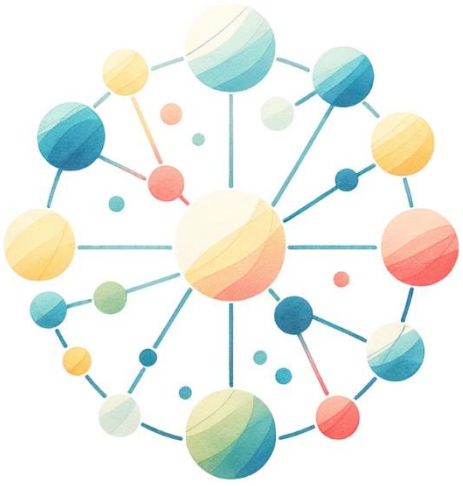
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Мелешко Є.В., Міхав В.В.

РЕКОМЕНДАЦІЙНІ СИСТЕМИ У СКЛАДНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ



НАВЧАЛЬНИЙ ПОСІБНИК

Кропивницький
2025

УДК 004.7+519.876.5
М 47

Мелешко Є.В., Міхав В.В.

М 47 Рекомендаційні системи у складних комп'ютерних мережах: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: ЦНТУ, 2025. – 156 с.

У цьому навчальному посібнику розглядаються моделі та методи роботи рекомендаційних систем у складних комп'ютерних мережах, їх показники якості та робастності, а також підіймаються питання їх інформаційної безпеки та імітаційного моделювання. Матеріал про імітаційне моделювання рекомендаційних систем подається на основі теорії складних мереж, яка вивчає системи з великою кількістю компонентів, що можуть динамічно змінюватися і взаємодіяти між собою, і використовується для аналізу, моделювання, прогнозування, візуалізації, оптимального управління системами. Посібник розраховано на студентів, що навчаються на технічних спеціальностях, пов'язаних з програмуванням та проектуванням комп'ютерних систем та мереж.

УДК 004.7+519.876.5

Рецензенти:

Петренюк Володимир Ілліч, кандидат фізико-математичних наук, доцент, доцент кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету

Дрєєв Олександр Миколайович, кандидат технічних наук, доцент, доцент кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету

Рекомендовано Вченою радою
Центральноукраїнського національного технічного університету
як навчальний посібник для студентів закладів вищої освіти
протокол № 5 від 28.01.2025 р.

ЗМІСТ

ВСТУП

У зв’язку зі зростанням обсягів інформації в Інтернеті, збільшенням розмірів та складності комп’ютерних мереж, а також їх додатків, зокрема, рекомендаційних систем, стають все важливішими питання підвищення якості обслуговування та оперативності пошуку даних й забезпечення інформаційної безпеки у них, які все частіше досліджують за допомогою теорії складних мереж.

Теорія складних мереж є новою науковою галуззю, що вивчає системи з великою кількістю компонентів, які можуть динамічно змінюватися та взаємодіяти між собою, наприклад, комп’ютерні, соціальні, біологічні, електричні, транспортні мережі, і використовується для моделювання, аналізу, візуалізації, прогнозування, оптимального управління поведінкою різних систем та мереж. Усе це робить актуальним використання теорії складних мереж для розробки методів підвищення якості обслуговування та кібербезпеки у рекомендаційних системах для складних комп’ютерних мереж.

Запропонований посібник розглядає питання побудови рекомендаційних систем, оцінки їх якості та робастності, а також забезпечення інформаційної безпеки їх роботи. У ньому наведені моделі та методи колаборативної та контентної фільтрації для формування рекомендацій користувачам складних комп’ютерних мереж з метою підвищення оперативності пошуку даних у великих масивах інформації.

Даний посібник призначений в першу чергу для студентів, які вивчають програмування та проектування комп’ютерних систем і мереж, а також аналіз даних у них. Він також буде корисний для широкого кола технічних фахівців, що бажають підвищити свій рівень знань в цій області.

РОЗДІЛ 1.
МОДЕЛІ ТА МЕТОДИ СИНТЕЗУ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1. Види рекомендаційних систем

Під рекомендаційними системами будемо розуміти програмне забезпечення, що використовується для прогнозування того, які об’єкти (товари, послуги, публікації, фільми, новини тощо) будуть цікаві користувачу у майбутньому, на основі зібраної раніше інформації про нього та його дії у системі (веб-сайті чи додатку). Елементами рекомендаційної системи будемо називати сукупність користувачів та об’єктів системи.

Перш ніж перейти до розгляду моделей та методів синтезу рекомендаційних систем, проведемо їх класифікацію за призначенням та способами збору даних про користувачів.

Основні сфери застосування рекомендаційних систем на сьогоднішній день можна розділити на наступні [1-9]:

1. *Рекомендаційні системи в електронній комерції* – виконують завдання по збільшенню відсотка продажів товарів та послуг в Інтернет-магазинах. Крім того, призначення рекомендаційної системи в електронній комерції – скоротити час пошуку потрібних товарів та послуг відвідувачам веб-сайту.

2. *Рекомендаційні системи на контент-орієнтованих веб-сайтах та в соціальних мережах* – покликані збільшувати час перебування відвідувача на сайті, глибину перегляду і залученість, полегшують пошук та доступ користувача до потрібної йому інформації.

3. *Рекомендаційні системи в пошукових роботах* – застосовуються для підвищення релевантності та покращення ранжування пошукової видачі для користувача.

За типом одержувача рекомендацій рекомендаційні системи можна класифікувати на [6, 9]:

1. *Індивідуальні рекомендаційні системи* – для кожного користувача

системи формується окремий список рекомендацій.

2. *Групові рекомендаційні системи* – список рекомендацій формується не для окремого користувача, а для певної групи користувачів. Розбиття користувачів на групи може відбуватися за різними критеріями.

Рекомендаційні системи формують свої пропозиції користувачам на основі зібраної про них інформації. Інформацію про поточні вподобання користувачів можна збирати по-різному. За методами збору даних про користувачів рекомендаційні системи можна класифікувати на [6, 10, 9]:

1. *Системи з явним збором даних.* Користувач добровільно надає системі необхідні для її роботи дані. Явний збір даних може полягати в проханні користувачу поставити диференційовану оцінку тому чи іншому об’єкту, створити список «улюблених» об’єктів або заповнити анкету, пройти тест. Основний недолік методу – користувачу необхідно здійснювати деякий набір дій, в чому він може бути незацікавленим або не мати часу на їх виконання.

2. *Системи з неявним збором даних.* Відбувається за допомогою спостереження за поведінкою користувача – оцінками, покупками, переходами за посиланнями тощо. Перевагою методу є те, що для збору даних користувачу не треба здійснювати ніяких додаткових дій, все, що йому потрібно – просто користуватися веб-сайтом в своєму звичайному режимі. Але даний метод підіймає ряд етичних питань, наприклад, таких як приватність даних.

3. *Системи, що поєднують обидва типи збору даних.* Частина даних одержується явно від користувача, а інша частина збирається непомітно для нього.

На сьогоднішній день найбільш поширеними є рекомендаційні системи з неявним збором даних про користувачів, що формують індивідуальні рекомендації та використовуються переважно на контент-орієнтованих веб-сайтах, в Інтернет-магазинах та віртуальних соціальних мережах. Користувачів важко мотивувати надавати дані про свої вподобання у явній формі, тож набагато легше їх зібрати неявно, і таких даних, що можна використати для

створення рекомендацій найбільше на веб-сайтах великих Інтернет-магазинів та популярних соціальних мереж, чим вони і користуються, переважно у маркетингових цілях.

Розглянемо основні моделі та методи роботи рекомендаційних систем, що існують на сьогоднішній день.

Після дослідження робіт [4, 6, 9-27], було виявлено, що існують наступні моделі роботи рекомендаційних систем:

– *Моделі сусідства* (Neighborhood models) – досить поширені моделі, що використовують коефіцієнти подоби між елементами системи (користувачами, об'єктами) для того, щоб формувати рекомендації на основі ступеню схожості елементів системи.

– *Матричні факторизаційні моделі* (Matrix factorization models) – також популярні моделі, які використовують для формування рекомендацій приховані фактори, які одержують шляхом факторизації матриці рейтингів та/або матриці даних профілю користувача.

– *Моделі на основі класифікації та кластеризації даних* – такі моделі розділяють дані на кластери на основі різних статистичних методів, методів кластеризації або методів машинного навчання, використовуються у системах, де є багато інформації про елементи системи, які можна представити у вигляді векторів ознак.

– *Моделі на основі знань* (Knowledge-based models) – рідковикористовувані моделі, що застосовують знання про предметну область, для якої розробляється рекомендаційна система та дозволяють формувати набори правил та базу знань, на основі яких розробляються інтелектуальні чи експертні системи.

– *Моделі, засновані на знаннях про соціальні зв'язки* (community-based models) – рідковикористовувані моделі, що застосовують знання про соціальні зв'язки користувачів для формування рекомендацій.

– *Моделі зміни вподобань користувачів у часі* – моделі, що дозволяють прогнозувати періодичні та неперіодичні зміни у вподобаннях користувачів з

часом, як правило, є доповненням до попередніх моделей, зокрема, існують матричні факторизаційні моделі, що враховують фактор часу.

Моделі робастних рекомендаційних систем – передбачають наявність підсистеми інформаційної безпеки, така підсистема може мати деякі або усі з наступних функцій: виявлення атак та ідентифікація профілів ботів, рандомізація списків рекомендацій для зашумлення даних з метою захисту приватності вподобань користувачів, виявлення та видалення зі списку рекомендацій або маркування ризикованих об'єктів. Є доповненням до попередніх моделей, будь-яку рекомендаційну систему можна доповнити підсистемою інформаційної безпеки.

Дослідження робіт [4, 6, 9-27] показало, що існує велика кількість різних методів роботи рекомендаційних систем, а саме:

– *Методи колаборативної фільтрації (collaborative filtering)* – застосовують інформацію про рейтинги, які користувачі виставляють об'єктам, та засновані на визначенні схожості користувачів чи об'єктів. Дані методи, як правило, застосовують моделі сусідства або матричні факторизаційні моделі. Можуть використовувати моделі, засновані на знаннях.

– *Методи контентної фільтрації (content-based filtering)* – застосовують описи об'єктів та дані з профілю користувача, часто засновані на комп'ютерній лінгвістиці. Використовують моделі класифікації та кластеризації даних. Можуть використовувати моделі, засновані на знаннях.

– *Методи фільтрації, заснованої на знаннях про предметну область (knowledge-based filtering)* – використовують знання про поведінку користувачів у системі, для якої розробляється рекомендаційна система, використовують базу знань для формувань рекомендацій. Використовують моделі, засновані на знаннях.

– *Методи фільтрації, заснованої на знаннях про соціальні зв'язки користувачів (community-based filtering)* – можуть застосовуватися, якщо на веб-сайті чи в додатку є елементи соціальної мережі, які дозволяють отримати

граф соціальних зв'язків користувачів, в такому разі, при побудові рекомендацій користувачу, буде враховуватися інформація про вподобання його друзів. Використовують моделі, засновані на знаннях про соціальні зв'язки.

– *Методи контекстної фільтрації, зокрема, на основі демографічної інформації (context-based filtering)* – формування рекомендацій відбувається на основі контекстної інформації, наприклад, демографічних даних про користувача. Як правило, використовуються як доповнення до інших методів, наприклад, методів колаборативної чи контентної фільтрації, зокрема, для подолання проблеми холодного старту. Можуть використовувати моделі класифікації та кластеризації даних та моделі на основі знань.

– *Гібридні методи* – поєднують у собі декілька методів та/або моделей роботи рекомендаційних систем.

Розглянемо найбільш поширені комбінації існуючих на сьогоднішній день моделей та методів синтезу рекомендаційних систем для контент-орієнтованих веб-сайтів та віртуальних соціальних мереж.

Дамо наступне формальне визначення загальній рекомендаційній системі:

Рекомендаційна система – це система, яка працює з множиною елементів $E = U \cup O$, що складається з підмножини об'єктів O та підмножини користувачів U, а саме, збирає дані про ці елементи, прогнозує вподобання (оцінки, інтерес тощо) користувачів, і на основі них формує списки рекомендацій $R = \left(r_1, r_2, ..., r_n \middle| n \le N\right)$ з одних елементів системи E_1 для інших елементів системи E_2 довжиною n, де $E_1, E_2 \subset E$. Також рекомендаційна система має множину ознак $A = \{a_1, a_2, ..., a_m\}$, які можуть бути наявними у елементів системи. Кожен елемент рекомендаційної системи має множину коефіцієнтів $K = \{k_{i,1}, k_{i,2}, ..., k_{i,q}\}$, що характеризують ступінь його приналежності до кожної з можливих ознак. А також кожен користувач системи має інтереси, представлені множиною коефіцієнтів

$K = \{k_{u,1}, k_{u,2}, ..., k_{u,q}\}$ – що характеризують ступінь його інтересу до кожної з ознак елементів системи. Елементи списків рекомендацій r_i можуть бути впорядковані за різними принципами, але найчастіше вони впорядковані за значеннями прогнозованих оцінок (або ступенем можливого інтересу) для них одержувача рекомендацій.

Використовуючи дане формальне визначення опишемо можливі способи застосування рекомендаційних систем.

1. Створення списку рекомендацій для користувача, що містить об'єкти (наприклад, рекомендація відвідувачу Інтернет-магазину товарів):

$$R = \{(u_i, O_r) | u_i \in U, O_r \subset O\}, \quad (1.1)$$

де u_i – користувач, для якого формуються рекомендації; O_r – множина рекомендованих об'єктів.

2. Створення списку рекомендацій для групи користувачів, що містить об'єкти (наприклад, рекомендація відеороликів користувачам, що переглядають одночасно одне й те саме потокове відео):

$$R = \{(U_c, O_r) | U_c \subset U, O_r \subset O\}, \quad (1.2)$$

де U_c – група користувачів, для якої формуються рекомендації; O_r – рекомендовані об'єкти.

3. Створення списку рекомендацій для користувача, що містить інших користувачів (наприклад, рекомендація друзів у соціальній мережі):

$$R = \{(u_i, U_r) | u_i \in U, U_r \subset U\}, \quad (1.3)$$

де u_i – користувач, для якого формуються рекомендації; U_r – рекомендовані користувачі.

4. Створення списку рекомендацій для групи користувачів, що містить інших користувачів (наприклад, рекомендація співтовариству у соціальній мережі підписку на інші співтовариства):

$$R = \{(U_c, U_r) | U_c, U_r \subset U\}, \quad (1.4)$$

де U_c – група користувачів, для яких формуються рекомендації; U_r –

рекомендовані групи користувачів.

А також запропонуємо ще декілька можливих способів використання рекомендаційних систем, якщо використовувати їх для пошуку та фільтрації даних:

5. Пошук користувачів, що можуть зацікавитися певним об'єктом, відранжовані по рівню можливої зацікавленості (наприклад, пошук можливих покупців для товару, аудиторії для перегляду таргетованої реклами):

$$R = \left\{ \left(o_j, U_r \right) \middle| o_j \in O, U_r \subset U \right\}, \tag{1.5}$$

де o_j – об'єкт, для якого здійснюється пошук; U_r – рекомендовані користувачі.

6. Пошук користувачів, що можуть зацікавитися певною множиною об'єктів, відранжовані по рівню можливої зацікавленості (наприклад, пошук користувачів, що зацікавляться певним блогом з деяким набором статей):

$$R = \left\{ \left(O_s, U_r \right) \middle| O_s \subset O, U_r \subset U \right\}, \tag{1.6}$$

де O_s – група об'єктів, для яких здійснюється пошук; U_r – рекомендовані користувачі.

7. Пошук об'єктів схожих на заданий (наприклад, інформаційний пошук по шаблону, відранжований по ступеню інтересу користувача до них):

$$R = \left\{ \left(u_i, o_j, O_r \right) \middle| u_i \in U; o_j \in O; O_r \subset O \right\}, \tag{1.7}$$

де u_i – користувач, який здійснює пошук; o_j – об'єкт, який являється шаблоном пошуку; O_r – рекомендовані об'єкти.

8. Формування списків об'єктів, схожих на певну групу об'єктів, на основі оцінок (відгуків) різних користувачів для них (наприклад, формування списку контенту, що відповідає певним ознакам – (не)спам, (не)деструктивний контент, тематичний контент тощо):

$$R = \left\{ \left(U_c, O_s, O_r \right) \middle| U_c \subset U; O_s, O_r \subset O \right\}, \tag{1.8}$$

де U_c – група користувачів, на основі відгуків яких формуються списки; O_s – група об'єктів, на основі яких здійснюється пошук; O_r – рекомендовані об'єкти.

Найчастіше рекомендаційні системи застосовуються для створень

рекомендацій користувачам або групам користувачів за схемами описаними у перших трьох випадках (1.1)-(1.3).

Рекомендації R користувачам системи формуються на основі даних D, зібраних про елементи системи за допомогою деякого алгоритму фільтрації даних:

$$R(u_i, D, N) = \left(r_1, r_2, ..., r_n \middle| n \le N\right), \tag{1.9}$$

$$D = \{D_1, D_2, D_3, D_4, D_5\}, \tag{1.10}$$

де D – види доступних рекомендаційній системі даних, які можна використати для формування рекомендацій: D_1 – дані, що прямо показують відношення користувачів до переглянутих елементів системи: оцінки або лайки/дизлайки, які користувачі виставляють елементам системи; D_2 – дані, які опосередковано показують відношення користувачів до переглянутих елементів системи: перегляди сторінок елементів системи, здійснені покупки, написані коментарі тощо; D_3 – дані, які дозволяють віднести користувача до певної групи користувачів з відомими інтересами: дані з профілю, зокрема, демографічні дані, контекстні дані; D_4 – властивості та опис об'єктів системи; D_5 – результати опитувань, якщо система використовує опитування.

Дані типів D_1, D_2 та D_5 по суті являються зворотним зв'язком від користувачів для рекомендаційної системи. D_3 та D_4 є описовими даними.

Довжина списку рекомендацій залежить від вимог конкретного додатку або веб-ресурсу.

