqrt(mean_squared_error(true_ratings, predictions_denorm))
    mae = mean_absolute_error(true_ratings, predictions_denorm)

    true_binary = (true_ratings >= 4).astype(int)
    pred_binary = (predictions_denorm >= 4).astype(int)

    accuracy = accuracy_score(true_binary, pred_binary)
    precision = precision_score(true_binary, pred_binary, zero_division=0)
    recall = recall_score(true_binary, pred_binary, zero_division=0)
    f1 = f1_score(true_binary, pred_binary, zero_division=0)

    metrics = {

```

```

    'RMSE': rmse,
    'MAE': mae,
    'Accuracy': accuracy,
    'Precision': precision,
    'Recall': recall,
    'F1_Score': f1
}

```

```

print(f'Model Performance Metrics:')
print(f'RMSE: {rmse:.4f}')
print(f'MAE: {mae:.4f}')
print(f'Accuracy: {accuracy:.4f}')
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')

```

```

return metrics

```

```

def fit(self, reviews_df, metadata_df, epochs=100, batch_size=512, lr=0.01, weight_decay=0.01):

```

```

    """Train the enhanced model"""

```

```

    print("Training enhanced recommendation model...")

```

```

    data = self.prepare_data(reviews_df, metadata_df)

```

```

    data['user_encoded'] = self.user_encoder.fit_transform(data['user_id'])

```

```

    data['item_encoded'] = self.item_encoder.fit_transform(data['parent_asin'])

```

```

    user_ids = torch.LongTensor(data['user_encoded'].values)

```

```

    item_ids = torch.LongTensor(data['item_encoded'].values)

```

```

    ratings = torch.FloatTensor(data['rating_normalized'].values)

```

```

    sentiments = torch.FloatTensor(data['sentiment'].values)

```

```

    train_idx, val_idx = train_test_split(range(len(data)), test_size=0.2, random_state=42)

```

```

n_users = len(self.user_encoder.classes_)
n_items = len(self.item_encoder.classes_)
self.model = self.create_enhanced_model(n_users, n_items).to(self.device)

optimizer = optim.AdamW(self.model.parameters(), lr=lr, weight_decay=weight_decay)

scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=10,
factor=0.5)

criterion = nn.MSELoss()

train_losses = []
val_losses = []
best_val_loss = float('inf')
patience_counter = 0
patience = 15

print(f'Training for {epochs} epochs...')
for epoch in range(epochs):
    self.model.train()
    train_loss = 0
    train_batches = 0

    train_indices = np.random.permutation(train_idx)

    for i in range(0, len(train_indices), batch_size):
        batch_idx = train_indices[i:i + batch_size]

        batch_users = user_ids[batch_idx].to(self.device)
        batch_items = item_ids[batch_idx].to(self.device)
        batch_ratings = ratings[batch_idx].to(self.device)
        batch_sentiments = sentiments[batch_idx].to(self.device)

```

```

optimizer.zero_grad()

predictions = self.model(batch_users, batch_items, batch_sentiments)
loss = criterion(predictions, batch_ratings)
loss.backward()

# Gradient clipping
torch.nn.utils.clip_grad_norm_(self.model.parameters(), max_norm=1.0)

optimizer.step()

train_loss += loss.item()
train_batches += 1

# Validation
self.model.eval()
val_loss = 0
val_batches = 0

with torch.no_grad():
    for i in range(0, len(val_idx), batch_size):
        batch_idx = val_idx[i:i + batch_size]

        batch_users = user_ids[batch_idx].to(self.device)
        batch_items = item_ids[batch_idx].to(self.device)
        batch_ratings = ratings[batch_idx].to(self.device)
        batch_sentiments = sentiments[batch_idx].to(self.device)

        predictions = self.model(batch_users, batch_items, batch_sentiments)
        loss = criterion(predictions, batch_ratings)
        val_loss += loss.item()
        val_batches += 1

```

```

avg_train_loss = train_loss / train_batches
avg_val_loss = val_loss / val_batches

train_losses.append(avg_train_loss)
val_losses.append(avg_val_loss)

scheduler.step(avg_val_loss)

# Early stopping
if avg_val_loss < best_val_loss:
    best_val_loss = avg_val_loss
    patience_counter = 0
    # Save best model
    torch.save(self.model.state_dict(), 'best_model.pth')
else:
    patience_counter += 1

if epoch % 10 == 0:
    print(f'Epoch {epoch}: Train Loss: {avg_train_loss:.4f}, Val Loss: {avg_val_loss:.4f}, LR:
{optimizer.param_groups[0]["lr"]:.6f}')

