Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ОСОБЛИВОСТІ РОЗРОБКИ ДВУСТОРОННІХ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Текстова частина до курсової роботи

за спеціальністю «Комп’ютерні науки» 122

Керівник курсової роботи

д-р. техн. наук Глибовець А. М.

_____

(підпис)

Виконала студентка 3 курсу

Кучерук Д.В.

«____» _____ 2022 р.

Київ 2022

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри мережних технологій,
професор, доктор ф.-м. наук
_____ Г.І. Малашонок
«____»_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці 3-го курсу, факультету інформатики__
Кучерук Дарії Вікторівні__
Тема: Особливості розробки двусторонніх рекомендаційних систем

Зміст ТЧ до курсової роботи:

Зміст
Анотація

Вступ
1. Аналіз задачі побудови рекомендаційних систем
2. Дослідження основних алгоритмів побудови рекомендаційних систем
3. Реалізація та порівняльний аналіз результатів роботи двусторонніх рекомендаційних систем
Висновки
Список використаної літератури
Додатки

Дата видачі «___» _____ 2022 р. Керівник _____
(підпис)
Завдання отримала _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапу курсового проекту (роботи)	Термін виконання
1.	Обрання теми курсової роботи	08.10.2021
1.	Отримання завдання на курсову роботу	03.12.2021
2.	Огляд літератури за темою роботи	25.01.2022 – 05.03.2022
4.	Створення практичної частини роботи	06.03.2020-29.04.2021
5.	Написання текстової частини роботи	30.04.2021-04.05.2021
8.	Доопрацювання практичної частини роботи	05.05.2021-29.05.2021
11.	Остаточне оформлення текстової частини роботи та презентації	30.05.2021-05.06.2021
12.	Подання роботи на кафедру для перевірки на плагіат	06.06.2021

Зміст

Анотація

Рекомендаційні алгоритми використовуються у великій кількості сфер діяльності, таких як логістика, туризм, оптова торгівля, медицина, маркетинг, фінанси і багато інших. Дана курсова робота спрямована на огляд, аналіз та порівняння одних з фундаментальних алгоритмів отримання рекомендацій. Також у роботі розглядаються методи підготовки даних перед обробкою та способи отримання якісного аналізу отриманих рекомендацій.

Продемонстровано створення програмного коду реалізації алгоритмів колаборативної фільтрації та рекомендацій на основі метаданих об'єктів, використовуючи інструменти мови програмування Python та користувацького інтерфейсу JupyterLab.

Висновком є аналіз результатів роботи реалізованих підходів до побудови двусторонніх рекомендаційних систем.

Вступ

Найбільші технологічні компанії світу конкурують одна з іншою, пропонуючи все більш вдосконалені рекомендації своїм користувачам. Наприклад, всесвітньо відома компанія електронної комерції Amazon отримує 35 відсотків свого прибутку завдяки рекомендаційним алгоритмам [1]. Також, у 2009 році сервіс інтернет-телебачення Netflix нагородив команду розробників «BellKor's Pragmatic Chaos» одним мільйоном доларів США за покращення передбачень алгоритму компанії на 10.06 відсотків [2].

Такий попит на рекомендаційні системи пов’язаний з важливими задачами, які вони можуть вирішувати. З розвитком мережевих технологій з’явились великі бази даних, робота з якими ставала значно складнішою з роками. Користувачі систем з такими базами мали особисто обробляти велику кількість пропозицій, втрачаючи багато часу на фільтрацію результатів пошуку. Рекомендаційні системи допомагають користувачам позбутися проблеми інформаційного шуму та робити вибір лише з колекції релевантних для них об’єктів. Тому побудова такого алгоритму стає не тільки цікавою задачею, але й необхідністю в сучасному світі.

Задача побудови рекомендаційних систем має чітко зрозумілу суть, проте її вирішення потребують багатьох експериментів та порівняльних аналізів існуючих підходів. Оскільки на 2022 рік не існує однозначно визначеного універсального алгоритму рекомендацій, підготовки даних та налаштування параметрів рекомендаційної системи, за мету даної роботи було обрано вивчення існуючих підходів, реалізацію основоположних алгоритмів, враховуючи особливості предметної області, та порівняльний аналіз результатів рекомендацій.

Запропоновані реалізації основних сучасних підходів в побудові рекомендаційних систем були зроблені в домені онлайн-знайомств з фокусом на двусторонніх рекомендаційних системах з врахуванням вподобань обох сторін.

Таким чином, виконана робота сприяє розвитку сфери рекомендаційних систем, демонструючи порівняльний аналіз сучасних підходів, а також пропонуючи способи аналізу даних та оцінки роботи таких систем.

1 АНАЛІЗ ЗАДАЧІ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Знайомство з рекомендаційними системи

Рекомендаційною системою називається система, яка виконує інформаційну фільтрацію, що в результаті полегшує користувачу вибір об’єкту з запропонованої вибірки. Подібна фільтрація може бути виконана на основі окремих факторів, а також їх комбінації. Наприклад, такими факторами можуть бути самостійно визначені користувачем вподобання, історія поведінки користувача та неявні вподобання користувача.

Результатами рекомендацій можуть бути будь-які товари в інтернет-магазинах, відео та аудіо файли на відповідних сервісах, профілі людей в соціальних мережах та інші сутності.

1.2 Історія створення концепції рекомендаційних систем

Першою рекомендаційною системою направленою на вподобання користувачів була програма Grundy, що на основі отриманих від користувача запитів з вказаними вподобаннями видавала рекомендації книг з бібліотеки [3]. Програмний застосунок аналізував відповіді користувача, призначав його в групу за інтересами і за результатами перерозподілу рекомендував літературу.

Такий підхід у 2022 році може здатися примітивним та не надійним, проте в кінці 1970-их років таке рішення було науковою новизною, бо вперше автоматизовані рекомендації стали відповідати персональним вподобанням кожного користувача.

1.3 Підходи до створення рекомендаційних систем

Перший підхід до створення рекомендаційних систем оснований на неперсональних рекомендаціях. В таких системах користуються рейтингами, оцінками, кількістю переглядів та іншими кількісними характеристиками, що вказують на популярність продукту. Чим популярніший продукт – тим вище він займає місце у списку рекомендацій. Прикладом таких рекомендацій може бути

список ТОП-250 кращих фільмів на інтернет-платформі з базою даних кінокартин для перегляду.

Значним недоліком такого підходу є його суть. Популярність фільму «Форсаж» не дає гарантій, що він сподобається цінителю французького кінематографу. Таким чином цей недолік призвів до задачі створення персоналізованих рекомендаційних систем.

Основні алгоритми , що використовують у персоналізованих рекомендаціях - це контентна фільтрація, колаборативна фільтрація та гібридні системи [4].

Ідея контентної фільтрації полягає в тому, що користувачу рекомендуються об’єкти, що є схожими на ті, з якими користувач інтерактував раніше. Такий підхід вже має свої переваги над неперсоналізованими рекомендаціями, проте не позбавився і недоліків. Наприклад, для такого виду фільтрації обов’язковою вимогою є історія поведінки користувачів. У разі відсутності історії, або іншими словами, коли в системі з’являється новий користувач, рекомендації на основі історії побудувати неможливо. Проблема створення рекомендацій для нових користувачів в системі часто називається проблемою холодного старту [5].

Колаборативна фільтрація концептуально полягає в тому, що маючи інформацію про користувача чи об’єкт, система створює групу поведінок користувачів і створює рекомендацію на основі вподобань групи. Розглянемо основні види колаборативної фільтрації, починаючи з фільтрації основаної на даних про користувача (User based collaborative filtering).

User-based колаборативна фільтрація полягає в тому, що маючи інформацію про користувача, система знаходить схожих до нього користувачів та пропонує даному користувачу об’єкти, які сподобались схожим на нього користувачам. Така рекомендаційна система зможе надати релевантну

рекомендацію в тому випадку холодного старту, коли в системі з’явився новий користувач, який заповнив свій профіль даними про себе, дозволив обробку геопозиції та поділився іншою персональною інформацією.