Базова вимога до якості роботи рекомендаційної системи, це забезпечення мінімальної різниці між прогнозованими та реальними вподобаннями користувачів:

$$d(P, R) \to \min, \tag{1.11}$$

де вектор $R = (r_1, r_2, ..., r_n)$ містить список прогнозованих рекомендацій (оцінок) користувача, впорядкований по спаданню за величиною оцінок; вектор $P = (p_1, p_2, ..., p_n)$ містить справжні вподобання (оцінки) користувача, невідомі

системі на етапі формування списку рекомендацій.

Дані рекомендаційної системи зручно представляти у вигляді графу, де користувачі та об’єкти – вершини графу, а дії користувачів (перегляди, оцінки тощо) та результати роботи рекомендаційної системи (рекомендації, коефіцієнти подоби тощо) – ребра. Тож моделювати дані рекомендаційної системи та поведінку користувачів зручно за допомогою складних графів. Для якісного моделювання елементів та процесів рекомендаційної системи необхідно знати властивості, якими вони володіють.

Рекомендаційна система представляє собою мережу зв’язків між користувачами та об’єктами веб-ресурсу. Її можна розглядати як різновид соціального графу. *Соціальний граф* – граф, вузлами якого є соціальні об'єкти (наприклад, користувачі, співтовариства користувачів, об’єкти контенту тощо), а ребрами – соціальні зв'язки між ними.

Існує декілька підходів до моделювання соціальних мереж та соціальних графів, зокрема, найчастіше застосовують наступні [3, 14, 28-30]: моделі випадкових графів, моделі складних мереж, теоретико-ігрові моделі тощо.

1.2. Колаборативна фільтрація на основі сусідства в рекомендаційних системах

Методи колаборативної фільтрації, що використовують моделі сусідства, здійснюють прогнозування вподобань користувачів на основі ступеню сусідства між елементами рекомендаційної системи [6, 9, 16, 17, 31]. Ступінь сусідства визначається за допомогою коефіцієнтів подоби, обчислення яких здійснюється на основі різних параметрів та метрик.

Тож першим кроком методів, заснованих на колаборативній фільтрації, є обчислення *коефіцієнтів подоби* користувачів та/або об’єктів системи, обчислення яких здійснюються найчастіше на основі оцінок, які користувачі виставляють об’єктам. Крім оцінок можуть використовуватися дані про

здійснені перегляди, поставлені теги, написані коментарі тощо.

У найбільш простому випадку збираються дані про поставлені користувачами оцінки та записуються у матрицю рейтингів (рис. 1.1).

	Об’єкт 1	**Об’єкт 2**	**…**	**Об’єкт n**
Користувач 1	5	3	…	3
Користувач 2	4	–	…	2
…	…	…	…	…
Користувач m	–	4	…	4

Рис. 1.1. Матриця рейтингів

В матриці рейтингів значення є оцінками конкретного користувача для конкретного об’єкту. Відсутні значення в таблиці рейтингів є невідомими, тобто, для певного об’єкту певний користувач не виставив оцінку.

Розглянемо метрики, що можуть використовуватися для визначення коефіцієнтів подоби у колаборативній фільтрації.

Якщо об’єкт (або користувач), що описується m ознаками, представити точкою у k-мірному просторі, то подібність об'єктів один з одним буде визначатися як відстань в даному метричному просторі. У випадку з матрицею рейтингів таке представлення можливе – ознаками будуть в такому разі оцінки об’єктам.

Найбільш поширені метрики подоби, що використовуються в такому випадку: евклідова відстань (1.12), зважена евклідова відстань (1.13), відстань Хемінга (Манхеттенська відстань) (1.14), відстань Мінковського (1.15), відстань Махаланобіса (1.16), кореляція Пірсона (1.17), косинусна подоба (1.18):

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{m} (x_{1i} - x_{2i})^2}, \qquad (1.12)$$

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{m} w_i (x_{1i} - x_{2i})^2}, \tag{1.13}$$

$$d(x_1, x_2) = \sum_{i=1}^{m} \left| x_{1i} - x_{2i} \right|, \tag{1.14}$$

$$d(x_1, x_2) = \left(\sum_{i=1}^{m} \left| x_{1i} - x_{2i} \right|^p \right)^{1/p}, \tag{1.15}$$

$$d(x_1, x_2) = \sqrt{\left(\overline{X}_1 - \overline{X}_2\right)^{\delta} \sum^{-1} \left(\overline{X}_1 - \overline{X}_2\right)}, \tag{1.16}$$

$$d(x_1, x_2) = \frac{\sum_{i=1}^{m} (x_{1i} - \overline{x}_1)(x_{2i} - \overline{x}_2)}{\sqrt{\sum_{i=1}^{m} (x_{1i} - \overline{x}_1)^2} \sqrt{\sum_{i=1}^{m} (x_{2i} - \overline{x}_2)^2}}, \tag{1.17}$$

$$d(x_1, x_2) = \frac{\overline{X}_1 \cdot \overline{X}_2}{\left\|\overline{X}_1\right\| \cdot \left\|\overline{X}_2\right\|} = \frac{\sum_{i=1}^{m} x_{1i} \cdot x_{2i}}{\sqrt{\sum_{i=1}^{m} (x_{1i})^2} \sqrt{\sum_{i=1}^{m} (x_{2i})^2}}, \tag{1.18}$$

де $d(x_1, x_2)$ – відстань між об'єктами x_1 та x_2; x_{1i}, x_{2i} – значення i-ї ознаки відповідно у 1-го та 2-го об'єкту; w_i – вага, що привласнюється i-ій змінній; $\sum^{-1}$ – матриця зворотна коваріаційній матриці, розрахованій по всій вибірці; $\overline{X}_1$, $\overline{X}_2$ – вектори значень ознак у 1-го та 2-го об'єкту; $\overline{x}_1$, $\overline{x}_2$ – середні значення ознак відповідно у 1-го та 2-го об'єкту; m – кількість ознак.

Оцінювати схожість об'єктів (або користувачів) за допомогою мір відстані зручно при використанні числових ознак. Але часто зустрічаються ознаки, що виміряються в інших шкалах (наприклад, в ранговій або номінальній). І це може трапитися, якщо використовувати в колаборативній фільтрації не оцінки, а інші відомості про дії користувача. В цьому випадку всі ознаки, які використовуються для класифікації, представляються у вигляді двійкового коду. Тобто, кожен об'єкт описується вектором $\overline{X}_i = (x_{i1}, x_{i2}, ..., x_{im})$, де $i = \overline{1,n}$, кожна з компонент якого приймає значення 0 або 1. Для визначення подоби i-го та j-го об'єктів в такому випадку найбільш часто застосовують наступні метрики: коефіцієнт Рао (1.19),

коефіцієнт Хаммана (1.20), коефіцієнт Роджерса та Танімото (1.21), коефіцієнт Джекарда (1.22), коефіцієнт Дейка (1.23), коефіцієнт композиційної подоби (1.24):

$$S_{ij} = \frac{n_{ij}^{(1,1)}}{m}, (0 \le S_{ij} \le 1), \tag{1.19}$$

$$S_{ij} = \frac{p_{ij} - q_{ij}}{m}, (p_{ij} = q_{ij} \to S_{ij} = 0), \tag{1.20}$$

$$S_{ij} = \frac{n_{ij}^{(1,1)}}{n_i^{(1)} + n_j^{(1)} + n_{ij}^{(1,1)}}, (0 \le S_{ij} \le 1), \tag{1.21}$$

$$S_{ij} = \frac{n_{ij}^{(1,1)}}{n_{ij}^{(1,1)} + q_{ij}}, (0 \le S_{ij} \le 1), \tag{1.22}$$

$$S_{ij} = \frac{2n_{ij}^{(1,1)}}{2n_{ij}^{(1,1)} + q_{ij}}, (0 \le S_{ij} \le 1), \tag{1.23}$$

$$S_{ij} = \frac{p_{ij}}{2m - p_{ij}} = \frac{p_{ij}}{m - q_{ij}}, (0 \le S_{ij} \le 1), \tag{1.24}$$

де $p_{ij} = n_{ij}^{(1,1)} + n_{ij}^{(0,0)}$ – загальна кількість співпадаючих ознак; $q_{ij} = n_{ij}^{(0,1)} + n_{ij}^{(1,0)}$ – загальна кількість неспівпадаючих ознак; $n_{ij}^{(1,1)}$ – число співпадаючих одиничних ознак у обох пар об'єктів (пар (1,1)); $n_{ij}^{(0,0)}$ – число співпадаючих нульових ознак у обох пар об'єктів (пар (0,0)); $n_{ij}^{(1,0)}$ – кількість співпадаючих одиничних ознак у *i*-го та нульових ознак у *j*-го об'єктів (пар (1,0)); $n_{ij}^{(0,1)}$ – кількість співпадаючих нульових ознак у *i*-го та одиничних ознак у *j*-го об'єктів (пар (0,1)); $n_i^{(1)}$, $n_j^{(1)}$ – число одиничних ознак у *i*-го та одиничних ознак у *j*-го об'єктів відповідно; m – загальна кількість ознак, за якими здійснюється порівняння.

Методи колаборативної фільтрації на основі моделі сусідства поділяється на два види [6, 9, 20]:

1. Засновані на схожості користувачів (User/User, User-based).
2. Засновані на схожості об’єктів (Item/Item, Item-based).

Колаборативна фільтрація, заснована на схожості користувачів (user-based). Розглянемо найпростіший спосіб реалізації колаборативної фільтрації, заснованої на схожості користувачів. Всі інші способи є ускладненнями даного.

Після того, як для рекомендаційної системи зібрані дані та побудована матриця користувачів-об’єктів, для формування рекомендацій певному користувачу необхідно здійснити наступну послідовність дій:

1. Обчислити множину коефіцієнтів подоби даного користувача з усіма іншими користувачами (1.25):

$$K_i = \{k_{i,1}, k_{i,2}, \ldots, k_{i,j}\}, \tag{1.25}$$

де $k_{i,j}$ – коефіцієнт подоби між *i*-тим та *j*-тим користувачами.

2. Ранжувати користувачів відносно *i*-того користувача за ступенем подоби на нього. Обрати TopN найбільш схожих на нього користувачів або усіх користувачів, для яких вдалося обчислити коефіцієнт подоби.

3. Обчислити для об’єктів системи, які ще не переглядав (не оцінював) *i*-тий користувач, зважену суму оцінок інших користувачів (1.26):

$$S_{i,q} = \sum_{j=1}^{n} r_{q,j} \cdot k_{i,j}, \tag{1.26}$$

де $r_{q,j}$ – оцінка *q*-того об’єкту, поставлена *j*-тим користувачем; *n* – довжина списку TopN схожих на *i*-того користувача користувачів.

4. Розділити кожну зважену суму на суму коефіцієнтів подоби всіх користувачів, що ставили оцінки відповідному об’єкту (1.27), щоб об’єкти, що отримали більше оцінок не одержали перевагу:

$$\tilde{r}_{i,q} = \frac{S_{i,q}}{\sum_{j=1}^{n} k_{i,j}}. \tag{1.27}$$

Рівняння (1.27) дає прогноз оцінки *i*-того користувача *q*-тому об’єкту.

5. Відсортувати множину об’єктів, одержану після третього кроку на

основі значень прогнозованих оцінок та рекомендувати користувачу перші N об’єктів у списку.

В реальних системах для кожної рекомендації неможливо використовувати в обчисленнях дані всіх сотень тисяч чи навіть мільйонів користувачів системи, тому у формулах використовуються K найближчих сусідів – користувачів максимально схожих на користувача, якому обчислюються рекомендації (згаданий вище TopN).

Один з варіантів цієї групи методів, що має назву алгоритм GroupLens [32], замість кроків 3 та 4 розглянутих вище, для прогнозування оцінки використовує наступну формулу:

$$\tilde{r}_{i,q}=\bar{r}_i+\frac{\sum_{j=1}^{n}\left(r_{j,q}-\bar{r}_j\right)\cdot k_{i,j}}{\sum_{j=1}^{n}\left|k_{i,j}\right|}, \qquad (1.28)$$

де $\bar{r}_i$ – середня оцінка i-того користувача; $\bar{r}_j$ – середня оцінка j-того користувача.

Методи колаборативної фільтрації засновані на схожості користувачів потребують постійних перерахунків коефіцієнтів подоби, так як дані про дії користувачів постійно оновлюються і їх вподобання з часом змінюються.

Колаборативна фільтрація, заснована на схожості об’єктів (item-based). Основна ідея даних методів полягає у тому, щоб для кожного об’єкту заздалегідь визначити множину схожих на нього об’єктів. Тоді для формування рекомендацій певному користувачу достатньо буде знайти ті об’єкти, яким він поставив найбільші оцінки, та створити зважений список N об’єктів, максимально схожих на них. Результати порівняння об’єктів змінюються не так часто, як результати порівняння користувачів. Тож на першому кроці необхідно дослідити всі наявні дані, а подальші перерахунки можна робити рідко, вибираючи моменти часу, коли навантаження на веб-сайт мінімальне.

Методи колаборативної фільтрації засновані на схожості об’єктів

працюють швидше, ніж методи засновані на схожості користувачів, так як багато обчислень можна здійснити заздалегідь. Тож їх можна з успіхом використовувати на великих об’ємах даних.

Для прогнозування оцінки на основі даного підходу треба знайти зважене середнє для оцінених користувачем об’єктів:

$$\tilde{r}_{i,q}=\bar{r}_i+\frac{\sum_{p=1}^{n}\left(r_{j,p}-\bar{r}_i\right)\cdot k_{q,p}}{\sum_{p=1}^{n}\left|k_{q,p}\right|}, \qquad (1.29)$$

де $k_{q,p}$ – коефіцієнт кореляції між q-тим об’єктом та p-тим об’єктом; n – довжина списку схожих на q-тий об’єкт об’єктів.

Використовуючи методи колаборативної фільтрації засновані на схожості об’єктів можна рідше перераховувати коефіцієнти подоби, ніж в методах заснованих на схожості користувачів, адже коефіцієнти подоби для об’єктів змінюються рідше, ніж для користувачів. Також Item-based методи мають більшу стійкість до інформаційних атак, ніж User-based методи.

Методи колаборативної фільтрації на основі моделі сусідства не потребують використання складних ресурсозатратних алгоритмів, показують високу точність прогнозування вподобань для користувачів, які уже виставили певну, мінімально необхідну, кількість оцінок об’єктам системи. Чим більше оцінок користувач виставить об’єктам системи, тим точніше ці методи сформують для нього списки рекомендацій. Недоліками цих методів є вразливість до проблеми холодного старту та інформаційних атак.

1.3. Колаборативна фільтрація на основі матричної факторизації в рекомендаційних системах

Методи колаборативної фільтрації, що використовують матричні факторизаційні моделі, здійснюють прогнозування вподобань користувачів на

основі прихованих факторів елементів системи, що впливають на інтереси користувачів та можуть бути одержані за допомогою факторизації матриці рейтингів або змішаної матриці, що містить рейтинги та ознаки елементів системи [4, 6, 15, 33, 34, 36].

Приховані фактори, виявлені за допомогою факторизації, дозволяють заповнити відсутні у матриці рейтингів комірки, тобто, прогнозувати відсутні рейтинги.

Факторизація – це процес декомпозиції об’єкту (зокрема, матриці) в набір інших об’єктів (факторів), добуток яких дає початковий об’єкт [35]. Факторизація дозволяє виділити ключові компоненти об’єкту факторизації.

Основна ідея факторизації матриці рейтингів рекомендаційної системи зображена на рис. 1.2.

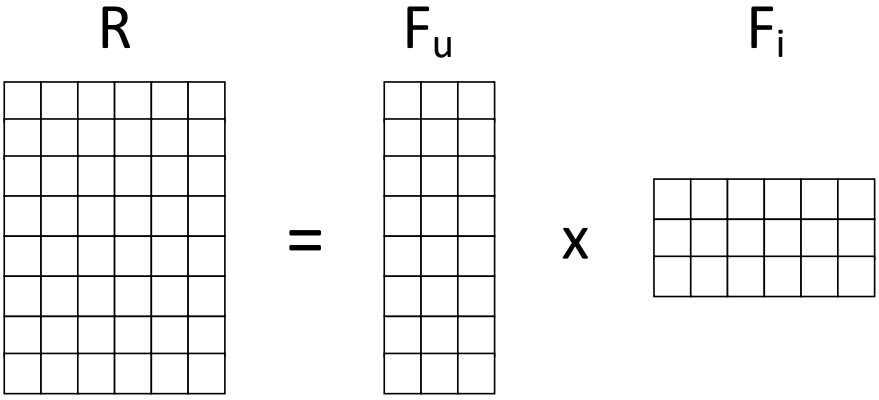


Рис. 1.2. Принцип факторизації матриці рейтингів

Матриця рейтингів R розмірності $n \times m$, де n – кількість користувачів, m – кількість об’єктів, факторизується на дві матриці: матрицю прихованих факторів користувачів F_u, розмірності $n \times k$, де k – кількість прихованих факторів, та матрицю прихованих факторів об’єктів F_i, розмірності $k \times m$.

Загальний алгоритм факторизації матриці рейтингів виглядає наступним чином:

1. Ініціалізуємо матриці F_u та F_i випадковими значеннями.

2. Перемножуємо матриці F_u та F_i і порівнюємо результат з R, обчислюємо помилки для кожної комірки та загальну помилку.

3. Мінімізуємо помилки за допомогою деякого алгоритму машинного навчання (наприклад, градієнтного спуску, методу найменших квадратів тощо), поки загальна помилка не знизиться до прийнятного значення.

Найбільш відомими методами колаборативної фільтрації, що застосовують матричну факторизаційну модель є FunkSVD, SVD++, Asymmetric SVD, timeSVD [4, 6, 15, 33, 36-38]. Усі ці моделі одержали назву від методу факторизації матриць Singular value decomposition (сингулярний розклад матриць), хоча безпосередньо його вони не використовують, а лише засновані на спільній з ним ідеї – одержати для певної матриці деякі матриці, добуток яких дасть матрицю наближену до початкової. У випадку з матрицями рейтингів, ця одержана наближена матриця буде містити наближені дані у відомих рейтингах, а в комірках, де в початковій матриці рейтинги були невідомі, з'являться прогнозовані рейтинги.