# Early stopping
if patience_counter >= patience:
    print(f'Early stopping at epoch {epoch}')
    break

# Load best model
self.model.load_state_dict(torch.load('best_model.pth'))
self.data = data

# Evaluate model

```

```
val_data = data.iloc[val_idx].copy()
metrics = self.evaluate_model(val_data)
```

```
return train_losses, val_losses, metrics
```

```
def predict(self, user_id, item_id, review_text=""):
```

```
    """Predict rating"""
```

```
    if self.model is None:
```

```
        raise ValueError("Model is not trained!")
```

```
    try:
```

```
        user_encoded = self.user_encoder.transform([user_id])[0]
```

```
        item_encoded = self.item_encoder.transform([item_id])[0]
```

```
        sentiment = self.get_sentiment_score(review_text) if review_text else 0.5
```

```
        user_tensor = torch.LongTensor([user_encoded]).to(self.device)
```

```
        item_tensor = torch.LongTensor([item_encoded]).to(self.device)
```

```
        sentiment_tensor = torch.FloatTensor([sentiment]).to(self.device)
```

```
        self.model.eval()
```

```
        with torch.no_grad():
```

```
            prediction = self.model(user_tensor, item_tensor, sentiment_tensor)
```

```
            # Denormalize prediction
```

```
            prediction_denorm = self.rating_scaler.inverse_transform([[prediction.cpu().item()]])[0][0]
```

```
        return max(1.0, min(5.0, prediction_denorm))
```

```
except ValueError:
```

```
    # Fallback to user or item average
```

```
    if user_id in self.user_stats:
```

```

        return self.user_stats[user_id][0]
    elif item_id in self.item_stats:
        return self.item_stats[item_id][0]
    else:
        return 3.0

def recommend_items(self, user_id, n_recommendations=10):
    """Recommend items for user"""
    if self.model is None:
        raise ValueError("Model is not trained!")

    try:
        user_encoded = self.user_encoder.transform([user_id])[0]
    except ValueError:
        print(f"User {user_id} not found")
        return []

    all_items = self.item_encoder.classes_

    user_items = set(self.data[self.data['user_id'] == user_id]['parent_asin'])
    candidate_items = [item for item in all_items if item not in user_items]

    if not candidate_items:
        return []

    predictions = []
    batch_size = 1000

    self.model.eval()
    with torch.no_grad():
        for i in range(0, len(candidate_items), batch_size):
            batch_items = candidate_items[i:i + batch_size]

            try:

```

```

item_encoded_batch = [self.item_encoder.transform([item])[0] for item in batch_items]

user_tensor = torch.LongTensor([user_encoded] * len(batch_items)).to(self.device)
item_tensor = torch.LongTensor(item_encoded_batch).to(self.device)

batch_predictions = self.model(user_tensor, item_tensor)

batch_predictions_denorm = self.rating_scaler.inverse_transform(
    batch_predictions.cpu().numpy().reshape(-1, 1)
)[: , 0]

for j, item in enumerate(batch_items):
    item_title = self.item_titles.get(item, item)
    predictions.append({
        'item_id': item,
        'item_title': item_title,
        'predicted_rating': batch_predictions_denorm[j]
    })
except:
    continue

predictions.sort(key=lambda x: x['predicted_rating'], reverse=True)
return predictions[:n_recommendations]

def load_data(reviews_file, metadata_file):
    """Load data"""
    print("Loading data...")

    reviews = []
    with open(reviews_file, 'r', encoding='utf-8') as f:
        for line in f:
            try:

```

```

        reviews.append(json.loads(line.strip()))
    except:
        continue
reviews_df = pd.DataFrame(reviews)

if 'timestamp' in reviews_df.columns:
    reviews_df['date'] = pd.to_datetime(reviews_df['timestamp'], unit='ms')
    reviews_df = reviews_df[reviews_df['date'] > '2022-12-31']

metadata = []
with open(metadata_file, 'r', encoding='utf-8') as f:
    for line in f:
        try:
            metadata.append(json.loads(line.strip()))
        except:
            continue
metadata_df = pd.DataFrame(metadata)

return reviews_df, metadata_df

def main():
    print("Enhanced Recommendation System")
    print("=" * 50)

    reviews_file = "/content/drive/MyDrive/ДИПЛОМ/All_Beauty.jsonl"
    metadata_file = "/content/drive/MyDrive/ДИПЛОМ/meta_All_Beauty.jsonl"

    try:
        reviews_df, metadata_df = load_data(reviews_file, metadata_file)
        print(f"Loaded {len(reviews_df)} reviews and {len(metadata_df)} items")

        recommender = EnhancedRecommendationSystem(
            n_factors=128,

```

```

        dropout_rate=0.3,
        use_deep_model=True
    )
train_losses, val_losses, metrics = recommender.fit(
    reviews_df, metadata_df,
    epochs=100,
    batch_size=512,
    lr=0.01,
    weight_decay=0.01
)

print("\nTraining completed!")
print("\nSelecting user for testing...")