Проблема холодного старту може бути вирішена за допомогою рекомендаційної системи, що основана на знаннях (Knowledge-based recommender system). Така система вирішує проблему відсутності історії поведінки користувача за допомогою конкретних запитів користувача.[6] Користувач має надати явну інформацію про свої вподобання, а система отримавши результат опитування починає в базі даних пошук об’єкту, що схожий на задані параметри. До створення Knowledge-based рекомендацій схожий функціонал виконували фільтри пошуку в системах, проте рекомендаційна система заснована на знаннях може принести в цей процес персоналізацію рекомендацій для кожного користувача.

Content-based колаборативна фільтрація надає користувачу рекомендації в залежності від об’єкту, який він зараз спостерігає. Система знаходить користувачів, яким даний об’єкт теж сподобався, фіксує історію їх поведінки та рекомендує об’єкти, що їм сподобались, даному користувачу.

User-based та Content-based колаборативні фільтрації відносяться до групи колаборативної фільтрації, заснованої на даних, що існують в пам’яті та не потребують параметричного машинного навчання [7].

Також існують алгоритми фільтрації, засновані на моделях машинного навчання. Наприклад, для вирішення задач рекомендаційних систем можуть використовуватись матрична факторизація, метод к-найближчих сусідів та нейронні мережі.

Персоналізовані рекомендації можуть бути згенеровані на базі гібридних рекомендаційних систем. Такі системи включають в себе кілька різних алгоритмів рекомендації з метою отримання кращих результати.

1.4 Аналіз особливостей двусторонніх рекомендаційних систем

В багатьох рекомендаційних системах користувачі та об’єкти рекомендацій є різними сутностями. Наприклад, системи рекомендацій одягу користувачам.

Двусторонні рекомендаційні системи відрізняються від звичайних тим, що при побудові рекомендації враховуються вподобання обох сторін. Одними з прикладів предметних областей, де такий підхід широко використовується – це платформи для онлайн-знайомств та системи пошуку сусідів по квартирі. [8]

Отже, рекомендаційні системи виконують важливу функцію в життєвому циклі сучасних інформаційних систем, допомагаючи користувачам зберігати якість та швидкість пошуку релевантних об’єктів. Також, існують різні підході до побудови рекомендаційних систем, що використовуються як окремо, так і разом для побудови найефективніших алгоритмів формування рекомендацій для користувачів.

2 ДОСЛІДЖЕННЯ ОСНОВНИХ АЛГОРИТМІВ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Дослідження методу побудови рекомендаційних систем, заснованого на контенті

Content-based підхід до побудови рекомендаційної системи має два популярні підходи до реалізації. Фільтрація на основі контенту є досить поширеним методом створення рекомендацій, проте обидва основні рішення мають свої переваги та недоліки.

По-перше, існує спосіб формування релевантних рекомендацій завдяки аналізу тільки мета-даних об’єкту. Для реалізації такого підходу треба мати порівняльну таблицю схожості одного об’єкту до іншого.

Часто описи об’єктів зберігаються у вигляді тексту, а тому виникає питання як програмним способом визначити схожість двох текстових об’єктів. Частим рішенням цього питання стає TF-IDF (Term Frequency - Inverse Document Frequency), ідея якого полягає в визначенні частоти появи слова в текстовому об’єкті, визначення важливості слів та вирахуванні оцінки кожному текстовому об’єкту [9].

TF відповідає за визначення частоти появи слова в текстовому об’єкті поділеної на кількість текстових об’єктів, в яких воно зустрічається.

$$TF(\text{слово}) = \frac{\text{частота появи слова в тексті}}{\text{сумарна кількість слів в тексті}}$$

IDF відповідає за визначення результату відношення всієї кількості текстових об’єктів до кількості текстових об’єктів, що мають в собі задане слово.

$$IDF(\text{слово}) = \frac{\text{кількість текстів всього}}{\text{сумарна кількість текстів, що містять дане слово}}$$

IDF використовується задля того, щоб надати рідкісним словам особливої важливості при підрахунку схожості двох текстових об’єктів.

В результаті отримання значень TF-IDF наступним кроком є побудова TF-IDF вектору, до якого вже стає можливим визначення схожості об’єктів за допомогою функцій подібності векторів, таких як косинусна подібність, евклідова відстань, відстань міських кварталів, Манхетеньська метрика, коефіцієнт кореляції Пірсона та інші.

Косинусна міра є одним із базових та широко поширених методів вирахування схожості векторних об’єктів. Особливо поширеним є цей метод в побудові рекомендаційних систем. Зручність цього методу полягає в тому, що він побудований на основі скалярного добутку в Евклідовому просторі.

$$\text{косинуснаПодібність}(x, y) = \frac{x * y}{|x||y|}$$

де x та y є векторами, а |x| та |y| їх довжинами відповідно.

Використання інструменту TF-IDF буває часто корисним, коли задача потребує знехтування тими словами, які зустрічаються занадто часто. Проте в інших випадках таке зниження оцінки може стати не доцільним і стає потреба в використанні альтернативних методів векторизації текстових об’єктів. Одним з таких методів може бути підрахунок слів у тексті і визначення їх частот.

По-друге, існує спосіб формування релевантних рекомендацій завдяки профілям користувачів та профілям об’єктів. Наприклад, якщо людина часто купляє продукцію певної компанії, то система буде давати рекомендацію купити інший продукт цієї ж компанії.

Досить важливою перевагою першого підходу з аналізом описів об’єктів є вирішення проблеми холодного старту. Окрім цього до переваг такого методу можна додати варіативність підходів до аналізу текстів. Проте важливим недоліком цього підходу стає те, що користувачеві пропонуються об’єкти

найбільш схожі між собою, що звужує вибірку рекомендацій та утворює бульбашку об’єктів, з якою користувачу складно вибратись.

У другого підходу теж є перевага вирішення проблеми холодного старту на доданок до незалежності від історії поведінки інших користувачів системи. Таким чином другий підхід аналізує лише об’єкти та самого користувача і не має вираховувати схожість самих користувачів один до одного. Великим недоліком такого методу є велика залежність від даних про користувача та описів об’єктів. У випадку, коли інформації з профілей недостатньо, рекомендації не будуть якісними.

2.2 Дослідження методів побудови рекомендаційних систем з колаборативною фільтрацією

Колаборативна фільтрація заснована на інформації про користувача (User-based collaborative filtering) має кілька основних способів реалізації.

Першим етапом розробки User-based collaborative filtering має бути створення матриці відношення користувачів до об’єктів. Така матриця містить в собі оцінки або рейтинги, які користувачі надали відповідним об’єктам. Такі рейтинги можуть бути виставлені явним способом самими користувачами, або вирахувані на основі існуючих даних про об’єкт та користувача.

Наступним етапом стає побудова матриці відношення користувачів один до одного. Матриця схожості користувачів може бути побудована за допомогою косинусної міри, коефіцієнту кореляції ранку Спірмена та інших. За допомогою побудованої матриці подібності стає можливим визначення найбільш схожих користувачів на даного користувача.

Маючи множину схожих до даного користувачів, наступним кроком є отримання списку тих об’єктів, які сподобались цим користувачам найбільше. Список вподобань користувачів фільтрується за частотою, висотою рейтингу та

наявності історії контакту даного користувача з об’єктами з рекомендаційного списку.

User-based collaborative filtering метод має важливий недолік, окрім холодного старту. Він полягає в тому, що користувачі мають тенденцію згодом змінювати свої вподобання, що змушує розробників системи підтримувати актуальність матриці схожості користувачів.

Натомість Item-based collaborative filtering надає більше гарантій в тривалій актуальності матриці подібності, оскільки описи об’єктів не змінюються згодом.