Перший метод з цієї групи методів був запропонований Сімоном Фанком під час конкурсу від Netflix у 2006. У своїй публікації у власному блозі [36], Фанк визначив матрицю рейтингів як добуток двох матриць пониженого рангу, перша – рядки прихованих факторів для користувачів, а друга – стовпчики прихованих факторів для об'єктів. Множення рядка користувача на стовпчик об'єкта дає прогнозований рейтинг для відповідної пари користувач-об'єкт.

Усі наступні методи, засновані на матричних факторизаційнійних моделях є покращеними модифікаціями моделі FunkSVD. Розглянемо всі ці методи детальніше.

FunkSVD. Найперший метод роботи рекомендаційних систем, що застосовує матричну факторизацію [36]. Даний метод полягає у наступному. Спочатку треба визначити базові предиктори (зміщення) $b_{u,i}$, які складаються з базових предикторів окремих користувачів b_u і базових предикторів окремих

об’єктів b_i, а також загального середнього рейтингу об’єктів у системі μ:

$$b_{u,i} = \mu + b_u + b_i. \tag{1.30}$$

Прогнозування оцінки для пари користувач-об’єкт здійснюється за наступною формулою:

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i \cdot p_u. \tag{1.31}$$

де q_i – вектор факторів об’єкту i, а p_u – вектор факторів користувача u.

На початку роботи алгоритму треба обчислити глобальну середню оцінку та усі предиктори. Потім треба знайти найкращі предиктори та фактори, що дозволяють прогнозувати рейтинги з найменшою помилкою.

Для визначення помилки використовується сума квадратів відхилень:

$$E = \sum_{(u,i)\in D} (r_{u,i} - \hat{r}_{u,i})^2, \tag{1.32}$$

$$E = \sum_{(u,i)\in D} (r_{u,i} - \mu - b_u - b_i - q_i \cdot p_u)^2, \tag{1.33}$$

де $r_{u,i}$ – справжній рейтинг об’єкту i у користувача u; $\hat{r}_{u,i}$ – прогнозований рейтинг.

Дана функція оптимізується градієнтним спуском, беруться часткові похідні по кожному аргументу, а рух під час градієнтного спуску відбувається у сторону, зворотню напрямку цих похідних. Для одержання адекватних результатів при роботі з реальними даними необхідно враховувати ймовірність оверфітінгу [39-41] (перенавчання системи) та виконувати регуляризацію [39, 42], щоб подолати дану проблему.

Регуляризація – додавання деякої додаткової інформації, щоб знайти рішення некоректно поставленої задачі, або щоб уникнути перенавчання. Загалом регуляризуючий вираз $R(f)$ додається до значення помилки, перед тим як визначати аргумент, що дає найменше значення помилки:

$$f_* = \arg\min_f \sum_{i=1}^{n} E(f(\hat{x}_i), \hat{y}_i) + \lambda \cdot R(f), \tag{1.34}$$

де E – функція, що визначає похибку передбачення $f(x)$ для значень y, а

параметр λ визначає важливість доданка для регуляризації. Зазвичай $R(f)$ визначається як штраф за складність функції f. Зокрема, поняття складності включає обмеження на гладкість та на норму векторного простору.

В FunkSVD оптимізаційний вираз можна записати наступним чином:

$$b_*, q_*, p_* = \arg\min_{b,q,p} \sum_{(u,i)}^{n} (r_{u,i} - \mu - b_u - b_i - q_i \cdot p_u)^2 + \\ + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 + \|q_i\|^2 + \|p_u\|^2 \right), \quad (1.35)$$

де λ – параметр регуляризації.

Якщо взяти від (1.35) часткові похідні по кожній із змінних, що оптимізуються, отримаємо прості правила для градієнтного спуску.

Під час градієнтного спуску у FunkSVD використовуються наступні правила для оптимізації змінних, що впливають на результат:

$$b_u = b_u + \gamma(e_{u,i} - \lambda b_u), \quad (1.36)$$

$$b_i = b_i + \gamma(e_{u,i} - \lambda b_i), \quad (1.37)$$

$$q_{i,k} = q_{i,k} + \gamma(e_{u,i} \cdot p_{u,k} - \lambda q_{i,k}), \quad (1.38)$$

$$p_{u,k} = p_{u,k} + \gamma(e_{u,i} \cdot q_{i,k} - \lambda p_{u,k}), \quad (1.39)$$

де $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$ – помилка на навчальному наборі; γ – швидкість навчання.

Можна очікувати більшої точності, виділивши окремі швидкості навчання γ_n і регуляризації λ_n для кожного типу досліджуваного параметра. Так, наприклад, рекомендується використовувати різні швидкості навчання для зсувів користувачів, зсувів об’єктів і самих факторів.

Важливо, що при такому підході невідомо, які саме характеристики об’єктів відповідають факторам. Тому дані моделі є неінтерпретуємими.

SVD++. Відрізняється від FunkSVD тим, що крім рейтингів (явного зворотного зв’язку від користувача) використовує також неявну інформацію про вподобання користувачів, наприклад, перегляди об’єктів, написання коментарів тощо [6, 4, 15].

Точність прогнозування у SVD++ поліпшується за рахунок врахування

неявного зворотного зв'язку, який забезпечує додаткову індикацію вподобань користувачів. Це особливо корисно для тих користувачів, які надали більше неявного зворотного зв’язку, ніж явного.

Для врахування неявного зворотного зв’язку (одного типу) від користувачів використовується другий набір факторів об’єктів, що пов’язує кожен об’єкт з вектором факторів $y_i \in R(u)$. Ці нові фактори об’єктів використовуються для характеристики користувачів на основі набору об’єктів, які вони неявно оцінили. У такій моделі рейтинги прогнозуються наступним чином:

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i \cdot \left(p_u + \left|R(u)\right|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right). \tag{1.40}$$

Набір $R(u)$ містить усі об’єкти, що були неявно оцінені користувачем u.

Характеристики користувача u моделюються як $p_u + \left|R(u)\right|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j$. Оскільки y_j – центровані навколо нуля (регуляризацією), сума нормалізується на $\left|R(u)\right|^{-\frac{1}{2}}$, щоб стабілізувати її дисперсію в межах діапазону спостережуваних значень $\left|R(u)\right|$.

Параметри моделі визначаються шляхом мінімізації відповідної регуляризованої квадратичної функції помилок за допомогою стохастичного градієнтного спуску. Оптимізація змінних, що впливають на результат прогнозу, обчислюється наступним чином:

$$b_u = b_u + \gamma\left(e_{u,i} - \lambda_1 b_u\right), \tag{1.41}$$

$$b_i = b_i + \gamma\left(e_{u,i} - \lambda_1 b_i\right), \tag{1.42}$$

$$q_{i,k} = q_{i,k} + \gamma\left(e_{u,i} \cdot \left(p_{u,k} + \left|R(u)\right|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right) - \lambda_2 q_{i,k} \right), \tag{1.43}$$

$$p_{u,k} = p_{u,k} + \gamma\left(e_{u,i} \cdot q_{i,k} - \lambda_2 p_{u,k}\right), \tag{1.44}$$

$$\forall j \in R(u) : y_j \leftarrow y_j + \gamma(e_{u,i} \cdot \left|R(u)\right|^{-\frac{1}{2}} q_{i,k} - \lambda_2 y_j), \tag{1.45}$$

Можна враховувати і декілька типів неявного зворотного зв'язку. Наприклад, якщо користувач u має два типи неявного оцінювання об'єктів: додавання в обране $N_1(u)$ та перегляд сторінки $N_2(u)$, тоді рейтинги можна прогнозувати наступним чином:

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i \cdot \cdot \left(p_u + \left|N_1(u)\right|^{-\frac{1}{2}} \sum_{j \in N_1(u)} y_{1j} + \left|N_2(u)\right|^{-\frac{1}{2}} \sum_{j \in N_2(u)} y_{2j} \right). \quad (1.46)$$

Відносна важливість кожного джерела неявного зворотного зв'язку буде автоматично визначена алгоритмом шляхом встановлення відповідних значень параметрів моделі.

Asymmetric SVD. Асиметричний SVD дозволяє додавати до моделі нових користувачів з декількома рейтингами, без необхідності перенавчати всю модель [6, 33, 38]. При додаванні нового користувача, його приховані фактори обчислюються наступним чином:

$$p_{u,k} = \sigma\left(b + \sum_{j \in R(u)} w_{r_{uj} s_j} \right), \quad (1.47)$$

де σ – сигмоїдна функція: $\sigma(x) = \frac{1}{1+e^{-x}}$; r_{uj} – рейтинг, який користувач u поставив об'єкту j; $R(u)$ – множина об'єктів, оцінених користувачем u, з відомими рейтингами; w, b, s – параметри, що шукаються градієнтним спуском.

Якщо необхідно використовувати неявні зворотні зв'язки, тоді для Asymmetric SVD можна застосувати наступну формулу:

$$p_{u,k} = \left|R(u)\right|^{-\frac{1}{2}} \sum_{j \in R(u)} r'_{u,j} \cdot s_j, \quad (1.48)$$

де $r'_{u,j} = r_{i,j} - (\mu + b_u + b_i)$; r_{uj} – рейтинг, який користувач u поставив об'єкту j; $R(u)$ – множина об'єктів, оцінених користувачем u, з відомими рейтингами; μ, b, s – параметри, що шукаються градієнтним спуском.

TimeSVD++. Це факторизаційна модель з врахуванням часу [6, 37]. Вподобання користувачів можуть залежати від часу, саме це враховує дана

модель. Зміни можуть мати як циклічний, так і не циклічний характер.

Популярність товару може змінюватися з часом. Це можна врахувати, трактуючи зміщення об'єкту b_i як функцію часу. Користувачі з часом можуть змінювати свої вподобання. Це можна також врахувати, прийнявши зміщення користувача b_u як функцію часу. Тоді базові предиктори будуть розраховуватися наступним чином:

$$b_{u,i} = \mu + b_u(t_{u,i}) + b_i(t_{u,i}). \quad (1.49)$$

Тут, $b_u(t_{u,i})$ та $b_i(t_{u,i})$ – реальні функції, які змінюються з часом. Точний спосіб побудови цих функцій повинен відображати розумний спосіб параметризації задіяних часових змін. Наприклад, у випадку рейтингу фільму очікується, що прихильність до фільму щодня трохи коливатиметься, а сильно змінюватиметься протягом більш тривалих періодів. З іншого боку вподобання користувачів можуть змінюватися щодня, відображаючи природню для поведінки клієнтів мінливість. Це вимагає обрання менших проміжків часу при моделюванні поведінки користувача та більших проміжків часу для моделювання часових ефектів, пов'язаних з об'єктами.

Базові предиктори окремих об'єктів можна визначити за наступною формулою:

$$b_i(t) = b_i + b_{i,Bin(t)}, \quad (1.50)$$

де b_i – незмінна у часі частина предиктору об'єкта; $b_{i,Bin(t)}$ – частина предиктору об'єкта, що змінюється у часі, $bin(t)$ – номер часового проміжку, на які поділені усі наявні для навчання системи дані.

Для параметризації часової поведінки користувача з різною складністю та точністю можна розглядати різні функції. Простий спосіб моделювання використовує лінійну функцію для фіксації можливого поступового зміщення вподобань користувачів. Для кожного користувача *u* позначаємо середню дату рейтингу за допомогою t_u. Тепер, якщо користувач оцінив фільм у момент часу *t*, то пов'язане з цим відхилення часу визначається як:

$$dev_u(t) = sign(t - t_u) \cdot |t - t_u|^{\beta}, \tag{1.51}$$

де $|t - t_u|$ вимірює кількість умовних часових одиниць (наприклад, днів) між датами t і t_u. Значення β встановлюється шляхом перехресної перевірки.

Базові предиктори окремих користувачів можна визначити за допомогою формули (1.52) або (1.53).

$$b_i(t) = b_u + \alpha_u \cdot dev_u(t), \tag{1.52}$$

де α_u – параметр алгоритму, який слід визначати під час градієнтного спуску.

Ця проста лінійна модель для наближення поведінки користувача вимагає визначення двох параметрів b_u та α_u для кожного користувача.

Більш гнучку параметризацію пропонують сплайни. Нехай u є користувачем, пов'язаним з n_u рейтингами. Часові точки k_u розподілені рівномірно по датах рейтингів користувача u як ядра, які керують такою функцією:

$$b_i(t) = b_u + \frac{\sum_{l=1}^{k_u} e^{-\sigma|t-t_l^u|} b_{t_l}^u}{\sum_{l=1}^{k_u} e^{-\sigma|t-t_l^u|}}, \tag{1.53}$$

де параметри b_u та t_l асоціюються з контрольними точками (ядрами) і автоматично дізнаються з даних. Таким чином, вподобання користувача формується як зважена в часі комбінація цих параметрів. Кількість ядер k_u врівноважує гнучкість та ефективність обчислень. Постійна σ визначає плавність сплайну.

При використанні формул (1.51-1.53) для визначення базових предикторів, одержимо наступну формулу для запису оптимізаційного виразу:

$$b_{u*}, b_{u,t*}, b_{i*}, b_{i,Bin(t)*}, \alpha_{u*} =$$

$$= \arg\min \sum_{(u,i,t)} \left(r_{u,i} - \mu - b_u - \alpha_u dev_u(t_{u,i}) - b_{u,t} - b_i - b_{i,Bin(t)}\right)^2 + , \tag{1.54}$$

$$+ \lambda\left(b_u^2 + \alpha_u^2 + b_{u,t}^2 + b_i^2 + b_{i,Bin(t)}^2\right)$$

Можна використовувати ту саму методологію для отримання більшої

кількості часових ефектів. Наприклад, для фіксації періодичних ефектів. Деякі об'єкти можуть бути більш популярними в конкретні пори року або впродовж певних свят. Періодичні ефекти можна знайти і для користувача. Наприклад, користувач може мати різні схеми купівлі протягом вихідних у порівнянні з робочим тижнем. Спосіб моделювання таких періодичних ефектів – виділити параметр для комбінацій періодів часу з об'єктами або користувачами. В такому разі базові предиктори можна розраховувати за наступними формулами:

$$b_i(t) = b_i + b_{i,Bin(t)} + b_{i,period(t)}, \tag{1.55}$$

$$b_u(t) = b_u + \alpha_u \cdot dev_u(t) + b_{u,t} + b_{u,period(t)}. \tag{1.56}$$

Приховані фактори користувачів для timeSVD без врахування періодичності, а лише з врахуванням зміщень у часі, можна визначити так:

$$p_{u,k}(t) = p_{u,k} + \alpha_{u,k} \cdot dev_u(t) + p_{u,k,t}, \ k = 1,..., f. \tag{1.57}$$

Рейтинги у TimeSVD++ можна прогнозувати за наступною формулою:

$$\hat{r}_{u,i} = \mu + b_u(t) + b_i(t) + q_i \cdot \left(p_u(t) + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right). \tag{1.58}$$

Отже, існує багато різних моделей поведінки користувача рекомендаційної системи, заснованих на матричній факторизації для визначення прихованих факторів, що впливають на його вподобання.

Перевагами таких моделей є висока стійкість до інформаційних атак та висока точність прогнозування вподобань. До недоліків слід віднести погану масштабованість, довгий час навчання, а також необхідність перенавчання рекомендаційної системи при появі нових даних, цей останній недолік частково вирішено у Asymmetric SVD.

1.4. Контентна фільтрація в рекомендаційних системах

Такі моделі передбачають, що вподобання користувачів визначаються властивостями об'єктів, які вони переглядали раніше [6, 9]. Рекомендаційні

системи з фільтрацією на основі змісту (контентною фільтрацією) беруть до уваги схожість об'єктів з інформацією відомою про користувача.

Інформація про користувача може бути отримана з його профілю та/або зібрана з його дій на веб-сайті – написаних відгуків та коментарів, придбаних товарів, переглянутих веб-сторінок тощо. Така інформація може бути зібрана як явними способами (дані з профілю користувачів та сторінок опису об’єктів), так і неявними способами (вилучення ключових слів та фраз [43, 44, 45], а також їх емоційного забарвлення [46-50] з коментарів та постів користувачів).

Для створення рекомендацій такі системи аналізують інформацію про користувача, формують ключові слова про його інтереси та вподобання. Також формуються ключові слова для об’єктів системи з їх описів, отриманих тегів тощо. Рекомендуватися будуть об’єкти з бази даних, вибрані на основі визначених ключових слів. Тому таку фільтрацію ще можна назвати фільтрацією по ключовим словам.

Методи контентної фільтрації застосовують методи кластеризації, статистичні методи та методи машинного навчання, наприклад, [6, 9]:

1. Класифікатори на основі Байєсівських мереж.
2. Класифікатори на основі нейронних мереж.
3. Класифікатори на основі дерев рішень.
4. Класифікатори на основі алгоритмів кластеризації.

Класифікатори на основі Байєсівських мереж. Одним з найвідоміших класифікаторів є наївний байєсівський класифікатор [9, 40, 51]. В його основі лежить ймовірнісна модель теореми Байєса. Для роботи алгоритмів з використанням даного класифікатора необхідно створити модель Байєса для кожного користувача, який оцінював будь-які об'єкти, на основі ознак цих об'єктів (для фільмів це можуть бути актори або жанри, для новин – ключові слова тощо). Для знаходження найбільш ймовірної категорії необхідно обчислити умовні ймовірності приналежності будь-якого об’єкту до кожної категорії і вибрати ту, яка має найбільшу ймовірність (1.59):

$$Pr(C|O) = \frac{Pr(O|C) \cdot Pr(C)}{Pr(O)}, \tag{1.59}$$

$$Pr(O|C) = \prod_{i=1}^{n} Pr(W_i|C), \tag{1.60}$$

де O – об'єкт (товар, послуга тощо); C – категорія; W – слово (ознака); n – кількість слів (ознак); $Pr(C)$ – повна ймовірність того, що випадково обраний об'єкт потрапляє у категорію C; $Pr(O)$ – повна ймовірність появи об'єкту O; $Pr(W_i|C)$ – умовна ймовірність того, що слово W_i наявне в описі об'єкту O, якщо об'єкт O відноситься до категорії C.

Для кожної категорії можна задати порогові значення. Тоді, щоб новий об'єкт був віднесений до деякої категорії, ймовірність його віднесення до цієї категорії повинна бути більша ймовірності його потрапляння в будь-яку іншу категорію на величину порогового значення.