positive_users = reviews_df.groupby('user_id')['rating'].agg(['min', 'count']).reset_index()
positive_users = positive_users[
    (positive_users['min'] > 3.5) &
    (positive_users['count'] >= 3)
]

if len(positive_users) == 0:
    print("No users with only positive ratings found, using fallback selection...")
    user_avg_ratings = reviews_df.groupby('user_id')['rating'].agg(['mean', 'count']).reset_index()
    user_avg_ratings = user_avg_ratings[
        (user_avg_ratings['mean'] > 4.0) &
    ]
    if len(user_avg_ratings) > 0:
        sample_user = user_avg_ratings.sample(1, random_state=42)['user_id'].iloc[0]
    else:
        sample_user = reviews_df['user_id'].iloc[0]
else:

```

```

    sample_user = positive_users.sample(1, random_state=42)['user_id'].iloc[0]
    user_ratings = reviews_df[reviews_df['user_id'] == sample_user]['rating']
    print(f"Selected user: {sample_user}")
    print(f"User's rating statistics:")
    print(f" - Number of ratings: {len(user_ratings)}")
    print(f" - Min rating: {user_ratings.min()}")
    print(f" - Max rating: {user_ratings.max()}")
    print(f" - Mean rating: {user_ratings.mean():.2f}")

    sample_item = reviews_df['parent_asin'].iloc[0]

    prediction = recommender.predict(sample_user, sample_item, "I love this product!")
    print(f"\nRating prediction: {prediction:.2f}")

    recommendations = recommender.recommend_items(sample_user, 5)
    print(f"\nTop-5 recommendations for user {sample_user}:")
    for i, rec in enumerate(recommendations, 1):
        print(f'{i}. {rec["item_title"][:60]}...')
        print(f"  Item ID: {rec['item_id']}")
        print(f"  Predicted Rating: {rec['predicted_rating']:.2f}")
        print()
    print("\n" + "="*60)
    print("ENHANCED MODEL PERFORMANCE SUMMARY")
    print("="*60)
    print(f"RMSE: {metrics['RMSE']:.4f} (Lower is better)")
    print(f"MAE: {metrics['MAE']:.4f} (Lower is better)")
    print(f"Accuracy: {metrics['Accuracy']:.4f}")
    print(f"Precision: {metrics['Precision']:.4f}")
    print(f"Recall: {metrics['Recall']:.4f}")
    print(f"F1 Score: {metrics['F1_Score']:.4f}")
except Exception as e:
    print(f"Error: {e}")

```

```
import traceback

traceback.print_exc()