Метод побудови рекомендаційних систем з колаборативною фільтрацією заснований на описах об’єктів має аналогічний підхід до User-based collaborative filtering. Вибір функції для вирахування схожості об’єктів залежить від потреб розробників системи. Такий вибір може бути остаточно визначений після експериментів та порівняльного аналізу результатів. Поширеним методом отримання коефіцієнтів подібності є косинусна міра.

Метод створення систем з колаборативною фільтрацією заснований на машинному навчанні (Model-based collaborative filtering) допомагає досягнути кращої масштабованості системи при розробці. Одним з підходів до побудови Model-based колаборативної фільтрації є матрична факторизація.

Використання матричної факторизації обумовлено тим, що матриця рейтингів , що користувачі виставили об’єктам, з великою ймовірністю є розрідженою, бо більшість користувачів не оцінили всі об’єкти в системі. За допомогою обробки та аналізу поведінки користувача отримуються сховані параметри впливу, які використовуються для побудови рекомендацій.

Маючи матриці користувачів та об’єктів, вираховується матриця рейтингів шляхом множення матриці користувачів на транспоновану матрицю об’єктів.

Застосування машинного навчання в даному методі полягає в мінімізації функції втрат. Основна ціль алгоритму – мати найменшу різницю між прогнозованими та реальними рейтингами, що вираховуються за допомогою скалярного добутку між матрицями користувачів та об’єктів.

3 РЕАЛІЗАЦІЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ДВУСТОРОННІХ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

3.1 Аналіз та обробка даних перед реалізацією рекомендаційної системи з User-based колаборативною фільтрацією

Для реалізації двусторонньої рекомендаційної системи з Content-based колаборативною фільтрацією було використано набір даних онлайн-сервісу знайомств.

Набір даних для даної задачі включає в себе 4 файли :

1) user_chats.csv включає в себе таку інформацію:
    a. user_id – ідентифікатор користувача;
    b. partner_id – ідентифікатор користувача-партнера, з яким користувач з ідентифікатором user_id може мати зафіксовану розмову;
    c. isDialogue – бінарний показник наявності або відсутності зафіксованої розмови між двома користувачами з ідентифікаторами user_id та partner_id відповідно;
    d. message_count – кількість повідомлень у діалозі між двома користувачами з ідентифікаторами user_id та partner_id відповідно.
2) user_likes.csv фіксує наявність вподобання між користувачами і включає в себе таку інформацію :
    a. user_id – ідентифікатор користувача, що позначив вподобання до користувача з ідентифікатором partner_id ;
    b. partner_id – ідентифікатор користувача, якого вподобав користувач з ідентифікатором user_id ;
3) user_profile.csv містить наступну інформацію :
    a. user_id – ідентифікатор користувача;
    b. height – зріст користувача;

c. weight – маса тіла користувача;
d. hairColor – колір волосся користувача;
e. eyesColor – колір очей користувача;
f. bodyType – тип фігури користувача;
g. appearance – зовнішній вигляд користувача;
h. goalMoney – фінансові цілі користувача;
i. goalRelation – особисті стосункові цілі користувача;
j. goalEvening – ціль на вечір користувача;
k. goalTravel – цілі користувача пов’язані з мандрівками та подорожами;
l. relationship – особистий статус користувача (наявність та вид особистих стосунків);
m. children – кількість дітей у користувача;
n. alcohol – відношення користувача до вжиття спиртних напоїв;
o. smoking – відношення користувача до паління;
p. job – робота користувача;
q. occupation – вид діяльності користувача;
r. interests – теми, якими цікавиться користувач;
s. about – короткий текст, що користувач написав для саморепрезентації;

4) user_profile_view.csv фіксує наявність переглядів профілів користувачів і містить наступну інформацію :
   a. user_id – ідентифікатор користувача, що позначив вподобання до користувача з ідентифікатором partner_id ;
   b. partner_id – ідентифікатор користувача, якого вподобав користувач з ідентифікатором user_id ;

Інформація в файлі user_profile.csv зберігається у формі числових даних. Наприклад, світлий колір волосся має значення «1.0» замість «світле».

Для аналізу даних першим чином було розглянуто кількість користувачів, що хоч раз проявляли активність у сервісі (рисунок 3.1).

```
len(likes['user_id'].unique())
```

```
239507
```

Рисунок 3.1 – Кількість унікальних активних користувачів

Отже, кількість унікальних користувачів до обробки даних дорівнює 239507.

До обробки даних середня кількість вподобань, що виставили користувачі дорівнює приблизно 57.5 вподобань на користувача (рисунок 3.2).

```
# середня кількість вподобань, що користувачі видають
likes_count_each_user = likes.groupby('user_id')['partner_id'].count()
statistics.mean(likes_count_each_user.tolist())
```

```
57.42119854534523
```

Рисунок 3.2 - Середня кількість вподобань, що виставив кожен користувач

Максимальна кількість лайків, яку виставив один користувач дорівнює 28062 (рисунок 3.3)

```
likes_count_each_user_df = pd.DataFrame(likes_count_each_user)
likes_count_each_user_df.partner_id.max()
```

```
28062
```

Рисунок 3.3 - Максимальна кількість вподобань, що виставив один користувач

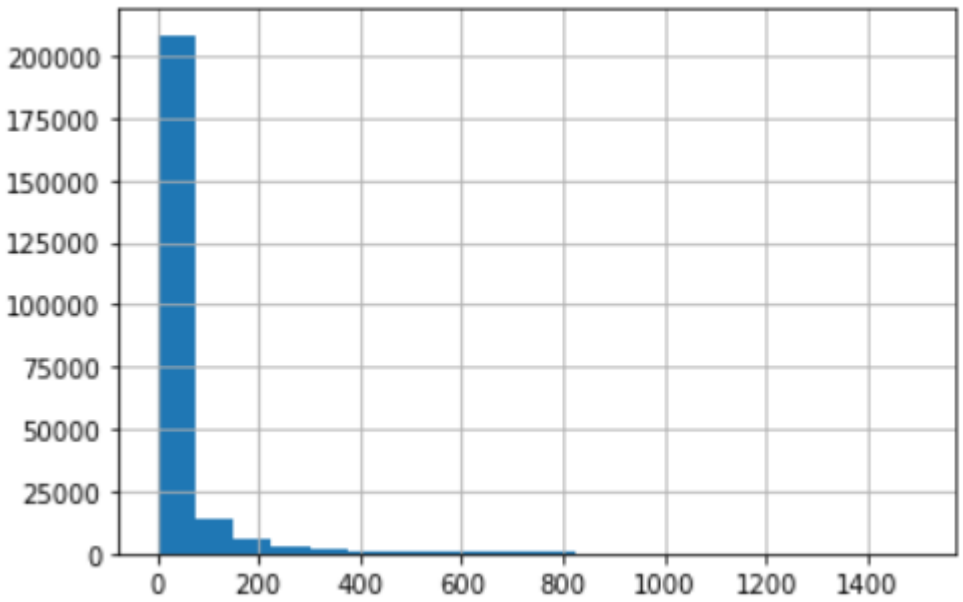

Рисунок 3.4 – Розподіл лайків та користувачів

З рисунку 3.4 спостерігається статистика, що більшість користувачів виставляють від 0 до 400 лайків.

```
# середня кількість повідомлень, що користувач пише зі свого аканту
texts_for_dialogue = chats.groupby('user_id')['message_count'].mean()
statistics.mean(texts_for_dialogue.tolist())
```

```
2.76307425909696
```

Рисунок 3.5 – Середня кількість повідомлень від користувача

З рисунку 3.5 видно, що до обробки даних середня кількість повідомлень, що один користувач пише зі свого акаунту дорівнює приблизно трьом.

```
# максимальна кількість повідомлень, що користувач написав з свого акаунту
messages_per_user_df = pd.DataFrame(chats.groupby('user_id')['message_count'].sum())
messages_per_user_df.message_count.max()
```

```
57048.0
```

Рисунок 3.6 – Максимальна кількість повідомлень, що написав один користувач

На малюнку 3.6 спостерігається статистика з максимальною кількістю повідомлень, що написав один користувач, що дорівнює 57048 повідомлень.

```
# максимальна кількість повідомлень, що користувач написав в середньому зі свого акаунту
texts_for_dialogue_df = pd.DataFrame(texts_for_dialogue)
texts_for_dialogue_df.message_count.max()
```

```
2299.0
```

Рисунок 3.7 – Максимальна кількість повідомлень, що написав користувач

З рисунку 3.6 та рисунку 3.7 видно, що до обробки даних максимальна кількість повідомлень, що користувач написав з одного акаунту дорівнює 57048, а максимальна кількість повідомлень, що користувач написав в одному діалозі дорівнює 2299. Такі показники значно відрізняється від поведінки більшості юзерів на рисунку 3.8, де спостерігається статистика, що більшість користувачів в середньому одному діалозі пишуть від 0 до 20 повідомлень.