Перевагою байєсівських класифікаторів є можливість навчання, простота алгоритму навчання, можливість перегляду даних про важливість ознак та одержану в процесі навчання. Основний недолік – неможливість враховувати залежність результату від поєднань ознак.

Класифікатори на основі нейронних мереж. Штучні нейронні мережі можуть мати різну архітектуру та різні алгоритми навчання, але всі вони складаються з нейронів [9, 40, 51], загальна структурна схема яких наведена на рис. 1.3.

На рис. 1.3 використані наступні позначення: x_1 … x_n – входи нейрона (синапси); w_1 … w_n – вагові коефіцієнти входів; S – зважена сума входів нейрона; $F(S)$ – функція активації нейрона; T – порогове значення (значення, після якого нейрон переходить у стан збудження), є не у всіх типів штучних нейронів; Y – вихід нейрона (аксон).

Зважена сума S обчислюється за наступною формулою:

$$S = \sum_{i=1}^{n} x_i \cdot w_i \ , \tag{1.61}$$

Функція активації *F(S)* – визначає залежність сигналу на виході нейрона від зваженої суми сигналів на його входах. В якості функції активації можуть використовуватися: лінійна функція, порогова функція, сигмоїдальна функція тощо.

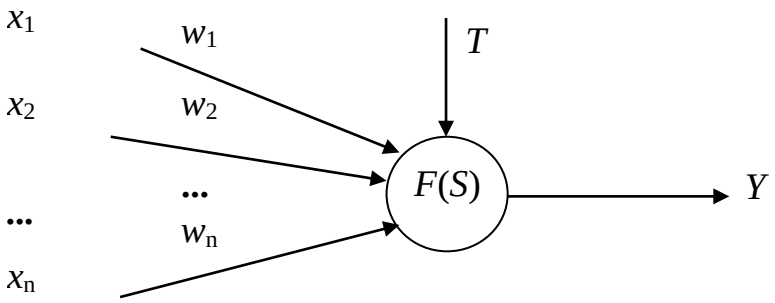


Рис. 1.3. Загальна структурна схема нейрона
у штучній нейронній мережі [40, 51]

В найпростіших випадках для контентної фільтрації можна застосовувати багатошаровий перцептрон, якщо є дані для попереднього навчання і мережу Кохонена, якщо доведеться вчити мережу в процесі використання. Розглянемо загальну структуру перцептрону для використання у контентній фільтрації (рис. 1.4).

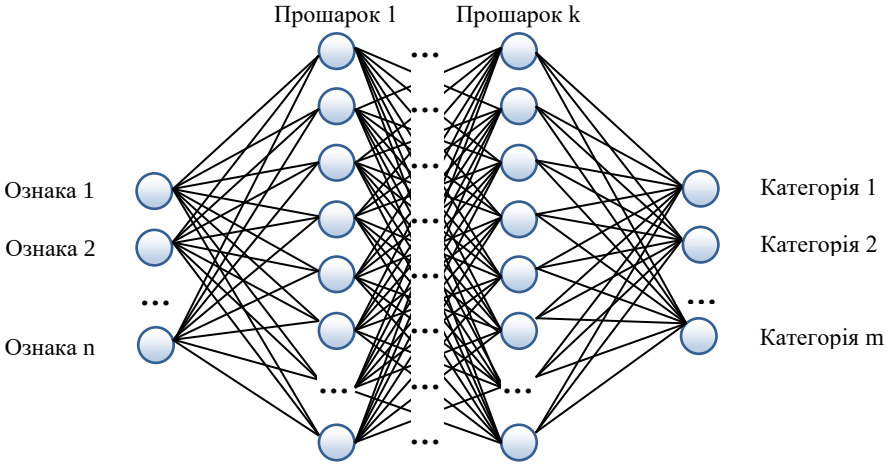


Рис. 1.4. Загальна схема нейронної мережі для класифікації об’єктів у рекомендаційній системі [9]

Входами нейромережі будуть коди ознак об’єктів, а виходами – коди категорій об’єктів. Нейромережа буде містити декілька прихованих прошарків, кількість прошарків та нейронів у них зазвичай визначаються експериментальним шляхом. Загальною є рекомендація, що кількість нейронів у прихованих прошарках має бути більшою за кількість нейронів у вхідному прошарку.

Зазвичай при побудові нейронної мережі всі вузли створюються заздалегідь. У контентній фільтрації для рекомендаційних систем в [9] пропонується новий прихований вузол створювати тоді, коли зустрічається нова комбінація ознак та створювати для нього зв'язки з вагами за замовчуванням.

Нейронні мережі здатні справлятися зі складними нелінійними функціями та знаходити залежності між різними вхідними даними. Вони допускають адаптивне навчання в процесі використання. Основні недоліки нейронних мереж – вони працюють як чорний ящик, відсутні тверді правила по вибору структури та розміру мережі, швидкості навчання.

Класифікатори на основі дерев рішень. Дерево рішень – це модель, яка являє собою сукупність правил для прийняття рішення [9, 51, 52], в даному разі – рішення, до якої категорії віднести об’єкт. Графічно таку модель можна уявити у вигляді дерева, де вузли – умови переходу, а листки – назви категорій. Якщо для даного об’єкту умова у вузлі істина то здійснюється перехід по лівому ребру, якщо ж ні, то по правому. Залежно від рішення, прийнятого у вузлах, об’єкт відноситься до певної категорії.

Метод дерев рішень реалізує *принцип рекурсивного поділу*. Ця стратегія також називається «розділяй і володарюй». У вузлах, починаючи з кореневого, вибирається ознака, значення якої використовується для розбиття всіх даних на 2 класи. Процес триває до тих пір, поки не виконається критерій зупинки. Це можливо в таких ситуаціях:

– Всі (або майже всі) дані даного вузла належать одному і тому ж класу.

– Не залишилося ознак, за якими можна побудувати нове розбиття.

– Дерево перевищило заздалегідь заданий «ліміт зростання» (якщо ліміт було встановлено).

Існують різні чисельні алгоритми побудови дерев рішень. Одним з найбільш відомих є алгоритм під назвою C5.0 [53], розроблений програмістом Джоном Квінланом. Фактично алгоритм C5.0 є стандартом процедури побудови дерев рішень. Його програмна реалізація поширюється на комерційній основі, але версія, вбудована в мову програмування для статистичної обробки даних R, доступна безкоштовно.

Дерева рішень корисні не тільки для класифікації, а також і для інтерпретації результатів. На відміну від байєсівського класифікатору вони легко справляються з взаємозалежними ознаками. Але дерева рішень не підтримують адаптивне навчання в процесі використання.

Класифікатори на основі алгоритмів кластеризації. Кластеризація (кластерний аналіз) – це задача розбиття множини об'єктів на групи, які називаються кластерами [9, 51]. Методи кластерного аналізу діляться на: ієрархічні та ітеративні. Ієрархічні в свою чергу поділяються на агломеративні і дивізимні.

Ієрархічні агломеративні методи послідовно об'єднують окремі об'єкти в кластери. Ієрархічні дивізимні методи кластеризації полягають, навпаки, у виділенні в окремий кластер об’єктів, що мають найменші показники схожості, при тому, що спочатку всі об’єкти розглядається як окремий кластер. Перевагами ієрархічних методів кластеризації є їх наочність і можливість отримати детальне уявлення про структуру даних. Ієрархічні алгоритми пов’язані з побудовою дендрограм, які описують близькість окремих точок і кластерів один до одного та представляють в графічному вигляді послідовність об’єднання (розділення) кластерів. Недоліки ієрархічних методів кластеризації – обмеження об’єму набору даних; вибір міри близькості; негнучкість отриманих класифікацій. Ієрархічні методи використовуються при невеликих

об’ємах наборів даних. При великій кількості даних вони не придатні. У таких випадках використовують ітеративні методи.

Ітеративні методи – методи кластеризації, в яких кластери формуються виходячи з умов розбиття, які можуть бути змінені користувачем для досягнення бажаної цілі. Ці методи можуть призвести до утворення перетину кластерів, коли один об’єкт може одночасно належати декільком кластерам. В процесі поділу нові кластери формуються до тих пір, поки не буде виконано правило зупинки.

Існує два підходи до розділення набору даних на певну кількість окремих кластерів.

Перший полягає у визначенні меж кластерів як найбільш щільних ділянок в багатовимірному просторі початкових даних, тобто, визначення кластера там, де є велике "згущення" точок.

Другий підхід полягає в мінімізації міри відмінності об’єктів. Найбільш поширений метод ітеративної кластеризації – *метод k-середніх* [9].

Ітеративні методи можна використовувати для великих об’ємів даних, також вони виявляють вищу стійкість по відношенню до шумів і викидів, некоректного вибору метрики, включення незначущих змінних в набір, що беруть участь в кластеризації. Недоліком ітеративних методів є те, що треба заздалегідь визначити кількість кластерів.

Якщо невідоме число кластерів, треба використовувати ієрархічні алгоритми або декілька разів використати ітеративні методи з різною кількістю кластерів.

Перевагою методів контентної фільтрації є те, що для початку роботи рекомендаційної системи не потрібно великої кількості зареєстрованих користувачів. Головним недоліком даного підходу є неможливість системи рекомендувати нові об’єкти, які не прив’язані до інтересів користувачів. При цьому виникає проблема підтримки зворотного зв’язку з користувачем.

1.5. Гібридні рекомендаційні системи

Переважна більшість реальних рекомендаційних систем є складними гібридами різних методів фільтрації даних. Існують різні способи об'єднання декількох різних методів в гібридний метод. Виділяють такі основні стратегії гібридизації рекомендаційних систем [11, 13]:

1. Зважена (Weighted). Різні рекомендаційні системи працюють незалежно, а результати їх роботи об'єднуються за допомогою зваженої суми:

$$R=\sum_{i=1}^{n} w_i \cdot A_i \ , \tag{1.62}$$

де w_i – вага алгоритму A_i.

Правильно підібрані ваги та алгоритми дозволяють значно поліпшити якість рекомендацій. Множину ваг бажано визначати за допомогою деякої функції:

$$W=\{w_1, w_2,..., w_n\}= f\left(p_1, p_2,..., p_m\right) \ , \tag{1.63}$$

де $p_1, p_2,..., p_m$ – набір деяких параметрів. Це дозволяє зробити алгоритм більш адаптивним і досягти більшої точності.

2. З перемиканням (Switching). На основі критерію для перемикання обирається яку рекомендаційну систему використати для поточного випадку. Наприклад, для нового об'єкту, у якого немає оцінок, можна обрати контентну фільтрацію, а для об'єкту, якому вже виставлено багато оцінок – колаборативну.

3. Змішана (Mixed). В список рекомендацій потрапляє об'єднання рекомендацій, побудованих різними рекомендаційними системами.

4. З комбінацією ознак (Feature Combination). Дані одержані за допомогою колаборативної фільтрації виступають у якості додаткових ознак у контентній фільтрації.

5. Каскадна (Cascade). Спочатку використовується більш швидка рекомендаційна система, потім більш повільна, але більш точна. Рекомендації

одержані першою рекомендаційною системою ранжуються, та вибираються перші n найкращих результатів, які знову ранжуються (уточнюється) за допомогою другої рекомендаційної системи. В список рекомендацій потрапляють k $(k \leq n)$ об'єктів, що мають максимальні прогнози оцінок за результатами роботи другої рекомендаційної системи.

6. Зі збільшенням числа ознак (Feature Augmentation). Вихідні дані від однієї або декількох рекомендаційних систем використовуються як вхідні ознаки для іншої системи. Таким чином можна на перший рівень обчислень додавати різні рекомендаційні системи, що працюють з різним набором ознак. Така система дозволяє гнучко додавати нові вхідні дані.

7. З мета-навчанням (Meta-level). Мета-навчання передбачає, що рекомендаційна система буде навчатися на вибірці, ознаками якої є результати роботи інших рекомендаційних систем.

Підсумовуючи проведене дослідження видів рекомендаційних систем наведемо їх загальну класифікацію на рис. 1.5.

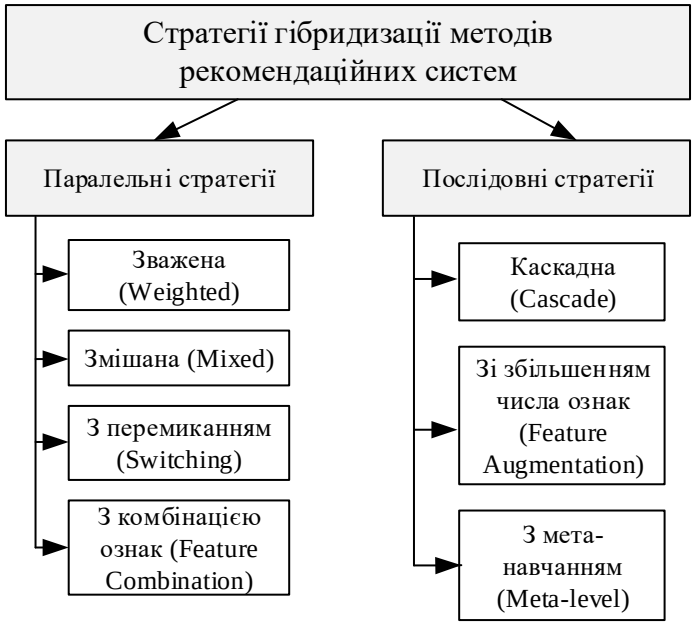


Рис. 1.5. Класифікація стратегій гібридизації методів роботи рекомендаційних систем

Вибір тієї або іншої стратегії гібридизації методів рекомендаційної системи залежить від потреб та вимог конкретного веб-сайту чи додатку, а також особливостей роботи самих методів фільтрації даних. Тому у кожній окремій ситуації стратегія гібридизації рекомендаційної системи має обиратися індивідуально, а правильність її вибору перевірятися експериментальним шляхом.

Контрольні питання

1. Що таке рекомендаційна система? Де використовуються такі системи?
2. Які існують способи застосування рекомендаційних систем?
3. Які бувають моделі та методи роботи рекомендаційних систем?
4. Які принципи роботи колаборативної фільтрації на основі сусідства?
5. Які метрики використовуються для визначення коефіцієнтів подоби у колаборативній фільтрації на основі сусідства?
6. Чим відрізняється колаборативна фільтрація Item-based від User-based? Які її переваги та недоліки?
7. Які принципи роботи колаборативної фільтрації на основі матричної факторизації?
8. У чому полягає суть методу FunkSVD? Які покращення цього методу існують? У чому полягає метод SVD і як він пов’язаний з FunkSVD?
9. Які існують методи контентної фільтрації у рекомендаційних системах?
10. Які існують способи гібридизації рекомендаційних систем?

Список літератури до розділу

1. Artemenko O., Kunanets O., Pasichnyk V. E-tourism recommender systems: a survey and development perspectives. Econtechmod an international quarterly journal, Vol. 6. No. 2. – 2017. – P. 91-95.

2. Bernardi L., Kamps J., Kiseleva J., Mueller M.J.I. The Continuous Cold Start Problem in e-Commerce Recommender Systems. – 2015. – 6 p. – [Electronic resource] – Access mode: https://arxiv.org/abs/1508.01177

3. Мелешко Є. В. Методологія забезпечення стійкості рекомендаційних систем до дестабілізуючих факторів у комп'ютерних мережах: дис. ... докт. техн. наук з комп'ютерних систем та компонентів: 05.13.05 – Комп'ютерні системи та компоненти / Є. В. Мелешко; Черкаський держ. технологічний ун-т. – Черкаси, 2020. – 323 с.

4. Cao J., Hu H., Luo T., Wang J., Huang M., Wang K., Wu Z., Zhang X. Distributed Design and Implementation of SVD++ Algorithm for E-commerce Personalized Recommender System // Embedded System Technology. ESTC 2015. Communications in Computer and Information Science, Vol. 572. Springer, Singapore. – 2015. – P. 30-44. – [Electronic resource] – Access mode: http://doi-org-443.webvpn.fjmu.edu.cn/10.1007/978-981-10-0421-6_4

5. Valois B.Jr.C., Oliveira M.A. Recommender systems in social networks // JISTEM J.Inf.Syst. Technol. Manag., Vol.8 No.3. – 2011. – P. 681-716. – [Electronic resource] – Access mode: https://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752011000300009

6. Ricci F., Rokach L., Shapira B., Kantor P.B. (Editors) Recommender Systems Handbook // Boston: Springer. – 2011. – 842 p. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-0-387-85820-3

7. Zhang Y.C., Séaghdha D.Ó., Quercia D., Jambor T. Auralist: Introducing Serendipity into Music Recommendation // WSDM '12: Proceedings of the fifth ACM international conference on Web search and data mining. – 2012. – P. 13-22. – [Electronic resource] – Access mode: https://doi.org/10.1145/2124295.2124300

8. Liu N.N., Zhao M., Xiang E.W., Yang Q. Online evolutionary collaborative filtering // In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10), Association for Computing Machinery, New York, NY, USA. – 2010. – P. 95-102.

9. Segaran T. Programming Collective Intelligence: Building Smart Web 2.0 Applications – O'Reilly. – 2007. – 362 c.

10. Singh K., Pramanik P. K. D., Dey A. K., Choudhury P. Recommender systems: an overview, research trends, and future directions // International Journal of Business and Systems Research, Vol. 15, No. 1. – 2020. – P. 14-52.

11. Burke R. Hybrid Web Recommender Systems // The Adaptive Web. Lecture Notes in Computer Science, Vol. 4321. Springer, Berlin, Heidelberg. – 2007. – C. 377-408.