if __name__ == "__main__":
    main()
```

ДОДАТОК Б ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДО ДОПОВІДІ

Дипломна робота
на здобуття ступеня бакалавра на тему
“Рекомендаційна система для e-commerce
проектів”

Науковий керівник:
доцент, PhD, Гуськова
Віра Геннадіївна

Виконала:
студентка групи КА-12
Фисак Софія
Романівна

2

Об’єкт дослідження – користувацькі оцінки та відгуки на товари з e-commerce платформи Amazon.

Предмет дослідження – алгоритми та методи побудови рекомендаційних систем, зокрема ті, що поєднують колаборативну фільтрацію та емоційний аналіз тексту.

Мета роботи – створити гібридну рекомендаційну систему для e-commerce, яка поєднає матричну факторизацію з емоційним аналізом відгуків користувачів, для надання персоналізованих та емоційно релевантних товарних рекомендацій.

Актуальність теми

У сучасному світі онлайн-магазини вже неможливо уявити без персоналізованих рекомендацій, які стали невіддільною частиною нашого цифрового досвіду. Рекомендаційні системи слугують важливим містком між користувачем і величезною кількістю доступних продуктів, допомагаючи економити час та робити вибір точнішим і комфортнішим. Для e-commerce проектів ці системи є ключовим інструментом, що безпосередньо впливає на конверсії, прибуток і довіру клієнтів. Кожна успішна рекомендація є результатом складних алгоритмів, аналізу поведінкових даних та виявлення прихованих шаблонів, що лежать в основі конкурентоспроможності сучасних платформ. Таким чином, рекомендаційні системи є яскравим прикладом того, як штучний інтелект трансформує нашу повсякденну взаємодію з цифровим простором.

Визначення та еволюція

Рекомендаційні системи (РС) — це програмні інструменти та алгоритмічні рішення, призначені для надання персоналізованих пропозицій користувачам на основі їхніх інтересів, поведінки чи вподобань. Вони стали невід'ємною частиною цифрового простору, оскільки обсяги контенту й товарів значно перевищили можливості ручного пошуку та вибору.

Перші РС з'явилися у 90-х роках, а справжній прорив відбувся з розвитком великих даних та методів машинного навчання. Сьогодні РС охоплюють майже всі сфери життя, від інтернет-магазинів до стрімінгових платформ та соціальних мереж, допомагаючи користувачам знаходити бажаний контент і продукти. Їхня універсальність дозволяє адаптувати алгоритми до будь-якої галузі, де є дані про вибір чи взаємодію користувачів.

Запропонований підхід

Гіпотеза: уподобання користувачів суттєво залежать від емоційних реакцій на продукти, виражених у відгуках.

Основна ідея: прогнозування рейтингу користувача з урахуванням історичних вподобань, характеристик товару та емоційного контексту відгуків.

Гібридний підхід: поєднання колаборативної фільтрації (матричної факторизації) з емоційним аналізом (трансформаторний аналіз настроїв).

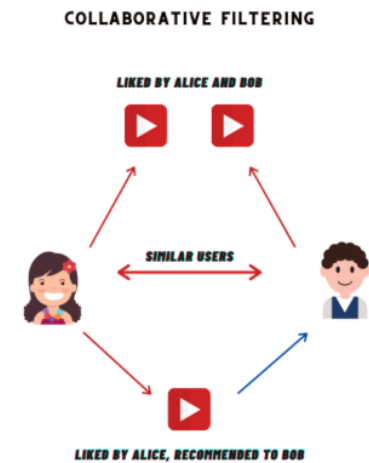
Використані дані

Для побудови рекомендаційної системи в даній роботі було вирішено обрати дані Amazon Reviews 2023, зібрані 23 командою McAuley Lab.

```
{'rating': 5.0,  
  'title': 'Such a lovely scent but not overpowering.',  
  'text': 'This spray is really nice. It smells really good, goes on really fine, and does the trick. I will se  
  'images': [],  
  'asin': 'B00YQ6X8E0',  
  'parent_asin': 'B00YQ6X8E0',  
  'user_id': 'AGKHLEW2SOWHNMFIJGBEC7INQ',  
  'timestamp': 1588687728923,  
  'helpful_vote': 0,  
  'verified_purchase': True}
```

Побудова рекомендаційної системи

Колаборативна фільтрація передбачає наявність великої матриці взаємодій користувачів та об'єктів (фільмів, товарів тощо). Елемент у цій матриці — це рейтинг, який користувач u надав об'єкту i , або індикатор взаємодії (наприклад, клік або покупка). Ідея методу полягає в тому, щоб передбачити невідомі елементи матриці, використовуючи відомі оцінки та подібності між користувачами або об'єктами.



Емоційний аналіз

Емоційний аналіз, або аналіз настрою (Sentiment Analysis), є ключовою областю обробки природної мови (NLP), яка дозволяє визначити емоційний тон, ставлення або суб'єктивну інформацію, що міститься в тексті. Його основна мета — класифікувати висловлювання як позитивні, негативні чи нейтральні, а також виявляти конкретні емоції. Це дає комп'ютерам можливість "розуміти" людські почуття, виражені у тексті. У розробленій системі було використано модель RoBERTa від Cardiff University. Вона є удосконаленою версією архітектури BERT, спеціалізованою на аналізі настрою в соціальних медіа, що робить її ідеальною для обробки користувацьких відгуків.

Побудована система



Якісний аналіз результатів - історичні покупки