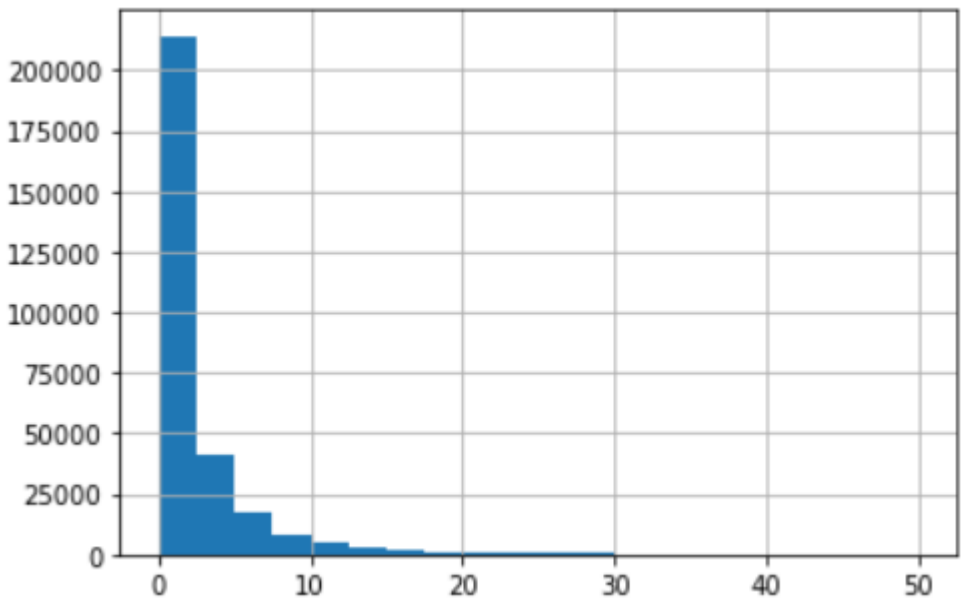

Рисунок 3.8 – Графік розподілу користувачів до середньої кількості повідомлень, що вони надсилають в одному діалозі.

Отже, виходячи з аналізу вхідних даних, виникає потреба в підготовці та обробці даних перед їх використання для побудови рекомендацій

Для подальшої роботи пропонується позбавитись надто активних та зовсім не активних користувачів. Активними користувачами будемо вважати тих, хто поставив більше 5000 лайків, а неактивними тих, хто поставив менше 100 лайків (рисунок 3.9).

```
# отримання таблички користувач-кількість поставлених ним лайків
likes_count_each_user_df = pd.DataFrame(likes_count_each_user)
filtered_likes_count_each_user_df = likes_count_each_user_df[likes_count_each_user_df.partner_id >= 100]
filtered_likes_count_each_user_df = likes_count_each_user_df[likes_count_each_user_df.partner_id <= 5000]
# список унікальних ідентифікаторів занадто активних та неактивних користувачів
active_users = filtered_likes_count_each_user_df.index.tolist()
```

Рисунок 3.9 – Обробка даних з прибиранням користувачів з нестандартною активністю

Також, треба прибрати користувачів, що пишуть дуже мало повідомлень та тих, хто пише занадто багато повідомлень. Видаляються з набору даних користувачі, що написали зі свого акаунту менше 100 повідомлень та більше 500 (рисунок 3.10).

```
# отримання таблички користувач-кількість поставлених ним лайків
likes_count_each_user_df = pd.DataFrame(likes_count_each_user)
filtered_likes_count_each_user_df = likes_count_each_user_df[likes_count_each_user_df.partner_id >= 100]
filtered_likes_count_each_user_df = likes_count_each_user_df[likes_count_each_user_df.partner_id <= 5000]
# список унікальних ідентифікаторів занадто активних та неактивних користувачів
active_users = filtered_likes_count_each_user_df.index.tolist()

# прибираємо користувачів, які в суммі написали менше 100 повідомлень або більше 500
messages_per_user_df = pd.DataFrame(chats.groupby('user_id')['message_count'].sum())
filtered_messages_per_user_df = messages_per_user_df[messages_per_user_df.message_count >= 100]
filtered_messages_per_user_df = filtered_messages_per_user_df[filtered_messages_per_user_df.message_count <= 500]
# список унікальних ідентифікаторів "не соціальних" та "занадто соціальних" користувачів
social_users = filtered_messages_per_user_df.index.tolist()
```

Рисунок 3.10 – Друге прибиранням користувачів з нестандартною активністю

Маючи список користувачів, що виходять за межі поширеної поведінки, з таблиці чатів прибираються усі стовпці з зазначеного списку (рисунок 3.11)

```
# видалення користувачів з нестандартною активністю
filtered_chats = chats[chats.user_id.isin(social_users)]
filtered_chats = filtered_chats[filtered_chats.user_id.isin(active_users)]
filtered_chats = filtered_chats[filtered_chats.partner_id.isin(active_users)]
filtered_chats = filtered_chats[filtered_chats.partner_id.isin(social_users)]
```

Рисунок 3.11 – Прибирання стовпців зі списку нестандартних користувачів

Показник	До обробки даних	Після обробки даних
Кількість унікальних активних користувачів	239507	24741
Середня кількість лайків, що надав користувач	57.42	33.54
Максимальна кількість лайків, що надав користувач	28062	1807
Середня кількість повідомлень в одному чаті	2.76	6.12
Максимальна кількість повідомлень, що користувач написав в середньому зі свого акаунту	2299	479
Максимальна кількість повідомлень, що написав один користувач зі свого акаунту	57048	479.0

Таблиця 3.1 – порівняльний аналіз результатів обробки даних

Отже, обробивши дані, кількість користувачів з показниками активності, що значно відрізняються від середніх значень, стає помітно, що кількість користувачів в наборі стає меншою, а середня кількість повідомлень в одному чаті підвищується (таблиця 3.1).

3.2 Реалізація підходу до створення рекомендаційної системи заснованого на User-based колаборативній фільтрації

Оскільки в оригінальній вибірці даних немає явних рейтингів, пропонується вирахувати рейтинги програмним способом.

Спочатку прибирається з таблиці чатів стовпець «isDialogue». Потім додається стовпець «like» за допомогою лівого зовнішнього з’єднування даної таблиці з таблицею лайків. Пусті значення змінюємо на 0. Далі доєднається присутність перегляду профілю аналогічним до стовпця «like» чином.

Після з’єднання потрібних стовпчиків в одній таблиці пропонується вирахування рейтингу за наступною формулою:

$$\text{Рейтинг} =$$

$$[\text{наявніть лайку}] * 5 +$$

$$\frac{[\text{кількість повідомлень у чаті}]}{\text{максимальна зафіксована кількість повідомлень в одному чаті}} * 3 +$$

$$[\text{наявніть перегляду профілю}] * 2$$

Таким чином, факт великого інтересу одного користувача до іншого визначається числом близьким до 10.