12. Campos P.G., Díez F., Cantador I. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols // User Model User-Adap Inter 24. – 2014. – P. 67-119. – [Electronic resource] – Access mode: https://doi.org/10.1007/s11257-012-9136-x

13. Çano E., Morisio M. Hybrid recommender systems: A systematic literature review // Journal Intelligent Data Analysis, Vol. 21, No. 6. – 2017. – P. 1487-1524. – [Electronic resource] – Access mode: URL: https://arxiv.org/abs/1901.03888

14. He J., Chu W.W. A Social Network-Based Recommender System (SNRS) // In: Memon N., Xu J., Hicks D., Chen H. (eds) Data Mining for Social Network Data. Annals of Information Systems, vol 12. Springer, Boston, MA. – 2010. – 32 p. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-1-4419-6287-4_4

15. Jia Y., Zhang C., Lu Q., Wang P. Users' brands preference based on SVD++ in recommender systems // IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA). – 2014. – [Electronic resource] – Access mode: P. 1175-1178. https://doi.org/10.1109/wartia.2014.6976489

16. Jones M. Recommender systems, Part 1. Introduction to approaches and algorithms. Learn about the concepts that underlie web recommendation engines // Official Web-site of IBM company. – 2013. – [Electronic resource] – Access mode: https://www.ibm.com/developerworks/opensource/library/os-recommender1/index.html?S_TACT=105AGX99&S_CMP=CP

17. Jones M. Recommender systems, Part 2. Introducing open source engines. Explore software for building a recommendation capability // Official Web-site of IBM company. – 2013. – [Electronic resource] – Access mode: https://www.ibm.com/developerworks/library/os-recommender2/index.html?S_TACT=105AGX99&S_CMP=CP

18. Lin Z., Chen H. Recommendation over time: a probabilistic model of time-aware recommender systems // Science China Information Sciences volume 62, Article number 212105. – 2019. – [Electronic resource] – Access mode: https://doi.org/10.1007/s11432-018-9915-8

19. Ozsoy M.G., Polat F. Trust based recommendation systems // Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. – 2013. – P. 1267-1274.

20. Sarwar B., Karypis G., Konstan J., Riedl J. Item-based collaborative filtering recommendation algorithms // Proceedings of the Tenth International World Wide Web Conference. – 2001. – P. 285-295.

21. Stekh Y., Artsibasov V. Adaptive clustering algorithm for recommender systems // Вісник НУ "Львівська політехніка": Комп'ютерні системи проектування. Теорія і практика, № 747. – 2012. – С. 75-78.

22. Stekh Y., Logvinenko A., Lobur M., Artsibasov V. Model and methods for building Web recommendation systems // Proc. of the VIth International Conference on Computer Science and Information Technologies (CSIT 2011). – Lviv, 2011. – P. 314-316.

23. Wei S., Ye N., Zhang Q. Time-Aware Collaborative Filtering for Recommender Systems // Pattern Recognition, Communications in Computer and Information Science (CCPR 2012), Vol. 321, Springer, Berlin, Heidelberg. – 2012. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-3-642-33506-8_81

24. Yuan Q., Cong G., Ma Z., Sun A., Magnenat-Thalmann N. Time-aware point-of-interest recommendation // In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval (SIGIR

’13). Association for Computing Machinery, New York, NY, USA. – 2013. – P. 363-372. – [Electronic resource] – Access mode: https://doi.org/10.1145/2484028.2484030

25. Zhang C., Liu J., Qu Y., Han T., Ge X., Zeng A. Enhancing the robustness of recommender systems against spammers // PLoS ONE 13(11): e0206458. – 2018. – [Electronic resource] – Access mode: https://doi.org/10.1371/journal.pone.0206458

26. Лобур М.В., Стех Ю.В., Шварц М.Є. Побудова асоціативних правил для прогнозування рекомендацій в колаборативних рекомендаційних системах // Квалілогія книги: Збірник наукових праць Української академії друкарства. Львів, № 2 (32). – 2017. – С. 82-86.

27. Лобур М.В., Шварц М.Є., Стех Ю.В. Моделі і методи прогнозування рекомендацій для колаборативних рекомендаційних систем // Вісник Національного університету «Львівська політехніка»: Інформаційні системи та мережі. Львів, № 901. – 2018. – С. 68-75.

28. Barabási A.-L. Network science // Cambridge University Press. – 2018. – 475 p. – [Electronic resource] – Access mode: http://networksciencebook.com/

29. Kumar R., Raghavan P., Rajagopalan S., Sivakumar D., Tomkins A., Upfal E. Stochastic models for the web graph // Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00), IEEE Computer Society, Redondo Beach, CA, USA. – 2000. – P. 57-65. – [Electronic resource] – Access mode: https://doi.org/10.1109/SFCS.2000.892065

30. Traag V.A. Algorithms and Dynamical Models for Communities and Reputation in Social Networks // Springer International Publishing. – 2014. – P. 229. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-3-319-06391-1

31. Мелешко Є.В., Семенов С.Г., Хох В.Д. Дослідження методів побудови рекомендаційних систем в мережі Інтернет // Збірник наукових праць "Системи управління, навігації та зв’язку". Випуск 1(47). – Полтава: ПНТУ ім. Ю. Кондратюка. – 2018. – С. 131-136.

32. Resnick P., Iacovou N., Suchak M., Bergstrom P., Riedl J. Grouplens: An open architecture for collaborative filtering of netnews // Proceedings of the ACM Conference on Computer Supported Cooperative Work. – 1994. – P. 175-186.

33. Koren Y. Factorization meets the neighbor-hood // Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. – 2008. – P. 426-434. – [Electronic resource] – Access mode: https://doi.org/10.1145/1401890.1401944

34. Мелешко Є.В., Хох В.Д., Босько В.В. Дослідження матричних факторизаційних моделей рекомендаційних систем // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 6 (58). – С. 58-62.

35. Krupnik I. Decomposition of a monic matrix polynomial into a product of linear factors // Linear Algebra Appl. – 1992. – P. 239-242.

36. Funk S. Netflix Update: Try This at Home // The Evolution of Cybernetics – A Journal by Simon Funk. – 2006. – [Electronic resource] – Access mode: https://sifter.org/~simon/journal/20061211.html

37. Koren Ye. Collaborative filtering with temporal dynamics // Proceeding KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. – 2009. – P. 447-456.

38. Töscher A., Jahrer M., Bell R.M. The BigChaos So-lution to the Netflix Grand Prize // Netflix prize documentation. – 2009. – [Electronic resource] – Access mode: https://www.netflixprize.com/assets/ GrandPrize2009_BPC_BigChaos.pdf

39. Brownlee J. Overfitting and underfitting with machine learning algorithms. – 2016. – [Electronic resource] – Access mode: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms

40. Mitchell T.M. Machine Learning // McGraw-Hill Education Ltd. – 1997. – 414 p.

41. Overfitting in machine learning: what it is and how to prevent it // Web-site EliteDataScience. – 2017. – [Electronic resource] – Access mode: https://elitedatascience.com/overfitting-in-machine-learning

42. Neumaier A. Solving ill-conditioned and singular lin-ear systems: A tutorial on regularization // SIAM Review, Vol. 40. – 1998. – P. 636-666.

43. Stone P., Hunt E. A computer approach to content analysis: Studies using the general inquirer system. // Spring Joint Computer Conference, AFIPS ’63, New York: ACM. – 1963. – P. 241–256.

44. Мелешко Є.В. Дослідження методів побудови рекомендаційних систем заснованих на фільтрації контенту // Збірник тез III Міжнародної науково-практичної конференції «Інформаційна безпека та комп’ютерні технології», м. Кропивницький, 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 234-237.

45. Мелешко Є.В., Босько В.В., Резніченко В.А. Дослідження методів комп’ютерної лінгвістики для аналізу контенту веб-сайтів // Збірник тез XXI Міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування», м. Кропивницький, 17-18 травня 2019 р. – Кропивницький: КЛА НАУ. – 2019. – С. 96-100.

46. Patel D., Saxena S., Verma T. Sentiment Analysis using Maximum Entropy Algorithm in Big Data // International Journal of Innovative Research in Science, Engineering and Technology. – 2016. – №5. – P. 8355-8361.

47. Taboada M., Brooke J., Tofiloski M., Voll K., Stede M. Lexicon-based methods for sentiment analysis // Computational linguistics, Volume 37, Issue 2. – 2011. – P. 267-307. – [Electronic resource] – Access mode: https://doi.org/10.1162/COLI_a_00049

48. Мелешко Є.В., Якименко М.С. Методи виявлення інформаційно-емоційних впливів у текстовій інформації // Збірник тез VI Міжнародної наукової конференції «Інформація. Комунікація. Суспільство (ICS-2017)», м.

Львів, 18-20 травня 2017 р. – Львів: Національний університет "Львівська політехніка". – 2017. – С. 30-31.

49. Шингалов Д.В., Мелешко Є.В., Минайленко Р.М., Резніченко В.А. Математична модель рекомендаційної системи з врахуванням емоційного забарвлення коментарів у якості контексту // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кропивницький: ЦНТУ, 2018. – Вип. 31. – С. 181-186.

50. Шингалов Д.В., Мелешко Є.В., Минайленко Р.М., Резніченко В.А. Методи автоматичного аналізу тональності контенту у соціальних мережах для виявлення інформаційно-психологічних впливів // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – 2017. – Вип. 30. – С. 196-202.

51. Глибовець М.М., Олецький О.В. Штучний інтелект: Підручник. – К.: Вид. дім "КМ Академія". – 2002. – 366 с.

52. Schmid H. Probabilistic part-of-speech tagging using decision trees // In International Conference on New Methods in Language Processing, Manchester. – 1994. – P. 44-49.

53. C5.0: An Informal Tutorial. – 2019. – [Electronic resource] – Access mode: https://www.rulequest.com/see5-unix.html

РОЗДІЛ 2.
ПОКАЗНИКИ ЯКОСТІ ТА РОБАСТНОСТІ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1. Показники якості рекомендаційних систем

Перевірку якості роботи рекомендаційної системи можна здійснити, використовуючи наступні дані [1, 2]:

1. За допомогою тестової вибірки, підготованої заздалегідь.

2. За допомогою моніторингу роботи рекомендаційної системи та збору даних у реальному часі.

У першому варіанті перевірки усі наявні дані слід розділити на навчаючу та тестову вибірку. Тестову вибірку не можна використовувати при навчанні рекомендаційної системи та при генерації списку рекомендацій. Тестова вибірка буде використана для порівняння даних з неї з даними зі списку рекомендацій. Для перевірки точності прогнозування вподобань при знаходженні однакових об’єктів у тестовій вибірці та у списку рекомендацій, їх прогнозовані та реально поставлені користувачем оцінки порівнюються. Таку перевірку легко організувати, однак її результати не досить інформативні, оскільки збігів у цих двох наборах даних буде досить мало.

Більш інформативним буде другий варіант, оскільки при моніторингу у реальному часі одразу буде видно як користувач реагує на кожну рекомендацію. Можна буде перевірити кожен з прогнозів в тій чи іншій формі – виявити факт обрання/необрання об’єктів зі списку рекомендацій користувачем, а при виставленні оцінки, порівняти її з прогнозованою. Але даний спосіб більш складно реалізувати, необхідно створити систему моніторингу веб-ресурсу, бажано мати доступ до нього на рівні адміністратора.

Найголовнішим показником якості роботи рекомендаційної системи є *точність прогнозування вподобань (Prediction Accuracy)*. Для перевірки

точності прогнозування вподобань користувачів порівнюють два вектори [1, 3]:

1) вектор $\hat{R} = (\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_n)$, що містить список прогнозованих оцінок користувача, впорядкований по спаданню за величиною оцінок;

2) вектор $R = (r_1, r_2, \ldots, r_n)$, що містить справжні оцінки користувача, невідомі системі на етапі формування списку рекомендацій.

Якщо користувачі виставляють оцінки об'єктам, точність прогнозування можна визначити за допомогою середньоквадратичної помилки (2.1) або середньої абсолютної помилки (2.2):

$$RMSE = \sqrt{\frac{1}{|\tau|} \sum_{(u,i)\in\tau} (\hat{r}_{ui} - r_{ui})^2}, \qquad (2.1)$$

$$MAE = \frac{1}{|\tau|} \sum_{(u,i)\in\tau} |\hat{r}_{ui} - r_{ui}|, \qquad (2.2)$$

де $\hat{r}_{ui}$ – прогнозовані рейтинги для тестового набору даних τ пар користувач-об'єкт (u, i), r_{ui} – справжні рейтинги.

Щоб здійснити вимірювання точності система генерує прогнозовані рейтинги $\hat{r}_{ui}$ для тестового набору даних τ пар користувач-об'єкт (u, i), для яких відомі справжні рейтинги r_{ui}. Реальні та прогнозовані рейтинги порівнюються.

Існує також багато інших показників якості роботи рекомендаційних систем [1-7], основні з них наведені нижче.

Прогнозування використання. Оцінювати точність роботи рекомендаційної системи можна завдяки реакції користувача на об'єкти наведені у списку рекомендацій. В такому разі інформація для оцінки роботи рекомендаційної системи з'являється в процесі її використання. Треба зберігати статистику реакцій користувачів на рекомендації та аналізувати її згодом. Можливі варіанти реакції користувача на об'єкти у списку рекомендацій, які і слід зберігати для статистики, наведені на рис. 2.1. Для визначення якості роботи рекомендаційної системи можна застосувати до зібраної статистики формули (2.3)-(2.5).

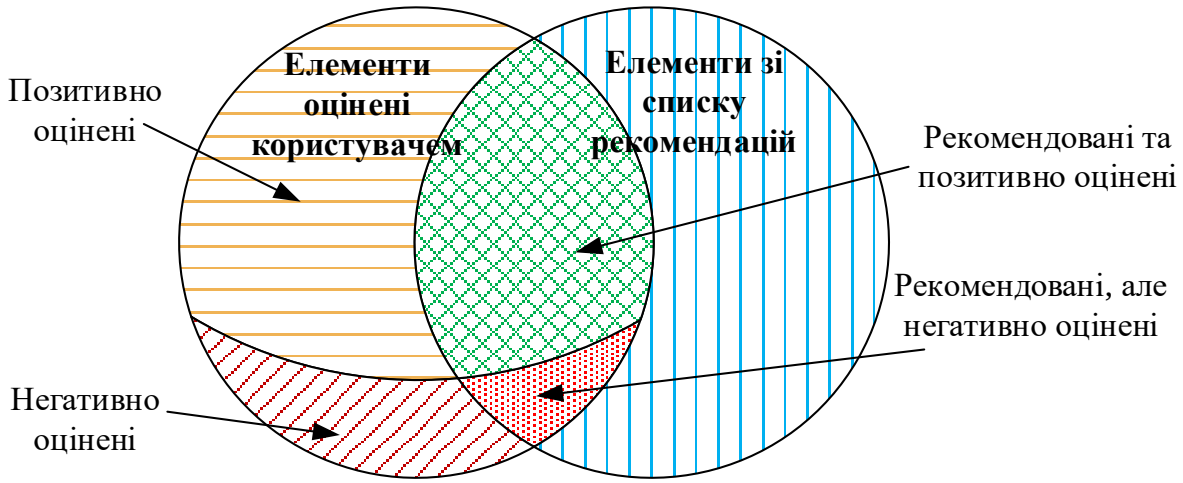


Рис. 2.1. Можливі варіанти реакції користувача на об’єкти у списку рекомендацій

Наведемо можливі результати прогнозу роботи рекомендаційної системи (табл. 2.1).

Таблиця 2.1. Класифікація можливих результатів рекомендації

	Рекомендували	**Не рекомендували**
Позитивно оцінено	True-Positive (*tp*)	False-Negative (*fn*)
Негативно оцінено	False-Positive (*fp*)	True-Negative (*tn*)

Як видно з таблиці, можливі чотири результати роботи рекомендаційної системи:

- *tp* – результати, в яких позитивний прогноз виявився вірним;
- *tn* – результати, в яких негативний прогноз виявився вірним;
- *fp* – результати, в яких позитивний прогноз виявився помилковим;
- *fn* – результати, в яких негативний прогноз виявився помилковим.

Можна підрахувати кількість подій, що відповідають кожній з комірок, та на основі одержаних значень оцінити точність роботи рекомендаційної системи, наприклад, на основі наступних показників [1]:

$$Precision = \frac{tp}{tp + fp}, \tag{2.3}$$

$$Recall\,(True\ Positive\ Rate) = \frac{tp}{tp + fn}, \tag{2.4}$$

$$False\ Positive\ Rate = \frac{fp}{fp + tn}, \tag{2.5}$$

Міри ранжирування. Важливою задачею є ранжування об'єктів у списку рекомендацій. За допомогою даної міри можна визначити наскільки вірно рекомендаційна система впорядкувала елементи у списку рекомендацій.

Міри ранжирування з використанням заздалегідь відомих рейтингів. Як міру ранжування можна використати Normalized Distance-based Performance Measure (NDPM) [7]. Якщо у нас є справжні рейтинги об'єктів r_{ui}, одержані в результаті дій користувача, та рейтинги, згенеровані рекомендаційною системою $\hat{r}_{ui}$ для n_u об'єктів i користувачу u, то можна одержати:

$$C^+ = \sum_{ij} \mathrm{sgn}(r_{ui} - r_{uj})\,\mathrm{sgn}(\hat{r}_{ui} - \hat{r}_{uj}), \tag{2.6}$$

$$C^- = \sum_{ij} \mathrm{sgn}(r_{ui} - r_{uj})\,\mathrm{sgn}(\hat{r}_{uj} - \hat{r}_{ui}), \tag{2.7}$$

$$C^u = \sum_{ij} \mathrm{sgn}^2(r_{ui} - r_{uj}), \tag{2.8}$$

$$C^s = \sum_{ij} \mathrm{sgn}^2(\hat{r}_{ui} - \hat{r}_{uj}), \tag{2.9}$$

$$C^{u0} = C^u - (C^+ + C^-), \tag{2.10}$$

де суми перевищують $\frac{1}{2} n_u (n_u - 1)$ пар об'єктів. Таким чином, C^u – це кількість пар об'єктів, для яких справжні рейтинги встановлюють впорядкування один відносно одного, тоді як C^+ та C^- – це кількість пар об'єктів, для яких рекомендаційна система встановила правильний порядок і неправильний порядок впрорядкування відповідно. C^{u0} – це кількість пар, для яких немає впорядкування в справжньому рейтингу, але рекомендаційна система встановлює для них деяке впорядкування. NDPM одержується наступним чином:

$$NDPM = \frac{C^- + 0.5C^{u0}}{C^u}, \tag{2.11}$$

Таким чином, міра NDPM дає найкращу оцінку 0 – для систем, які правильно передбачають кожне впорядкування пар об’єктів. А найгірша оцінка 1 – призначена системам, що суперечать усім вірним впорядкуванням пар об’єктів.