```
all_beauty[all_beauty['user_id']=='AFZ54WN7RT3ZYJQV3CPA5AIUTACA'].iloc[0]
```

	rating	title	text	images	asin	parent_asin	user_id	timestamp	helpful_vote	verified_purchase	date
415046	4.0	All in one	This product is an all in one, you wash and sh...		B0BQWTXV2Q	B0BQWTXV2Q	AFZ54WN7RT3ZYJQV3CPA5AIUTACA	1677891706547	0	False	2023-03-04 01:01:46.547
415047	4.0	Does the job	This scrub does the job well, it's saturated w...		B0BBH2RLDX	B0BBH2RLDX	AFZ54WN7RT3ZYJQV3CPA5AIUTACA	1673476540892	0	False	2023-01-11 22:35:40.892
415048	4.0	Jade stays cool even without ice	I like this roller, I was impressed when I tri...		B0BJJRV3MX	B0BJJRV3MX	AFZ54WN7RT3ZYJQV3CPA5AIUTACA	1673471484424	1	False	2023-01-11 21:11:24.424

```
meta_all_beauty[meta_all_beauty['parent_asin']=='B0BQWTXV2Q'].iloc[0, 1]
```

'Hanz de Fuko Everything Wash for Hair, Face & Body - All-in-One Soap & Shampoo for Men, Women & Kids, Safe for Sensitive Skin - Minty Rum & Sandalwood, 8 oz. Bottle'

```
meta_all_beauty[meta_all_beauty['parent_asin']=='B0BBH2RLDX'].iloc[0, 1]
```

'Sia Botanics Revive Facial Scrub Gentle Antioxidant Scrub for Brighter Skin and Decongested pores - 3 Ounce'

```
meta_all_beauty[meta_all_beauty['parent_asin']=='B0BJJRV3MX'].iloc[0, 1]
```

'2 In 1 Essential Oil Roller Bottles Jade Roller for Face Facial Massage Roller Skin Care Beauty Tool, Green'

Якісний аналіз результатів - рекомендація

Top-5 recommendations for user AFZ54WN7RT3ZYJQV3CPA5AIUTACA:

1. Pastel Cream Blush 41 | Pink | Fresh | Healthy & Natural Loo...
Item ID: B08LYD2FMX
Predicted Rating: 4.67
2. UCANBE Fruit Pie Filling 30 Shade Eyeshadow Palette...
Item ID: B08CWV29PX
Predicted Rating: 4.54
3. Saltair Body Wash (Exotic Pulp)...
Item ID: B09V1TNP24
Predicted Rating: 4.52
4. Automatic Curling Iron, Cordless Auto Hair Curler with 6 Tem...
Item ID: B0B49VKP9J
Predicted Rating: 4.47
5. Ice Roller For Face | Ice Roller, Face Roller, Ice Face Roll...
Item ID: B0BRYZW7R
Predicted Rating: 4.45

Оцінювання системи за метриками

RMSE (Root Mean Squared Error): 0.7193 – показує середню квадратичну помилку прогнозів рейтингів

MAE (Mean Absolute Error): 0.5103 – відображає середню абсолютну помилку, демонструючи, наскільки в середньому відхиляються прогнози від реальних значень.

Accuracy: 0.8286 – точність класифікації, що вказує на частку правильно передбачених відгуків.

Precision: 0.9500 – точність позитивних прогнозів, тобто яка частка передбачених позитивних відгуків дійсно виявилася позитивною.

Висновки

Розроблено функціональну модель рекомендаційної системи для e-commerce проектів.

Система базується на сучасному підході до аналізу даних про поведінку користувачів та використанні методів колаборативної фільтрації.

Досягнуто високої точності прогнозування та релевантності рекомендацій.

Практична цінність: можливість впровадження в e-commerce платформи, покращення користувацького досвіду, збільшення продажів, оптимізація маркетингових стратегій.

Подальший розвиток

- Удосконалення емоційної класифікації
- Інтеграція додаткових типів даних та контексту

Джерела

Zhenhua Dong, Zhe Wang, Jun Xu, Ruiming Tang, Jirong Wen. A Brief History of Recommender Systems. In Proceedings of ACM Conference(Conference'17). ACM, New York, NY, USA. 2022. 9 pa.

Dmytro Poluriezov. Від концепції до реалізації: як побудувати ефективну рекомендаційну систему на основі машинного навчання за два дні, 2024. URL: <https://dou.ua/forums/topic/47504/>

Cherry Adam. How ASOS utilises AI to suggest your next purchase, 2020. URL:<https://recommend.pro/how-asos-utilises-ai-to-suggest-your-next-purchase/>

Amazon Reviews 2023 Dataset, 2023. URL: <https://amazon-reviews-2023.github.io/>

Дякую за увагу!