Результатом вирахування рейтингів є таблиця з наступними стовпцями :

1) user_id – ідентифікатор користувача;
2) partner_id - ідентифікатор користувача, якому виставив рейтинг користувач з ідентифікатором user_id;
3) rating – рейтинг

Оскільки розглядається двустороння рекомендаційна система, варто зазначити, що у таблиці рейтингів користувач є як суб’єктом так і об’єктом процесу виставлення оцінок.

Наступним кроком на основі рейтингів створюється таблиця користувач-користувач з взаємними рейтингами, що користувачі надали один одному.

На основі таблиці з взаємними рейтингами стає можлива побудова списку схожих користувачів (рисунок 3.12). Для вирахування схожості була обрана косинусна міра, застосована до вирахуваних оцінок.

Спочатку виокремлюються рейтинги, що виставив даний користувач, від усіх інших користувачів. Наступним кроком функцією косинусної міри вираховується схожість даного користувача з рештою користувачів. Отримавши ідентифікатори схожих користувачів, цей список сортується та з нього обираються 3 найбільш схожих на даного користувачів.

```
def get_similar(id, matrix, k=3):

    current_u = matrix[matrix.index == id]
    rest_of_u = matrix[matrix.index != id]
    get_sim_values = cosine_similarity(current_u,rest_of_u)[0]
    identify_sims = dict(zip(rest_of_u.index.tolist(), get_sim_values.tolist()))

    identify_sims_desc = sorted(identify_sims.items(), key=operator.itemgetter(1)).reverse()
    return [i[0] for i in identify_sims_desc[:k]]
```

Рисунок 3.12 – Отримання схожих користувачів за значеннями рейтингів

Наступним кроком є пошук рекомендацій. Першим етапом аналізуються 3 найбільш схожі до даного користувачі. Середнє значення їх рейтингів використовуватиметься у фільтрації тих користувачів, з якими даний користувач ще не контактував, а потім сортується за спаданням. З отриманого списку забираються перші 20 рекомендацій з найвищими середніми рейтингами (рисунок 3.13).

```
def find_recommendations(userId, simIds, m, n_recs=20):
    # отримання середніх ейтингів, що виставили з найбільш схожих на даного користувачі
    get_similar = pd.DataFrame(m[m.index.isin(simIds)].mean(axis=0), columns=['mean'])

    # отримання транспонованого вектора з рейтингами, що виставив даний користувач
    # та фільтрація тільки тих користувачів, що наш юзер не оцінював
    ratings_df = m[m.index == userId].transpose()
    ratings_df.columns = ['score']
    ratings_df = ratings_df[ratings_df['score']==0]

    # список тільки тих середніх рейтингів, що користувач не оцінював раніше
    get_similar_filtered = get_similar[get_similar.index.isin(ratings_df.index.tolist())]
    get_similar_ordered = get_similar.sort_values(by=['mean'], ascending=False)

    best_matches_ids =  get_similar_ordered.head(n_recs).index.tolist()
    return user_profile[user_profile['user_id'].isin(best_matches_ids)]
```

Рисунок 3.13 – Створення рекомендацій

3.3 Аналіз результатів роботи створеної рекомендаційної системи заснованої на User-based колаборативній фільтрації

Для оцінки результатів алгоритму пропонується порівняти мета-данні тих користувачів, які раніше сподобались даному користувачу з мета-даними тих користувачів, яких порекомендувала система.

Максимальна кількість балів – 17. Якщо рекомендований користувач набрав 17 балів – це означає, що він з великою ймовірністю сподобається даному користувачу.

Для порівняння мета-даних пропонується проаналізувати попередні лайки користувача та зробити примірник його уподобань. Примірник вираховується за наступною схемою:

1) Отримання мета-даних тих користувачів, чий профіль переглянув даний користувач та поставив їм лайк. Тобто тих користувачів, які отримали рейтинг більше 7 балів від даного користувача;
2) Аналіз мета-даних цих користувачів. Примірник вподобань користувача складається з наступних показників:
    a. середній зріст попередніх вподобань;

b. середня маса тіла попередніх вподобань;
c. найчастіший уподобаний колір волосся;
d. найчастіший уподобаний колір очей;
e. найчастіший уподобаний тип фігури;
f. найчастіший уподобаний зовнішній вигляд;
g. найчастіший уподобаний фінансовий стимул;
h. найчастіша уподобана ціль на стосунки;
i. найчастіша уподобана ціль на подорожі;
j. найчастіший уподобаний сімейний статус;
k. найчастіший уподобаний показник кількості дітей;
l. найчастіший уподобаний показник відношення до вживання спиртних напоїв;
m. найчастіший уподобаний показник відношення до паління;
n. найчастіший уподобаний показник роботи;
o. найчастіший уподобаний показник роду занять;
p. найчастіший уподобаний набір хобі.

3) Аналіз мета-даних рекомендованих користувачів. Оцінка якості рекомендації залежить від кількості співпадінь з примірником уподобань користувача описаного в другому пункті цієї схеми.

Двусторонній підхід аналізу результатів буде полягати в тому, що для кожного рекомендованого користувача теж генерується список рекомендацій. Результатом двусторонньої рекомендації будуть користувачі, з якими за пропозицією колаборативної фільтрації відбудуться взаємні вподобання.

Отже, застосувавши на користувачі з ідентифікатором 44287673 функцію формування 20 рекомендацій з рисунку 3.13, маємо наступну статистику для простої колаборативної фільтрації на рисунку 3.14 і рисунку 3.15 та статистику для методу двусторонньої рекомендації на рисунку 3.16.

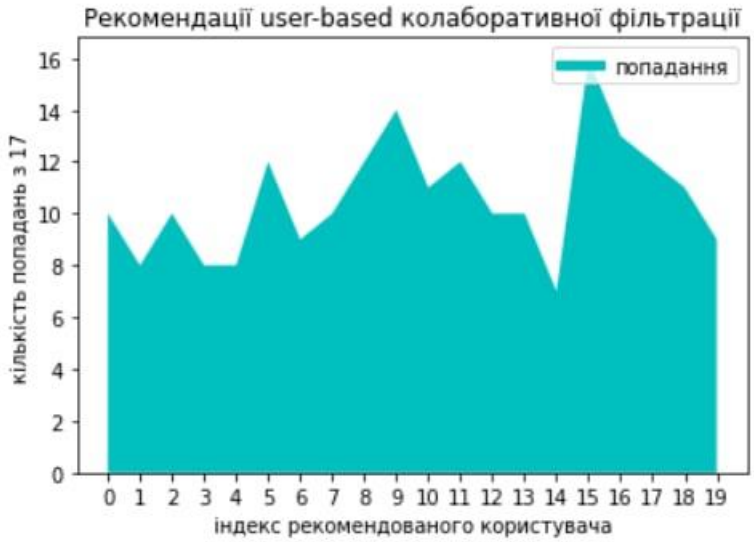

Рисунок 3.14 – Статистика рекомендацій з user-based колаборативною фільтрацією

```
1635  got this statistics right:  10.0 /17
2170  got this statistics right:  8.0 /17
8629  got this statistics right:  10.0 /17
10081  got this statistics right:  8.0 /17
10260  got this statistics right:  8.0 /17
12263  got this statistics right:  12.0 /17
37162  got this statistics right:  9.0 /17
58599  got this statistics right:  10.0 /17
92737  got this statistics right:  12.0 /17
99312  got this statistics right:  14.0 /17
135792  got this statistics right:  11.0 /17
150989  got this statistics right:  12.0 /17
153515  got this statistics right:  10.0 /17
153700  got this statistics right:  10.0 /17
155761  got this statistics right:  7.0 /17
162112  got this statistics right:  16.0 /17
162310  got this statistics right:  13.0 /17
181627  got this statistics right:  12.0 /17
196630  got this statistics right:  11.0 /17
262565  got this statistics right:  9.0 /17
```

```
statistics.mean(hits)
```

```
10.6
```