Міри ранжирування на основі корисності. Популярною альтернативою попереднього методу є припущення, що корисність переліку рекомендацій є сумарною, що визначається сумою корисності окремих рекомендацій. Корисність кожної рекомендації – це корисність рекомендованого об’єкту помножена на коефіцієнт зменшення, який залежить від позиції об’єкту в переліку рекомендацій. Одним з прикладів такої корисності є ймовірність того, що користувач буде дотримуватися рекомендації в даній позиції у списку. Зазвичай передбачається, що користувачі проглядають списки рекомендацій від початку до кінця, з урахуванням того, що переваги рекомендацій значно знижуються до кінця списку.

В багатьох додатках список рекомендацій далеко не самий основний спосіб пошуку об’єктів, він містить невелику кількість елементів, а в крайніх випадках може містити тільки один елемент. В таких системах користувач зазвичай переглядає тільки невелику частину списку рекомендацій – тільки декілька пунктів на початку. В таких випадках цінність рекомендацій знижується дуже швидко відносно позицій елементів у списку. Для таких додатків можна використати *метрику R-Score* [1, 4].

Метрика R-Score передбачає, що корисність рекомендацій знижується експоненціально вниз по впорядкованому списку рекомендацій, щоб отримати наступний бал для кожного користувача u:

$$R_u = \sum_j \frac{\max(r_{ui_j} - d, 0)}{2^{\frac{j-1}{\alpha-1}}}, \qquad (2.12)$$

де i_j – це об’єкт в j-й позиції, r_{ui} – рейтинг (оцінка) користувача i, d – значення максимальної негативної оцінки, а α – період напіврозпаду, який контролює

експоненціальне зниження значення позицій у ранжовому списку. У випадку задач прогнозування рейтингів r_{ui} – це рейтинг (оцінка), наданий користувачем для кожного елемента, а d – це негативна оцінка (наприклад, три зірки з п’яти), і алгоритм отримує кредити лише за позиції рейтингу вище негативних оцінок, тобто, вище значення d (наприклад, 4 або 5 зірок). У задачі передбачення використання r_{ui} звичайно дорівнює 1, якщо u вибирає i, та 0 в іншому випадку, якщо $d = 0$.

Отримані для кожного користувача результати агрегуються за допомогою:

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^*}, \tag{2.13}$$

де R_u^* – оцінка найкращого рейтингу для користувача u.

Крім точності формування та ранжування списку рекомендацій до рекомендаційної системи може висуватися багато інших вимог, наприклад: покриття, різноманітність, новизна, приватність, робастність до атак, адаптивність, масштабованість, пропускна здатність тощо. Покращення даних показників рекомендаційної системи може знизити її точність прогнозування оцінок, але підвищити загальну якість роботи. Розглянемо основні з них.

Важливим показником якості роботи рекомендаційної системи є *покриття (Coverage)* [1], що характеризує відсоток охоплення елементів системи у процесі формування рекомендацій, існує декілька його видів:

– *Покриття каталогу* (покриття простору об’єктів) – може визначатися як відсоток усіх елементів, які можуть бути рекомендовані, даний показник дозволяє виявити об’єкти, які нікому не рекомендуються.

– *Покриття простору користувачів* – може характеризуватися часткою користувачів або взаємодій користувачів, для яких система може рекомендувати об’єкти (у багатьох системах рекомендації можуть не надаватися для користувачів, про яких зібрано мало даних, через низьку впевненість у точності прогнозів), якщо рекомендації слід надавати всім

користувачам у системі, то необхідно йти на компроміс між покриттям та точністю.

– *Різноманітність збуту* – міра неоднорідності вибору різних об'єктів користувачами зі списку рекомендацій, для її визначення можна використовувати різні індекси, зокрема, коефіцієнт Джині (2.14) або ентропію Шеннона (2.15):

$$G = \frac{1}{n-1}\sum_{j=1}^{n}(2j - n - 1)p(i_j), \tag{2.14}$$

де $i_1, \ldots, i_n$ – це список об'єктів, впорядкованих за збільшенням частоти їх вибору користувачем або частоти їх появи у списку рекомендацій $p(i)$. Індекс $G = 0$, коли всі елементи вибираються однаково часто, а $G = 1$ – коли завжди вибирається один елемент.

$$H = -\sum_{i=1}^{n} p(i)\log p(i), \tag{2.15}$$

де ентропія дорівнює 0, коли завжди обирається або рекомендується один об'єкт, та дорівнює $\log n$, коли n об'єктів вибирається чи рекомендується однаково часто.

Точність прогнозування вподобань і покриття є настільки важливими показниками якості роботи рекомендаційних систем, що їх слід віднести до основних.

Розглянемо й інші показники, за допомогою яких можна оцінити якість роботи рекомендаційної системи за різними критеріями.

Усі сучасні рекомендаційні системи схильні до проблеми бульбашки фільтрів, що виникає, коли алгоритм формування списку рекомендацій підбирає інформацію, яку користувач хотів би бачити, і, в результаті, користувачі відділяються від інформації, яка їх не цікавить, тому що зовсім невідома їм або не подобається, фактично ізолюючи їх у власних «бульбашках». Для вирішення даної проблеми до рекомендаційних систем

висуваються наступні вимоги – список рекомендацій повинен володіти наступними властивостями [1, 5, 6, 8, 9]:

– *Різноманітність рекомендацій (Diversity).* Це міра схожості елементів списку. Елементи у рекомендацій системі не повинні бути майже однаковими, вони повинні містити різнотипні об’єкти (наприклад, фільми різних жанрів, а не тільки одного жанру, чи однієї трилогії). Для визначення схожості елементів можна застосовувати різні коефіцієнти подоби (коефіцієнт кореляції Пірсона, косинусну міру, евклідову відстань, відстань Хеммінга тощо), за допомогою яких попарно порівнювати елементи списку, після визначення рівня схожості між окремими елементами можна буде оцінити різноманітність списку рекомендацій вцілому.

– *Новизна рекомендацій (Novelty).* Нові об’єкти у системі можуть ще не мати оцінок і не бути популярними, але вони можуть бути цікавими користувачам через свою новизну. В той же час нові об’єкти необхідно комусь рекомендувати, щоб вони не залишилися без уваги. Якщо користувачу рекомендувати лише популярні об’єкти, скоріше за все він їх і так знає та обере без рекомендаційної системи, такі рекомендації не будуть містити для нього нової інформації. По суті, новизна – це характеристика елемента у списку рекомендацій протилежна його популярності, і в найпростішому випадку може визначатися за формулою [5]:

$$novelty(i) = -\log_2 p(i), \tag{2.16}$$

де $p(i)$ – ймовірність того, що об’єкт i потрапить у список рекомендацій (буде обрано).

Неочікуваність рекомендацій (Serendipity). Неочікуваність представляє собою деякий сюрприз у списку рекомендацій, несхожість на історію дій користувача. Не існує консенсусу у визначенні неочікуваності, однак більшість авторів вказує, що елемент, який має властивість неочікуваність, повинен бути важливим, новим та непрогнозованим для користувача [6]. Важливість для користувача виражається в його реакції на даний елемент після рекомендацій,

новизна виражається в тому, наскільки користувач знайомий з даним елементом. Елемент може бути незнайомим для користувача, якщо: 1) користувач ніколи не чув про даний елемент, 2) користувач чув про даний елемент, але ніколи не використовував, 3) користувач використовував даний елемент, але забув про це. Елемент може бути непрогнозованим для користувача, якщо: 1) користувач не очікує, що цей елемент буде для нього актуальним, 2) користувач не очікує, що цей елемент буде рекомендований йому, 3) користувач не знайшов би цього елементу самостійно, 4) цей елемент значно відрізняється від елементів, які як правило, обирає користувач, 5) користувач не очікує даного елементу у списку рекомендацій, оскільки він переглядав інші види елементів.

Збільшення різноманітності, неочікуваності та новизни рекомендацій може знизити точність прогнозування та точність ранжування, в той же час може підвищитися покриття каталогу об’єктів, різноманітність збуту та частково вирішитися проблема бульбашки фільтрів.

На даний час не існує загальноприйнятих мір та методів оцінки різноманітності, новизни та неочікуваності списків рекомендацій.

Серед показників якості рекомендаційних систем, що є важливими з погляду інформаційної безпеки можна виділити наступні [1, 10-13]:

1. *Приватність користувача (Privacy).* Міра того наскільки приватна інформація користувачів, яку вони надають рекомендаційній системі для одержання списків пропозицій, захищена від потрапляння її до третіх сторін.

2. *Ризик для користувача (Risk).* В деяких випадках рекомендації можуть бути пов’язані з ризиком. Наприклад, якщо об’єктами в рекомендаційній системі є акції, кредити, депозити, ліки, медичні послуги, політичні акції тощо. В таких випадках може бути необхідним врахування не тільки вподобань користувача при формуванні рекомендацій, а й інших факторів, врахування яких здатне мінімізувати ризик для користувача, що буде переглядати та обирати рекомендації.

3. *Робастність системи до атак (Robustness).* Здатність системи надавати адекватні рекомендації при появі некоректної інформації. Некоректна інформація може виникати при іформаційних атаках бот-мереж на систему для (збільшення або зменшення) рейтингів певних об’єктів з метою зміни частоти потрапляння їх у списки рекомендацій.

Також, в залежності від потреб певного веб-ресурсу чи додатку, можна виділити й інші показники якості роботи рекомендаційних систем, наприклад [1, 14, 15]:

– *Впевненість (Confidence)*. Рекомендаційна система може додавати до своїх рекомендацій процент впевненості у них. Так, наприклад, система може вказати для першої рекомендації прогнозовану оцінку користувача 5 балів з впевненістю на 95%, а для другої рекомендації – 5 балів з впевненістю на 89%. Користувач може враховувати даний параметр та обирати в першу чергу об’єкти з більшим значенням впевненості системи.

– *Довіра (Trust)*. Дана властивість характеризує наскільки користувач може довіряти даній рекомендаційній системі. Якщо рекомендаційна система запропонувала користувачу лише невідомі йому об’єкти, то користувач може засумніватися у правильності роботи системи, а якщо серед рекомендацій є певна кількість об’єктів, про які він знає і які йому подобаються, то рівень довіри до системи буде вищим. Тобто, деяка кількість об’єктів, про які відомо користувачу, може збільшити довіру до системи. Ще одним варіантом збільшення довіри до системи є пояснення до рекомендації. Наприклад, коли елемент зі списку рекомендацій подається у вигляді «Якщо Вам подобається X, то спробуйте Y». Також довіра користувача зростає, якщо рекомендації були йому корисні і відповідали його вподобанням, та знижується в протилежному випадку.

– *Адаптивність (Adaptivity)*. Реальна система рекомендацій може працювати у ситуації, коли колекції об’єктів, і/або інтереси користувачів швидко змінюються. Прикладом таких систем можуть бути рекомендаційні

системи новинних сайтів, коли об’єкти (новини) цікаві тільки на певному проміжку часу. В таких системах з одного боку цікавість до певного об’єкту існує тільки деякий час, з іншого боку якісь старі об’єкти можуть знову ставати дуже цікавими, якщо нові об’єкти у системі поновлюють інтерес до старих. В таких системах слід вибирати швидкі алгоритми, навіть якщо доводиться в деякій мірі жертвувати точністю. Інший тип адаптивності – це те, з якою швидкістю система адаптується до вподобань користувача або до змін в його профілі. Якщо дії користувача на сайті занадто повільно змінюють рекомендації, він може відмовитися оцінювати об’єкти, не отримавши зворотного зв’язку. Адаптивність алгоритмів можна оцінювати шляхом визначення різниці між списками рекомендацій до та після додавання нової інформації з профіля користувача, наприклад, за допомогою міри ентропії Шеннона.

– *Масштабованість (Scalability)*. Зі збільшенням об’ємів даних багато алгоритмів працюють значно повільніше або вимагають додаткових ресурсів, таких як обчислювальні потужності або пам’ять. Тому важливо враховувати просторову та часову складність алгоритмів. В реальних системах може виникнути необхідність згодитися на меншу точність рекомендацій для одержання більш масштабованої системи.

– *Пропускна здатність (Throughput)*. Це кількість рекомендацій, які система може створити та надати користувачам в одиницю часу.

– *Корисність (Utility)*. Значення, що характеризує, яку вигоду від рекомендацій отримує власник системи та/або користувач.

При побудові рекомендаційної системи для контент-орієнтованого веб-сайту чи додатку досить логічним кроком буде визначення списку показників якості, яким повинна задовольняти розроблювана рекомендаційна система, та вибір/розробка алгоритмів і методів її побудови на основі визначення у процесі їх тестування, наскільки вони задовольняють висунутим критеріям.

2.2. Показники робастності рекомендаційних систем

Стійкість (робастність) системи в комп'ютерній інженерії в загальному випадку означає міру здатності обчислювальної системи відновлюватися при виникненні помилкових ситуацій як зовнішнього, так і внутрішнього походження [2, 16].

Під стійкістю (робастністю) рекомендаційної системи часто розуміють [1, 13, 17] збереження точності рекомендацій за наявності підробленої інформації, яка зазвичай додається до системи навмисне з метою впливу на рекомендації користувачам. Для просування свого контенту на певному веб-ресурсі, треті особи можуть створити мережу ботів, що будуть викривляти рейтинги об'єктів системи, а отже, і ймовірність та кількість потраплянь їх у рекомендації звичайним користувачам системи. Такі дії, як правило, називають атакою на рекомендаційну систему [1, 13, 18-22].

Будемо виходити з більш широкого визначення для поняття *стійкість* та сформулюємо *стійкість рекомендаційних* систем наступним чином.

Стійкість (робастність) рекомендаційної системи – це міра здатності рекомендаційної системи створювати релевантні та точні рекомендації користувачам системи не зважаючи на зовнішні впливи у вигляді інформаційних атак, та внутрішні проблеми, такі, наприклад, як проблема холодного старту та постійного холодного старту, розрідженості даних, появи некоректної інформації тощо.

Таким чином задачу створення стійкої рекомендаційної системи слід розділити на дві:

1. Забезпечити стійкість рекомендаційної системи до інформаційних атак.

2. Забезпечити стійкість рекомендаційної системи до внутрішніх проблем системи, таких як проблема холодного страту, розрідженості даних тощо.

Оцінити стійкість рекомендаційної системи можна вимірюючи показники точності її роботи до та після виникнення дестабілізуючих факторів. Чим

менша різниця між цими двома одержаними значеннями точності, тим вища стійкість системи до тих дестабілізуючих факторів, які використовувалися у вимірюваннях.

При вимірюванні стійкості рекомендаційної системи можуть виникнути наступні складнощі. У реальних умовах роботи веб-ресурсу з рекомендаційною системою не завжди можна спостерігати ситуації породжені дестабілізуючими факторами. Наприклад, важко передбачити здійснення справжнього нападу на реальну систему і скористатися даним фактом для проведення випробувань системи, тому зазвичай такі стани системи моделюють за допомогою програмних симуляцій [1, 23]. Так, наприклад, у роботах [1, 23] експерименти проводяться на основі реальних даних з відкритих датасетів з додаванням у них даних створених бот-мережами, отриманих за допомогою програмного моделювання атак на рекомендаційну систему, що дозволяє визначити стійкість системи до різних видів атак та емпірично вимірювати середню вартість успішної атаки.

При оцінюванні стійкості системи слід враховувати, що вона буде різною по відношенню до різних дестабілізуючих факторів, зокрема, стійкість системи може значно відрізнятися для різних видів інформаційних атак.

Якщо необхідно розробити стійку до дестабілізуючих факторів рекомендаційну систему, то задачу оптимізації можна сформулювати наступним чином:

$$d(R_b, R_a) \to \min , \tag{2.17}$$

де вектор R_b містить список прогнозованих рекомендацій (оцінок) для користувачів, сформований у нормальному режимі роботи системи; а вектор R_a містить список прогнозованих рекомендацій (оцінок) для користувачів, сформований під час дії дестабілізуючих факторів.

Взагалі, створити рекомендаційну систему, стійку до всіх видів інформаційних атак та внутрішніх помилок, складно та затратно. Тому більш

доцільно оцінити вартість збитків від виникнення дестабілізуючих факторів та вартість усунення даних факторів, щоб на основі даної інформації приймати рішення про доцільність внесення тих чи інших змін у систему, направлених на підвищення її стійкості.

Існуючі методи оцінки стійкості (робастності) рекомендаційних систем направлені на визначення стійкості системи до інформаційних атак [1, 17, 24]. Хоча дані методи, після деякої адаптації, можна застосовувати і до визначення стійкості й до внутрішніх дестабілізуючих факторів. Отже, розглянемо існуючі методи оцінювання стійкості рекомендаційних систем до інформаційних атак.

Оскільки мета інформаційних атак на рекомендаційні системи – це зміна рейтингу цільового об’єкту, то для вимірювання стійкості рекомендаційної системи, потрібно оцінити, наскільки змінилися рейтинги об’єктів системи після атаки.

Також показники стійкості повинні фіксувати відмінності в рекомендованому статусі цільового об’єкту до та після атаки, тобто, визначати наскільки змінилася частота потрапляння цільового об’єкту у списки рекомендацій користувачам та чи покращилися його позиції у даних списках. Ці дані важливіші за дані про зміни у рейтингах, оскільки показують ефективність атаки. Але показники, спрямовані на виявлення змін у рекомендованому статусі об’єкту, можуть не зафіксувати невдалу атаку на відміну від показників орієнтованих на виявлення змін у рейтингах, так як незначна зміна рейтингу об’єкту системи може не вплинути на його рекомендований статус, але може бути зафіксована.

Багато дослідників використовують [1, 19] середній зсув прогнозування для оцінювання змін у прогнозованих рейтингах.

Нехай U_T та I_T – це набори користувачів та об’єктів системи. Для кожної пари user-item (u, i) зсув прогнозування може бути виміряний як:

$$\Delta_{u,i} = p'_{u,i} - p_{u,i}, \tag{2.18}$$

де p і p' є прогнози до- та після атаки відповідно.