Рисунок 3.15 – Статистика кількості попадань кожного рекомендованого юзера в уподобання даного користувача

Отже, з малюнку 3.15 видно, що середня кількість попадань близька до 11, що є показником того, що більшості рекомендованих користувачів буде легше знайти спільні теми у процесі знайомства.

```
number_of_mutual_recommendations = 0
for recomended_user in temp_res["user_id"].tolist():
    sec_side = find_recommendations(recomended_user, get_similar(recomended_user, rating_m), rating_m)
    sec_side = sec_side[sec_side["user_id"].isin([44287673])]
    number_of_mutual_recommendations += len(sec_side["user_id"].tolist())

print(number_of_mutual_recommendations, " взаємних рекомендацій було сформовано для користувача 44287673")
```

```
7  взаємних рекомендацій було сформовано для користувача 44287673
```

Рисунок 3.16 – Двустороння рекомендація

На малюнку 3.16 зображено, як з 20 рекомендацій сформованих колаборативною фільтрацією, в результат двусторонньої рекомендації вийшло 7 користувачів, яким система порекомендувала даного користувача.

3.4 Аналіз та обробка даних перед реалізацією Content-based рекомендаційної системи

Для побудови рекомендацій на основі описів об’єктів було використано дані з файлу user_profile.csv.

Перед реалізацію алгоритму необхідно обробити вхідні дані (рисунок 3.17), а саме першим кроком відчистити таблицю від строчок повністю заповнених пустими значеннями. Такі строчки не дають корисної інформації для аналізу і сповільнюють процес формування рекомендацій.

Наступним кроком решта пустих значень замінюються на значення «-1.0», виходячи з факту, що нумерація всіх значень параметрів починається з нуля. Така обробка знадобиться при вирахуванні схожості користувачів. Також для зручності використання функції бібліотеки текстової обробки даних в таблиці всі числові значення множаться на 10 та приводяться до типу цілого числа.

Важливим кроком є нормалізація стовпчику інтересів, де всі пусті значення замінюються значенням «-10» та приводяться до стрічкового типу даних.

```
user_profile = pd.read_csv( "C:/Users/Dariia/3D Objects/user_profile.csv",  dtype={"interests":str})
user_profile.drop('Unnamed: 0', axis=1, inplace=True)
user_profile.drop('about', axis=1, inplace=True)
user_profile.dropna(subset=['height','weight','hairColor','eyesColor','bodyType','appearance','goalMoney','

user_profile = user_profile.dropna(axis=0, how='all')
user_profile['interests'] = user_profile['interests'].fillna(value="-10,-10")
user_profile = user_profile.fillna(-1.0)
user_profile = user_profile[user_profile.user_id.isin(social_users)]
user_profile = user_profile[user_profile.user_id.isin(active_users)]

user_profile['user_id'] = user_profile['user_id'].astype('int')
user_profile['height'] = user_profile['height'].astype('int')*10
user_profile['weight'] = user_profile['weight'].astype('int')*10
user_profile['hairColor'] = user_profile['hairColor'].astype('int')*10
user_profile['eyesColor'] = user_profile['eyesColor'].astype('int')*10
user_profile['bodyType'] = user_profile['bodyType'].astype('int')*10
user_profile['appearance'] = user_profile['appearance'].astype('int')*10
user_profile['goalMoney'] = user_profile['goalMoney'].astype('int')*10
user_profile['goalRelation'] = user_profile['goalRelation'].astype('int')*10
user_profile['goalEvening'] = user_profile['goalEvening'].astype('int')*10
user_profile['goalTravel'] = user_profile['goalTravel'].astype('int')*10
user_profile['relationship'] = user_profile['relationship'].astype('int')*10
user_profile['children'] = user_profile['children'].astype('int')*10
user_profile['alcohol'] = user_profile['alcohol'].astype('int')*10
user_profile['smoking'] = user_profile['smoking'].astype('int')*10
user_profile['job'] = user_profile['job'].astype('int')*10
user_profile['occupation'] = user_profile['occupation'].astype('int')*10
```

Рисунок 3.17 – Обробка даних перед реалізацією Content-based рекомендаційної системи

3.5 Реалізація підходу до створення Content-based рекомендаційної системи

Перед створенням рекомендацій в таблиці з параметрами користувачів треба зробити окрему колонку, що міститиме всі параметри поєднанні між собою (рисунок 3.18).

Таке поєднане значення потрібно для знаходження схожих користувачів. Ідея полягає в тому, що в кожному такому поєднаному рядку рахується кількість однакових слів і на основі цього будуються числові вектори частот для кожного користувача. Такі вектори потім використовуються для порівняння користувачів з використанням косинусної міри.

Оскільки дані в таблиці зберігаються в числовому форматі і в поєднаному вигляді важко розпізнати суть числового значення, в процесі створення стовпчику з поєднаними значеннями до кожного числа додаються суфікси

вигляду «_перші літери параметру». Наприклад, значення 10 для волосся перетворюється в «10_he».

Також для побудови числового вектору було додано ваги до кожного параметру, що позначає важливість параметру та відповідно кількість разів він повторюється в результуючому поєднанні параметрів.

```
w_height = 2
w_weight = 2
w_haircolor = 1
w_eyescolor = 1
w_bodytype = 2
w_appearance = 3
w_goalmoney = 2
w_goalrelation = 3
w_goalEvening = 1
w_goalTravel = 2
w_relationship = 3
w_children = 3
w_alcohol = 3
w_smoking = 3
w_job = 2
w_occupation = 1
w_interests = 2

def squeeze(params):

    interests = ""
    for interest in ast.literal_eval(str(params['interests'])) :
        interests = interests.join(str(interest))

    interests = interests*w_interests+"_int"
    return (str(params.height)+"_he")*w_height + ' ' + (str(params.weight)+"_w")*w_weight + ' ' + (s

user_profile['squeezed'] = user_profile.apply(squeeze,axis=1)
user_profile = user_profile.reset_index()
user_profile
```

Рисунок 3.18 – Формування колонки поєднаних параметрів

Рекомендації отримуються на основі результатів застосування косинусної міри на числових векторах отриманих після збирання всіх параметрів користувачів в один показник. Результатом рекомендації є 20 користувачів, що найближчі до показнику вподобань користувача (рисунок 3.19).

```
def get_recs(u_id, u_profile,cosSims,k):

    number = u_profile[u_profile['user_id']==u_id].index.values[0]
    ratings = list(enumerate(cosSims[number]))
    ratings_desc = sorted(ratings, key=lambda i: i[1], reverse=True)

    user_profiles = pd.read_csv( "C:/Users/Dariia/3D Objects/user_profile.csv",  dtype={"interests":str})
    user_profiles.drop('Unnamed: 0', axis=1, inplace=True)
    user_profiles.drop('about', axis=1, inplace=True)

    recs_nums = [r[0] for r in ratings_desc[1:(k+1)]]
    ids = u_profile.iloc[recs_nums].user_id.tolist()
    return user_profiles[user_profiles.user_id.isin(ids)]

encoded_description = CountVectorizer().fit_transform(user_profile['squeezed'])
count_sims = cosine_similarity(encoded_description, encoded_description)
recs = get_recs(1, user_profile, count_sims, 20)
```