Позитивне значення $\Delta_{u,i}$ означає, що атака зуміла збільшити прогнозовані рейтинги об’єкту, а негативне – зменшити їх. Середній зсув прогнозування рейтингів об’єкта *i* для всіх користувачів можна обчислити наступним чином:

$$\Delta_i = \sum_{u \in U_T} \frac{\Delta_{i,u}}{|U_T|}, \tag{2.19}$$

де $|U_T|$ – кількість елементів у наборі користувачів U_T.

Аналогічно середній зсув прогнозування рейтингів для всіх об’єктів у тестовій виборці може бути обчислений як:

$$\overline{\Delta} = \sum_{i \in I_T} \frac{\Delta_i}{|I_T|}, \tag{2.20}$$

де $|I_T|$ – кількість елементів у наборі об’єктів I_T.

Зсув прогнозування дозволяє дослідити як атаки впливають на рейтинги цільових об’єктів. Однак навіть дуже сильні зміни у рейтингу об’єкту можуть не змінити його рекомендований статус. Така ситуація може виникнути, наприклад, якщо його початкова середня оцінка дуже низька, що навіть сильний її приріст недостатній для потрапляння у списки рекомендацій.

Для оцінювання впливу атаки на списки рекомендацій існує наступний показник – коефіцієнт звернень [1, 25, 19]. Нехай R_u – це набір найпопулярніших N рекомендацій для користувача u. Якщо цільовий об’єкт потрапляє в R_u, для користувача u, функція оцінювання результату атаки $H_{u,i}$ має значення 1; інакше – 0. Коефіцієнт звернень для елемента *i* визначається як:

$$HitRatio_i = \sum_{u \in U_T} \frac{H_{i,u}}{|U_T|}. \tag{2.21}$$

Середнє значення коефіцієнту звернень може бути обчислене як:

$$\overline{HitRatio} = \sum_{i \in I_T} \frac{HitRatio_i}{|I_T|}. \tag{2.22}$$

Для оцінювання стійкості різних методів колаборативної фільтрації формується два набори тестових даних, одні без додавання профілів, які моделюють атаку (профілів ботів), інші з додаванням таких профілів. Для кожного набору даних створюються списки рекомендацій, а потім обчислюються вищерозглянуті показники стійкості. Чим менша різниця між прогнозуванням вподобань, яку можна оцінити значеннями вищерозглянутих показників стійкості, для першого та другого наборів даних, тим більш стійка до дестабілізуючих факторів розглядувана рекомендаційна система.

Вибір шляхів забезпечення стійкості рекомендаційних систем залежить від того, до яких дестабілізуючих факторів слід її забезпечити.

Стійкість системи до холодного старту та низької якості вхідних даних. Якщо до системи додається новий користувач або об’єкт, про якого ще немає даних або їх дуже мало, для формування рекомендацій можна використовувати контекстну інформацію, інформацію про найрейтинговіші об’єкти системи тощо. В будь-якому випадку треба використовувати та поєднувати різні алгоритми фільтрації даних, серед яких є такі, що використовують контекстну та групову інформацію.

Стійкість системи до атак накручування рейтингів. Побудова стійких до інформаційних атак рекомендаційних систем базується на двох принципах: 1) виявлення спаму серед дій користувачів; 2) невраховування при побудові рекомендацій даних користувачів, що поширюють спам. На сьогоднішній день існують різні методи, які дозволяють виявити атаку на рекомендаційну систему, вони базуються на пошуку профілів спам-користувачів (ботів). Так як при атаці створюють не один фейковий профіль, а багато схожих, такі спам-користувачі будуть мати незвично високу схожість між собою та аномальну поведінку, у порівнянні зі звичайними користувачами, що часто дає можливість їх виявити. Однак, надійні методи виявлення атак на рекомендаційні системи та протидії ним все ще залишаються активною областю досліджень.

Стійкість до атак на приватність користувачів. Для забезпечення

приватності користувача рекомендаційної системи можна застосовувати наступні методи:

– Інформування користувачів про те, яку інформацію про них збирає рекомендаційна система, гнучкі налаштування параметрів конфіденційності.

– Анонімізація – інформація про користувача може частково видалятися або піддаватися обфускації (маскуванню) користувачем або власником рекомендаційної системи.

– Рандомізація – дані користувача (наприклад, виставлені об’єктам оцінки) можуть бути частково зашумлені випадковими значеннями. Необхідний рівень шуму залежить від того, як часто дані будуть використовуватися, і передбачає балансування між точністю прогнозування та конфіденційністю користувача.

– Шифрування даних користувача, що зберігаються в базі даних рекомендаційної системи.

Підвищувати стійкість рекомендаційної системи до зовнішніх дестабілізуючих факторів можна за допомогою наступних підходів [10, 13, 17, 19, 20, 25, 26]:

1. Заміняти одні методи створення списків рекомендацій на інші більш стійкі до дестабілізуючих факторів аналоги.

2. Використовувати методи виявлення профілів ботів та вилучати їх дані з розрахунків для формування рекомендацій користувачам системи.

3. Впровадити та враховувати у рекомендаційній системі параметр репутація та/або експертність для користувачів таким чином, щоб користувачі з низькою репутацією (експертністю), визначеною на основі їх попередніх дій, слабо впливали або не впливали на формування списків рекомендацій.

2.3. Проблеми рекомендаційних систем

Сучасні рекомендаційні системи мають ряд стандартних проблем та недоліків, пов’язаних з внутрішніми та зовнішніми дестабілізуючими

факторами.

До найпоширеніших загальних проблем рекомендаційних систем, пов’язаних з внутрішніми дестабілізуючими факторами у комп’ютерних мережах, можна віднести проблему холодного старту [1, 27, 28] та бульбашки фільтрів [9, 29].

Головними проблемами контентної фільтрації є складність виділення ознак об’єктів та проблема холодного старту для користувачів [1, 27, 28, 30].

Для колаборативної фільтрації актуальними є проблеми холодного старту для користувачів та холодного старту для об’єктів [1, 27, 28, 30].

Одна з головних проблем рекомендаційних систем – проблема холодного старту (Cold-start Problem, CSP). Вона виникає тоді, коли в системі з’являються нові елементи – або нові користувачі (User Cold-Start), історія вподобань яких порожня, або нові об’єкти (Item Cold-Start), у яких ще немає оцінок та/або набору ознак [1, 27, 28, 30].

У багатьох реальних системах проблема холодного старту може набувати характеру циклічної проблеми для вже відомих користувачів або об’єктів. Наприклад, якщо частина користувачів змінює свої інтереси. Дана проблема отримала назву проблеми постійного холодного старту (Continuous Cold-start Problem, CoCoS) [27].

Як і проблема холодного старту, проблема постійного холодного старту може виникати з користувачами (User Continuous Cold-start Problem) та з об'єктами (Item Continuous Cold-start Problem). User Continuous Cold-start Problem виникає для користувачів, що змінюють свої вподобання або рідко з’являються у системі та рідко оцінюють нові об’єкти. Item Continuous Cold-start Problem виникає при наявності об’єктів, властивості яких можуть змінитися з часом.

Для вирішення проблеми холодного старту, як правило, застосовують наступні підходи:

1) гібридизація рекомендаційної системи з поєднанням контентної та

колаборативної фільтрації [31, 32].

2) використання контексту, в якому створюються та надаються рекомендації (демографічні дані, час та дата тощо) [1, 30, 33-35].

Однак всі ці способи не підходять в разі проблеми постійного холодного старту, оскільки припускають, що після того, як користувач став «відомим», він залишається таким необмежену кількість часу, а об’єкти рекомендацій не можуть змінювати свої властивості. Для рішення даної проблеми треба не тільки прогнозувати вподобання користувачів, а й відслідковувати та прогнозувати зміну їх вподобань, а також враховувати можливість зміни властивостей об’єктів рекомендацій. Дану проблему на сьогоднішній день намагаються вирішувати методами машинного навчання, що підвищують адаптивність системи до постійних змін, а також врахуванням часу, як одного з параметрів при формуванні рекомендацій.

Не менш важливою проблемою рекомендаційної системи є проблема бульбашки фільтрів [29]. Класичні рекомендаційні системи пропонують користувачам об’єкти, виходячи лише з їх попередніх вподобань. Проблема бульбашки фільтрів виникає у рекомендаційних системах, коли алгоритм формування рекомендацій та видачі інформації вибірково підбирає дані, враховуючи тільки те, яку інформацію користувач переглядав і оцінював раніше, і, в результаті, користувачі відділяються від інформації, яка їх раніше не цікавила або невідома їм, або не подобається, фактично ізолюючи їх у власних «бульбашках» [9, 29]. Отже, користувач потрапляє у інформаційне середовище, в якому спостерігає лише обмежену кількість однотипних об’єктів. Це явище виникає, якщо користувач для одержання нової інформації використовує переважно списки рекомендацій. Наслідки, викликані бульбашкою фільтрів [36]:

1. Користувач не одержує альтернативну інформацію, яка може бути йому корисною (наприклад, види об’єктів, про які він зовсім не знає, але які ефективніше вирішать задачі його пошуку).

2. У користувача формується викривлена точка зору на інформаційне середовище, так як він не бачить картини в цілому (наприклад, при рекомендації новин).

3. Користувач може втратити інтерес до списку рекомендацій, так як йому весь час пропонують однотипні об’єкти (наприклад, втратить інтерес до прослуховування онлайн радіо з однотипним набором пісень).

Так як усі рекомендаційні системи як основну метрику якості своєї роботи використовують точність прогнозування вподобань користувачів на основі їх попередніх дій, а формальна постановка задачі прогнозу оцінок виглядає наступним чином [1, 3, 37]:

$$d(R,V) \to \min, \tag{2.23}$$

де $R = (r_1, r_2, \ldots, r_n)$ – вектор, що містить список прогнозованих оцінок користувача, впорядкований по спаданню за величиною оцінок; $V = (v_1, v_2, \ldots, v_n)$ – вектор, що містить справжні оцінки користувача, невідомі системі на етапі формування списку рекомендацій, то для всіх рекомендаційних систем проблема бульбашки фільтрів є актуальною, так як якісна рекомендаційна система повинна створювати рекомендації максимально схожі на попередні вподобання користувача.

Для вирішення проблеми бульбашки фільтрів, як правило, застосовують накладання додаткових вимог до процесу створення рекомендацій, наприклад, забезпечення наступних властивостей, що вимагають додавання випадкових елементів у списки рекомендацій [1, 5, 6, 8, 9]:

1. *Різноманітність* (Diversity) – міра схожості між елементам списку рекомендацій не повинна бути меншою певної заданої величини. Міру схожості між елементами можна визначати на основі коефіцієнтів подоби, наприклад, за допомогою однієї з формул (1.12)-(1.24).

2. *Неочікуваність* (Serendipity) – у список рекомендацій з деякою ймовірністю повинні потрапляти елементи підібрані випадковим чином або на основі методів, не пов’язаних з прогнозуванням вподобань користувача.

3. *Новизна* (Novelty) – у список рекомендацій з деякою ймовірністю повинні потрапляти елементи, які ще ніким не були оцінені або отримали мало оцінок.

При забезпеченні виконання розглянутих вище додаткових вимог до формування списку рекомендацій (різноманітності, неочікуваності та новизни), буде зменшуватися точність прогнозування вподобань, але можна буде подолати проблему бульбашки фільтрів та покращити інші показники якості роботи рекомендаційної системи, наприклад, покриття каталогу.

До зовнішніх дестабілізуючих факторів у роботі рекомендаційної системи можна віднести різні інформаційні загрози [1, 36, 38]. Існує декілька основних загроз інформаційній безпеці користувачів рекомендаційної системи, а саме: загроза одержати рекомендації, сформовані внаслідок накручування рейтингів певних об’єктів в результаті інформаційної атаки; загроза втрати приватності даних користувача, зокрема, приватності його вподобань, коли їх можуть дізнатися та використати треті особи для просування своїх інтересів; загроза одержати неякісні списки рекомендацій, використання яких може нести ризики для життя чи здоров’я користувача або впливати на рішення у соціальних чи політичних сферах у інтересах третіх сторін. Нижче розглянемо дані загрози детальніше.

Атаки накручування рейтингів (атаки ін’єкцією профілів). Некоректні рекомендації можуть виникати при атаках на рекомендаційну систему з метою збільшення (зменшення) рейтингів певних об’єктів. Реалізуються такі атаки шляхом створення множини акаунтів ботів (бот-мережі), які скоординовано виставляють високі (або низькі) оцінки певному об’єкту чи об’єтам [1, 10, 13, 19, 25, 39].

Загрози приватності користувачів. Рекомендаційні системи збирають велику кількість даних про користувачів, значну частину яких користувачі охоче надають самі в обмін на корисні рекомендації. Однак для більшості користувачів, важливо щоб їхні вподобання залишалися приватними, тобто,

жодна третя сторона не могла використовувати рекомендаційні системи, щоб дізнатися інформацію про них або їх вподобання. Дана загроза цілком реальна. Як один з прикладів можна навести скандальні ситуації з рекомендаціями друзів у Facebook [11], які виникали через експерименти з використанням даних геолокації [11], подібні рекомендації частково порушували приватні дані людей та давали інформацію третім особам про їх переміщення.

Ризиковані рекомендації. В деяких випадках рекомендації можуть бути пов'язані з ризиком [1]. Наприклад, якщо об'єктами в рекомендаційній системі є акції, кредити, депозити, ліки, медичні послуги, політичні акції тощо. В таких випадках може бути необхідним врахування не тільки вподобань користувача при формуванні рекомендацій, а й інших факторів, врахування яких здатне мінімізувати ризик для користувача, що буде переглядати та обирати рекомендації. Ризиковані рекомендації можна вилучати зі списку рекомендацій або маркувати їх відповідною інформацією для користувачів.

Контрольні питання

1. Які існують метрики для оцінки якості рекомендаційних систем?
2. Як визначаються точність та повнота роботи рекомендаційної системи? Чим вони відрізняються?
3. Які існують метрики для оцінки робастності рекомендаційних систем?
4. Які існують внутрішні та зовнішні фактори дестабілізації роботи рекомендаційної системи?
5. Що таке бульбашка фільтрів?
6. Якими способами можна вирішити проблему бульбашки фільтрів?
7. У чому полягає проблема холодного старту? Які види її бувають?
8. Як можна вирішити проблему холодного старту?
9. Як можна забезпечити різноманітність, неочікуваність та новизну елементів списку рекомендацій?

10. Які існують основні загрози інформаційній безпеці користувачів рекомендаційної системи?

Список літератури до розділу

1. Ricci F., Rokach L., Shapira B., Kantor P.B. (Editors) Recommender Systems Handbook // Boston: Springer. – 2011. – 842 p. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-0-387-85820-3
2. Мелешко Є. В. Методологія забезпечення стійкості рекомендаційних систем до дестабілізуючих факторів у комп'ютерних мережах: дис. ... докт. техн. наук з комп'ютерних систем та компонентів: 05.13.05 – Комп'ютерні системи та компоненти / Є. В. Мелешко; Черкаський держ. технологічний ун-т. – Черкаси, 2020. – 323 с.
3. Мелешко Є.В. Методи оцінки якості роботи рекомендаційних систем // Системи управління, навігації та зв'язку. – Полтава: ПНТУ, 2018. – Вип. 5 (51). – С. 92-97.
4. Breese S., Heckerman D., Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. // In Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, Volume 461, San Francisco, CA. – 1998. – P. 43-52.
5. Castells P., Vargas S., Wang J. Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance. – 2011. – 8 p. – [Electronic resource] – Access mode: https://www.semanticscholar.org/paper/Novelty-and-Diversity-Metrics-for-Recommender-and-Castells-Vargas/4ec6bd672aaaa075b42a751099eb9317857e6e0c
6. Kaminskas M., Bridge D. Measuring Surprise in Recommender Systems // In: Workshop on recommender systems evaluation: dimensions and design (REDD 2014), October 10, 2014, Silicon Valley, USA. – 2014. – 6 p.
7. Yao Y.Y. Measuring retrieval effectiveness based on user preference of

documents // Journal of the American Society for Information Science. – 1995. – №46. – P. 133-145.

8. Kotkov D., Konstan J.A., Zhao Q., Veijalainen J. Investigating Serendipity in Recommender Systems Based on Real User Feedback // In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC, Pau, France. – 2018. – P. 1341-1350. – [Electronic resource] – Access mode: https://dl.acm.org/doi/10.1145/3167132.3167276

9. Zhang Y.C., Séaghdha D.Ó., Quercia D., Jambor T. Auralist: Introducing Serendipity into Music Recommendation // WSDM '12: Proceedings of the fifth ACM international conference on Web search and data mining. – 2012. – P. 13-22. – [Electronic resource] – Access mode: https://doi.org/10.1145/2124295.2124300

10. Chirita P.A., Nejdl W., Zamfir C. Preventing shilling attacks in online recommender systems // In Proceedings of the ACM Workshop on Web Information and Data Management. – 2005. – P. 67-74.

11. Hill K. Facebook recommended that this psychiatrist's patients friend each other // News web-site Splinternews.com. – 2016. – [Electronic resource] – Access mode: https://splinternews.com/facebook-recommended-that-this-psychiatrists-patients-f-1793861472

12. Lam S.K., Riedl J. Shilling recommender systems for fun and profit // In Proceedings of the 13th International World Wide Web Conference. – 2004. – P. 393-402. – [Electronic resource] – Access mode: https://dl.acm.org/doi/10.1145/988672.988726

13. Mobasher B., Burke R., Bhaumik R.,Williams C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness // ACM Transactions on Internet Technology, Vol. 7(4). – 2007. – 41 p. – [Electronic resource] – Access mode: https://doi.org/10.1145/1278366.1278372

14. Mohammadi V., Rahmani A.M., Darwesh A.M., Sahafi A. Trust-based recommendation systems in Internet of Things: a systematic literature review // Human-centric Computing and Information Sciences. – 2019. – 61 p. – [Electronic

resource] – Access mode: https://doi.org/10.1186/s13673-019-0183-8

15. Segaran T. Programming Collective Intelligence: Building Smart Web 2.0 Applications – O'Reilly. – 2007. – 362 p.

16. Zhang C., Garlan D., Kang E. A behavioral notion of robustness for software systems // In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), Association for Computing Machinery, New York, NY, USA. – 2020. – P. 1–12. – [Electronic resource] – Access mode: https://doi.org/10.1145/3368089.3409753

17. Zhang C., Liu J., Qu Y., Han T., Ge X., Zeng A. Enhancing the robustness of recommender systems against spammers // PLoS ONE 13(11): e0206458. – 2018. – [Electronic resource] – Access mode: https://doi.org/10.1371/journal.pone.0206458

18. Mobasher B., Burke R.D., Sandvig J.J. Model-based collaborative filtering as a defense against profile injection attacks // In Proceedings of the 21st national conference on Artificial intelligence, Vol. 2 (AAAI'06). AAAI Press. – 2006. – P. 1388-1393. – [Electronic resource] – Access mode: https://dl.acm.org/doi/10.5555/1597348.1597409

19. O'Mahony M.P., Hurley N.J., Silvestre G.C.M. Promoting recommendations: An attack on collaborative filtering // DEXA, Lecture Notes in Computer Science, Vol. 2453. – 2002. – P. 494-503.