Рисунок 3.19 – побудова Content-based рекомендацій

3.6 Аналіз результатів роботи створеної Content-based рекомендаційної системи

На рисунку 3.20 видно, що односторонні рекомендації знаходяться близько до середніх вподобань користувача, проте двустороння рекомендація не дає позитивних результатів (рисунок 3.21).

```
rec_ids = recs.index
hits = []
evaluate_res(recs,rec_ids,users_preferences)
```

```
18709  got this statistics right:  15.0 /17
27272  got this statistics right:  13.0 /17
27275  got this statistics right:  13.0 /17
31654  got this statistics right:  14.0 /17
42908  got this statistics right:  14.0 /17
52006  got this statistics right:  15.0 /17
55753  got this statistics right:  15.0 /17
99838  got this statistics right:  16.0 /17
124859  got this statistics right:  15.0 /17
128547  got this statistics right:  16.0 /17
133474  got this statistics right:  15.0 /17
139699  got this statistics right:  14.0 /17
156161  got this statistics right:  14.0 /17
162846  got this statistics right:  14.0 /17
168036  got this statistics right:  14.0 /17
176156  got this statistics right:  16.0 /17
177062  got this statistics right:  15.0 /17
180600  got this statistics right:  16.0 /17
182873  got this statistics right:  17.0 /17
303080  got this statistics right:  16.0 /17
```

```
statistics.mean(hits)
```

```
14.85
```

Рисунок 3.20 – Результати односторонньої Content-based рекомендації

```
number_of_mutual_recommendations = 0
for recomended_user in recs["user_id"].tolist():
    sec_side = get_recs(recomended_user, user_profile, count_sims, 20)
    sec_side = sec_side[sec_side["user_id"].isin([1])]
    number_of_mutual_recommendations += len(sec_side["user_id"].tolist())

print(number_of_mutual_recommendations, " взаємних рекомендацій було сформовано для користувача 44287673")
```

```
0  взаємних рекомендацій було сформовано для користувача 44287673
```

Рисунок 3.21 - Результати двусторонньої Content-based рекомендації

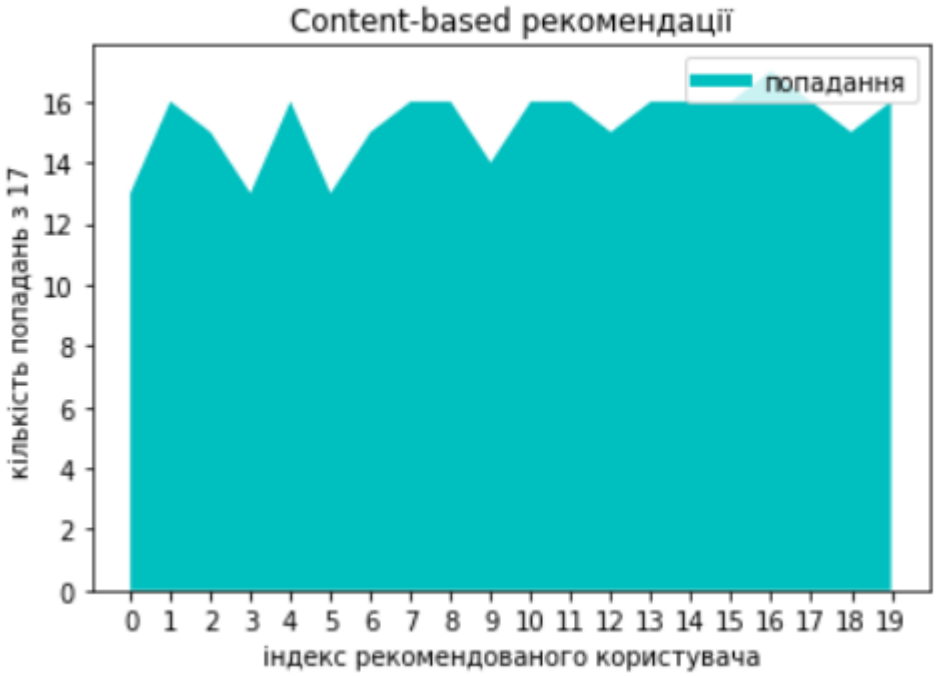

Рисунок 3.22 – Статистика content-based рекомендацій

З рисунку 3.22 зрозуміло, що рекомендації content-based методу мають високі показники попадання в уподобання користувача, проте не показують позитивних результатів при використанні двусторонніх рекомендацій.

Така поведінка пояснюється тим, що при контентній фільтрації не враховується попередня взаємодія користувачів один з одним. Тобто, якщо користувачі за кольором волосся та відношенням до паління схожі на вподобання користувача, це ще не означає що цим користувачам сподобається даний юзер.

3.7 Порівняльний аналіз роботи реалізацій двох підходів до створення рекомендаційних систем

В таблиці 3.2 зафіксовано порівняльний аналіз побудованих рекомендаційних систем на базі контентної обробки та з використанням User-based колаборативної фільтрації. Стає помітна тенденція, що контентний підхід видає кращі результати в односторонній рекомендації, коли user-based колаборативна фільтрація надає кращі результати для двусторонньої рекомендації.

Показник	Content-based підхід	User-based підхід
Середня кількість попадань в уподобання фіксованого тестового користувача	10.6	15.3
Кількість успішних двусторонніх рекомендацій з 20 у фіксованого тестового користувача	7	0
Середня кількість успішних двусторонніх рекомендацій у 5 фіксованих тестових користувачів з 10 рекомендацій	0.8	0

Таблиця 3.2

Також було проведено порівняльний аналіз якості рекомендацій на основі підрахунку середньої кількості попадань в уподобання десяти фіксованих тестових користувачів для user-based колаборативної фільтрації (рисунок 3.23) та content-based підходу (рисунок 3.24).

Результати показують, що більшість user-based рекомендацій знаходяться в діапазоні від 7 до 10 балів попадання в уподобання користувача. Оскількі рекомандації даного методу засновані на історії поведінки користувачів, а саме генерація рекомендацій на основі тих лайків, що поставили схожі на даного

користувачі, побудовані пропозиції дають можливість варіативності вибору. Також важливо позначити, що хоча content-based рекомендації мають високе співпадіння з уподобаннями користувача (в середньому від 11 до 15), все рівно залишається проблема монотонності пропозицій та відсутності варіативності пропонованих об’єктів.

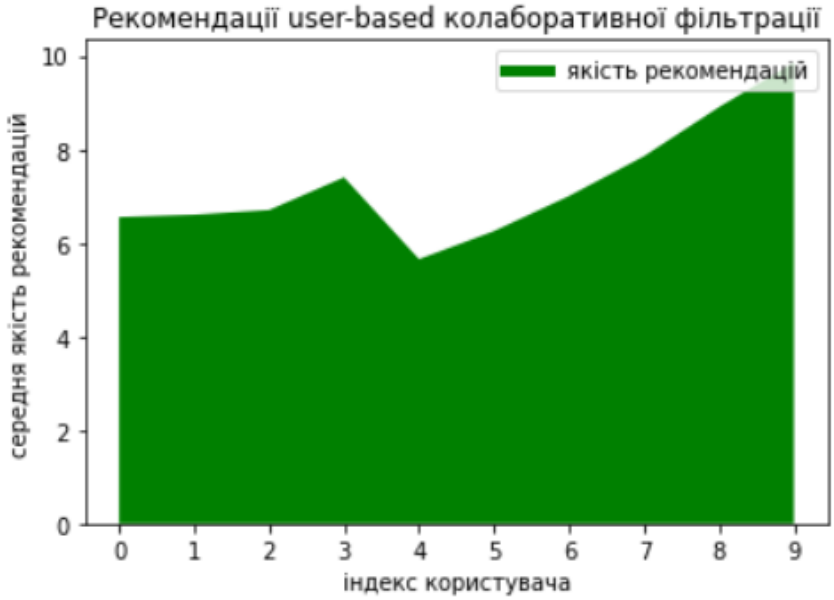

Рисунок 3.23 – якості рекомендацій User-based колаборативної фільтрації

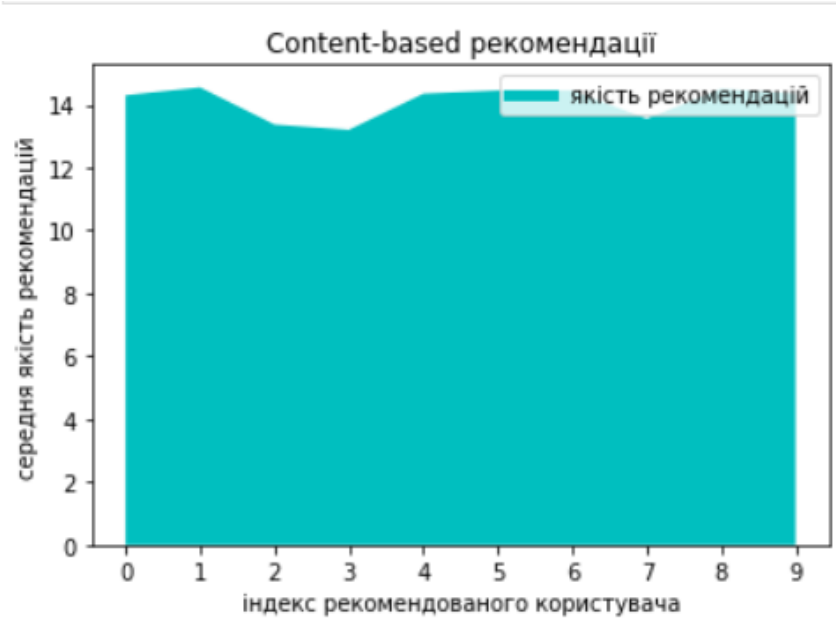

Рисунок 3.24 – якості рекомендацій Content-based колаборативної фільтрації

Важливим є аналіз двусторонніх рекомендацій обох систем. З рисунку 3.25 та рисунку 3.26 видно, що взаємні рекомендації засновані на user-based колаборативній фільтрації мають кращі результати за контенту фільтрацію. Такий результат пов’язаний з тим, що контента фільтрація не враховує попередню поведінку користувачів та не користується схожістю користувачів за їх історією активності.

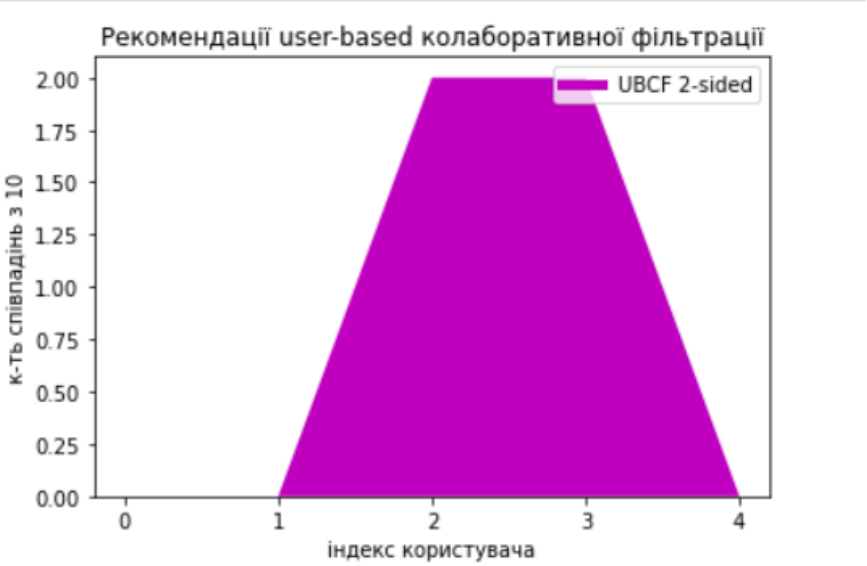

Рисунок 3.25 – Аналіз двусторонньої User-based колаборативної фільтрації

```
for cb_tid in cb_twosided_testset:
    number_of_mutual_recommendations = 0

    recs = get_recs(cb_tid, 10)
    recs['interests'] = recs['interests'].fillna(value="")

    for recomended_user in recs["user_id"].tolist():
            sec_side = get_recs(recomended_user, 10)
            sec_ids = sec_side.index
            sec_side = sec_side[sec_side["user_id"].isin([cb_tid])]
            number_of_mutual_recommendations += len(sec_side["user_id"].tolist())
    cb_mutuals.append(number_of_mutual_recommendations)
cb_mutuals
```

```
[0, 0, 0, 0, 0]
```

Рисунок 3.26 - Аналіз двусторонньої Content-based рекомендації

Отже, обидва підходи мають слабкі та сильні сторони у своїй ефективності. Content-based рекомендації добре виконують односторонні

рекомендації та можуть бути релевантні у системах, де об’єкт та суб’єкт рекомендації – це різні сутності. User-based колаборативна фільтрація надає трохи гірші односторонні рекомендації, проте у двусторонніх рекомендаціях надає значно більшу кількість релевантних результатів. Такий підхід матиме успіх у системах, де передбачаються двусторонні рекомендації.

Висновки

По завершенню виконання даної курсової роботи було досліджено основні методи побудови рекомендаційних систем, таких як методи колаборативної фільтрації, методи засновані на машинному навчанні, методи засновані на описовій інформаціїї про об’єкт та інші. Також були опановані навички побудови двох основних методів розробки рекомендаційних систем.

Виконане дослідження призвело до більш глибокого розуміння масштабності застосування рекомендаційних систем у сфері інформаційних технологій. Аналіз підходів до побудови рекомендацій показав, що різні задачі для рекомендацій потребують різних методів рішення, а іноді і поєднання алгоритмів.

Було розглянуто та проаналізовано особливості розробки двусторонніх рекомендаційних систем на прикладі рекомендаційної системи для онлайн-знайомств.

На основі результатів виконання даної курсової роботи було зроблено висновки про особливості методу колаборативної фільтрації та методу заснованому на аналізі описів об’єктів. Головним чином, порівняльний аналіз показав, що методи колаборативної фільтрації надають кращі результати в двусторонній рекомендації, проте метод заснований на описах об’єктів має кращі показники для односторонніх рекомендацій.

Список використаної літератури

1. How retailers can keep up with consumers [Електронний ресурс] / Ian MacKenzie, Chris Meyer, and Steve Noble. – 2013. – Режим доступу до ресурсу: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers
2. BellKor's Pragmatic Chaos Wins $1 Million Netflix Prize by Mere Minutes [Електронний ресурс] / Eliot Van Buskirk. – 2009. – Режим доступу до ресурсу: https://www.wired.com/2009/09/bellkors-pragmatic-chaos-wins-1-million-netflix-prize/
3. E. Rich (1979): User modeling via stereotypes, Cognitive Science, Vol. 3, No. 4, pp. 329–354.
4. Types of Recommendation Systems & Their Use Cases [Електронний ресурс] / Maruti Techlabs. – 2021. – Режим доступу до ресурсу: https://medium.com/mlearning-ai/what-are-the-types-of-recommendation-systems-3487cbafa7c9
5. Cold-Start Problem in Recommender Systems and its Mitigation Techniques [Електронний ресурс] / Vijaysinh Lendave. – 2021. – Режим доступу до ресурсу: https://analyticsindiamag.com/cold-start-problem-in-recommender-systems-and-its-mitigation-techniques/
6. Knowledge-Based Recommender Systems: An Overview [Електронний ресурс] / Jackson Wu. – 2019. – Режим доступу до ресурсу: https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba
7. Various Implementations of Collaborative Filtering [Електронний ресурс] / Prince Grover. – 2017. – Режим доступу до ресурсу: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
8. A. Goswami, F. Hedayati, and P. Mohapatra, "Recommendation Systems for Markets with Two Sided Preferences," 2014 13th International Conference

on Machine Learning and Applications, 2014, pp. 282-287, doi: 10.1109/ICMLA.2014.51.

9. Introduction to TWO approaches of Content-based Recommendation System [Електронний ресурс] / Kevin Luk. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c