20. Williams A. C., Mobasher B., Burke R. Defending recommender systems: detection of profile injection attacks // Service Oriented Computing and Applications. – 2007. – P. 157–170.

21. Zhou W., Wen J., Koh Y.S., Alam S., Dobbie G. Attack detection in recommender systems based on target item analysis // International Joint Conference on Neural Networks (IJCNN 2014), Beijing. – 2014. – P. 332-339. – [Electronic resource] – Access mode: https://ieeexplore.ieee.org/document/6889419

22. Zhou W., Wen J., Qu Q., Zeng J., Cheng T. Shilling attack detection for

recommender systems based on credibility of group users and rating time series // PLoS ONE 13(5): e0196533. – 2018. – [Electronic resource] – Access mode: https://doi.org/10.1371/journal.pone.0196533

23. Kaur P., Goel S. Shilling attack models in recommender system // International Conference on Inventive Computation Technologies (ICICT), Coimbatore. – 2016. – P. 1-5. – [Electronic resource] – Access mode: https://ieeexplore.ieee.org/document/7824865/

24. Мелешко Є.В., Хох В.Д., Улічев О.С. Дослідження робастності рекомендаційних систем з колаборативною фільтрацією до інформаційних атак // Електронне фахове наукове видання Кібербезпека: освіта, наука, техніка.– Київ: КУБГ, 2019. – Т.1, № 5. – С. 95-104.

25. Kumari T., Punam B. A Comprehensive Study of Shilling Attacks in Recommender Systems // IJCSI International Journal of Computer Science Issues, Volume 14, Issue 4. – 2017. – [Electronic resource] – Access mode: https://www.ijcsi.org/articles/A-comprehensive-study-of-shilling-attacks-in-recommender-systems.php

26. Gunes I., Kaleli C., Bilge A., Polat H. Shilling attacks against recommender systems: a comprehensive survey // Artificial Intelligence Review, Vol. 42. – 2014. – P. 767-799. – [Electronic resource] – Access mode: https://doi.org/10.1007/s10462-012-9364-9

27. Bernardi L., Kamps J., Kiseleva J., Mueller M.J.I. The Continuous Cold Start Problem in e-Commerce Recommender Systems. – 2015. – 6 p. – [Electronic resource] – Access mode: https://arxiv.org/abs/1508.01177

28. Huba G. The Cold Start Problem for Recommender Systems // Yuspify web-site of the machine-learning-based recommendations system for web-stores. – 2015. – [Electronic resource] – Access mode: https://www.yuspify.com/blog/cold-start-problem-recommender-systems/

29. Weisberg J. Bubble Trouble: Is Web personalization turning us into solipsistic twits? – 2011. – [Electronic resource] – Access mode:

https://slate.com/news-and-politics/2011/06/eli-pariser-s-the-filter-bubble-is-web-personalization-turning-us-into-solipsistic-twits.html

30. Мелешко Є.В., Семенов С.Г., Хох В.Д. Дослідження методів побудови рекомендаційних систем в мережі Інтернет // Збірник наукових праць "Системи управління, навігації та зв'язку". Випуск 1(47). – Полтава: ПНТУ ім. Ю. Кондратюка. – 2018. – С. 131-136.

31. Burke R. Hybrid Web Recommender Systems // The Adaptive Web. Lecture Notes in Computer Science, Vol. 4321. Springer, Berlin, Heidelberg. – 2007. – С. 377-408.

32. Çano E., Morisio M. Hybrid recommender systems: A systematic literature review // Journal Intelligent Data Analysis, Vol. 21, No. 6. – 2017. – P. 1487-1524. – [Electronic resource] – Access mode: https://arxiv.org/abs/1901.03888

33. Javed U., Shaukat K., Hameed I. A., Iqbal F., Mahboob Alam T., Luo S. A Review of Content-Based and Context-Based Recommendation Systems // International Journal of Emerging Technologies in Learning (iJET), Kassel, Germany, Vol. 16(3). –2021. – P. 274-306. – [Electronic resource] – Access mode: https://www.learntechlib.org/p/219036/

34. Suhaim A. B., Berri J. Context-Aware Recommender Systems for Social Networks: Review, Challenges and Opportunities // in IEEE Access, Vol. 9. – P. 57440-57463, – 2021. – DOI: 10.1109/ACCESS.2021.3072165 – [Electronic resource] – Access mode: https://ieeexplore.ieee.org/abstract/document/9399450/

35. Шингалов Д.В., Мелешко Є.В., Минайленко Р.М., Резніченко В.А. Математична модель рекомендаційної системи з врахуванням емоційного забарвлення коментарів у якості контексту // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кропивницький: ЦНТУ, 2018. – Вип. 31. – С. 181-186.

36. Мелешко Є.В. Загрози інформаційній безпеці у рекомендаційних системах соціальних медіа // Збірник тез VIII Всеукраїнської науково-

практичної конференції «Безпека інформаційних технологій (ITSec 2018)», м. Київ, 16-18 травня 2018 р. – Київ: НАУ. – 2018. – С. 24-25.

37. Мелешко Є.В. Проблеми сучасних рекомендаційних систем та методи їх рішення // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 4 (50). – С. 120-124.

38. Мелешко Є.В., Хох В.Д., Улічев О.С. Дослідження відомих моделей атак на рекомендаційні системи з колаборативною фільтрацією // Збірник наукових праць Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – №. 5 (57). – С. 67-71.

39. Mobasher B., Burke R., Bhaumik R., Williams C. Effective attack models for shilling item-based collaborative filtering system // In Proceedings of the WebKDD Workshop, held in conjuction with ACM SIGKDD 2005, Chicago, Illinois. – 2005. – 8 p.

40. Hill K. Facebook says it did 'a test' last year using people's locations to make friend suggestions // News web-site Splinternews.com. – 2016. – [Electronic resource] – Access mode: https://splinternews.com/facebook-says-it-did-a-test-last-year-using-peoples-loc-1793857952

РОЗДІЛ 3.
ІНФОРМАЦІЙНА БЕЗПЕКА РЕКОМЕНДАЦІЙНИХ СИСТЕМ

3.1. Базові моделі інформаційних атак на рекомендаційні системи

Основним видом атак на рекомендаційні системи є атаки ін’єкцією профілів, що мають декілька відомих базових моделей, зокрема, випадкову, середню та популярну моделі атаки [1-7].

Атаки ін’єкцією профілів – це основний тип атак на рекомендаційні системи, що полягає у створенні певної кількості профілів ботів (мережі ботів), які за допомогою узгоджених дій змінюють рейтинги, а як наслідок, і частоту потрапляння цільових елементів системи у списки рекомендацій. Також на етапі підготовки до атаки профілі ботів можуть збирати статистичні дані про вподобання користувачів, використовуючи, списки рекомендацій, які їм надає система, цей необов’язковий початковий етап атаки часто називають атакою зондом [7]. Щоб дізнатися вподобання цільової групи користувачів, профілі ботів заповнюють даними характерними для цієї групи.

Для атак ін’єкцією профілів завжди буде цільовий об’єкт та об’єкти для наповнення профілю бота [7, 8]. Рейтинг цільового об’єкту зловмиснику треба збільшити або зменшити, а нецільові об’єкти, будуть оцінюватися для наповнення профілю бота та намагання зробити його максимально схожим на профіль справжніх користувачів атакованої системи.

Зловмисник для здійснення впливу повинен досить точно імітувати дії звичайних користувачів, щоб не бути виявленим. А стійка до атак рекомендаційна система повинна працювати так, щоб результат від дій зловмисників був настільки малоефективним, щоб у них не було стимулів продовжувати атаки, а справжні користувачі продовжували одержувати релевантні невикривлені рекомендації.

Часто перед основною атакою зловмисники намагаються відтворити

портрет типового користувача певного сегменту користувачів системи, зібрати інформацію про його вподобання та типові особисті характеристики (демографічні дані тощо). З цією метою створюються профілі ботів, схожі на користувачів з потрібного сегменту, щоб одержати списки рекомендацій такі ж як одержують користувачі цього сегменту, а також створюються парсери для збору відкритих даних на веб-ресурсі.

Атакою на рекомендаційну систему будемо вважати узгоджені зусилля великої кількісті профілів щодо зміщення результатів її роботи таким чином, щоб деяка група користувачів або усі користувачі почали отримувати рекомендації, що суперечать їх потребам.

З точки зору зловмисника, найкраща атака проти рекомендаційної системи – це найбільший вплив на рейтинги за найменшу кількість зусиль з його боку.

Принцип дії інформаційних атак ін’єкцією профілів на рекомендаційну систему з колаборативною фільтрацією зображено на спрощеному прикладі на рис. 3.1.

Рис. 3.1. Принцип дії інформаційних атак ін’єкцією профілів на рекомендаційні системи з колаборативною фільтрацією

На рис. 3.1 зображено приклад частини бази даних рейтингів рекомендаційної системи, на яку здійснюється атака. Позитивні оцінки об’єктам позначено символом «+», негативні символом «-», а відсутні, які система буде намагатися спрогнозувати – «?». В даному випадку системі треба спрогнозувати оцінки користувачів 5 та 6 для об’єкту f. Якщо система спрогнозує позитивні оцінки, то об’єкт f з великою ймовірністю потрапить у списки рекомендацій даним користувачам. Тому боти виставляють позитивні оцінки об’єкту f, а іншим об’єктам системи виставляють оцінки схожі на ті, які виставили користувачі 5 та 6. З огляду на те, що звичайні користувачі, схожі на користувачів 5 і 6, які оцінювали об’єкт f, ставили йому негативні оцінки, то без дій зловмисників рекомендаційна система спрогнозувала б низьку оцінку даному об’єкту, і він не потрапив би до рекомендацій.

Для того, щоб нейтралізувати дану атаку треба визначити, які профілі є ботами, та не враховувати їх оцінки при формуванні списків рекомендацій.

Розглянемо наступну модель профілю бота, що атакує рекомендаційну систему:

Рис. 3.2. Модель профілю бота,
що атакує рекомендаційну систему

Як видно з рис. 3.2, профіль бота містить наступні типи оцінок:

– Оцінки об’єктам з множини I_f для імітації дій справжніх користувачів,

оцінки даним об’єктам зловмисник змінювати не прагне, а навпаки намагається підібрати для них значення максимально схожі на справжні для цільової групи користувачів, на яких він прагне впливати.

– Оцінки об’єктам з множини I_{ti}, це максимальні (чи близькі до них) оцінки у системі для цільових об’єктів, яким зловмисник прагне підвищити рейтинг.

– Оцінки об’єктам з множини I_{td}, це мінімальні (чи близькі до них) оцінки у системі для цільових об’єктів, яким зловмисник прагне знизити рейтинг.

Кількість цільових об’єктів у бота може варіюватися від 1 до K та міститися тільки у множині I_{ti} або тільки у множині I_{td}, чи в обох цих множинах, а кількість об’єктів для наповнення профілю – від 0 до N.

Дії ботів можуть дати результат тільки тоді, коли вся мережа ботів виставить оцінки всім цільовим об’єктам і при цьому боти не будуть виявлені та нейтралізовані.

Слід зазначити, що успішна інформаційна атака може збільшити різницю між прогнозованими та справжніми вподобаннями (тобто, знизити точність рекомендацій) у разі, якщо інформаційна атака накручує рейтинги об’єктам, які переважно не подобаються користувачам. Або не змінить точність роботи системи, у випадку коли рейтинги накручуються об’єктам, які переважно подобаються користувачам, і мета атаки – привернути до них більше уваги.

Розглянемо базові моделі атак на рекомендаційні системи з колаборативною фільтрацією.

Найперші моделі атак було запропоновано в [6] – це випадкові та середні моделі атак. Обидві ці моделі атак передбачають генерацію профілів ботів, що будуть випадковим чином виставляти оцінки об’єктам з множини I_f. В наступних роботах [2-5, 7] розглянуто також більш складні та інформаційноємкі атаки.

Розглянемо найбільш відомі та поширені моделі атак на рекомендаційні системи.

Випадкова атака (Random Attack)

У профілях ботів множина I_f буде заповнюватися оцінками для об'єктів, вибраних випадковим чином. Оцінки обраним об'єктам будуть підбиратися також випадковим чином, але так, щоб вони були близькі до глобальної середньої оцінки у системі, наприклад, буде використовуватися нормальний розподіл з математичним сподіванням, рівним глобальній середній оцінці. Цільовому об'єкту буте ставитися максимальна r_{max} або мінімальна оцінка r_{min}, в залежності від цілей атаки. Знання та зусилля, необхідні для здійснення такої атаки, є досить мінімальними – глобальну середню оцінку у багатьох системах можна легко дізнатися напряму або за допомогою опосередкованих даних. Ця атака не є особливо ефективною.

Середня атака (Average Attack)

Використовує індивідуальні середні значення оцінок кожного об'єкту для створення множини I_f. Інформації для даної атаки треба зібрати більше. Однак середня атака може бути успішною навіть при використанні невеликого набору елементів у I_f, що дозволяє зменшити кількість необхідної для збору інформації. Але ціною такого зменшення необхідних даних буде велика кількість профілів з практично однаковими множинами оцінених елементів, що буде, звичайно, легко виявити. Ця атака більш ефективна, ніж випадкова. Але вона практично неефективна для алгоритмів колаборативної фільтрації на основі моделі сусідства типу item-based.

Середня атака вимагає відносно великої кількості знань про статистику дій справжніх користувачів у системі. Розумний захист рекомендаційної системи від таких атак буде ускладнювати нападнику збір необхідних даних. Для обходу такого захисту використовуються інші атаки, для яких вимоги до кількості знань значно нижчі.

Розглянемо існуючі атаки, що вимагають менше знань, ніж середня атака, але працюють ефективніше, ніж випадкова атака.

Атака приєднання до більшості (Bandwagon Attack)

Мета цієї атаки – асоціювати атакований об’єкт з невеликою кількістю об’єктів, які часто оцінюються користувачами (назвемо їх широковідомими). Зловмисник створює профілі ботів, що містять у множині I_f оцінки широковідомим об’єктам. Такі профілі мають високу ймовірність бути схожими на велику кількість користувачів, оскільки широковідомі об’єкти – це ті, які оцінили багато користувачів. Дані для такої атаки одержати досить легко. Отже, серед широковідомих об’єктів випадковим чином обирається декілька. Цим об’єктам ставляться максимальні оцінки разом із цільовим об’єктом. Деякій частині об’єктів у множині I_f можуть ставитися випадкові оцінки, наприклад, як у випадковій атаці для того, щоб урізноманітнити профілі ботів. Це досить ефективна атака, але, як і середня, стає неефективною при використанні проти колаборативної фільтрації на основі моделі сусідства типу item-based.

Сегментна атака (Segment Attack)

Основна ідея даної атаки полягає у тому, щоб змінювати рейтинг об’єкту у цільовій групі користувачів з відомими або легко передбачуваними вподобаннями. Тобто, цільовому об’єкту рейтинг буде накручуватися тільки у певному сегменті користувачів, щоб він потрапляв у рекомендації тільки до них. Інакше, якщо цільовий об’єкт потрапить у рекомендації користувачам з інших сегментів, він може почати отримувати від них низькі оцінки, яких буде більше, ніж накручених оцінок. Щоб здійснити таку атаку треба знайти реальних користувачів, які належать до цільового сегменту та зібрати дані про оцінки, які вони зазвичай виставляють об’єктам системи. Як і в атаці приєднання до більшості, зазвичай визначається, які об’єкти в цільовому сегменті є широковідомими. Цим об’єктам присвоюється максимальне значення оцінки разом із цільовим об’єктом. Щоб забезпечити максимальний ефект від атаки, деякі об’єкти для множини I_f обираються випадково та одержують мінімальні оцінки, що дозволяє зробити профілі ботів різними. Дана

атака є ефективною проти алгоритмів колаборативної фільтрації на основі моделі сусідства типу item-based.

Слід зазначити, що усі розглянуті вище моделі атак можуть використовуватися для пониження рейтингу об’єкту, але існують спеціалізовані атаки, які працюють краще, ніж інші, саме для пониження рейтингів.

Розглянемо моделі атак призначені для пониження рейтингів об’єктів рекомендаційної системи.

Атака любов/ненависть (Love/Hate Attack)

Ця атака дуже проста – без вимог до знань. Цільовому об’єкту присвоюється мінімальна оцінка r_{min}, а об’єкти для множини I_f обираються випадковим чином та одержують максимальні оцінки r_{max}. Незважаючи на надзвичайну простоту, це одна з найефективніших атак на пониження рейтингу проти алгоритмів колаборативної фільтрації на основі моделі сусідства типу user-based.

Атака обернена приєднанню до більшості (Reverse Bandwagon Attack)

Це варіант атаки приєднання до більшості, описаний вище, в якому для I_f вибираються широковідомі об’єкти, яким переважна більшість користувачів ставить низькі оцінки. Цим об’єктам у профілях ботів присвоюються низькі оцінки, а також низька оцінка присвоюється цільовому об’єкту. Таким чином, цільовий об’єкт починає асоціюватися з об’єктами, що не подобаються великій кількості користувачів, і це збільшує ймовірність того, що для об’єкта будуть прогнозуватися низькі оцінки і він не буде потрапляти у списки рекомендацій. Хоча ця атака не є настільки ефективною, як середня атака з великою кількістю знань для user-based систем, вона є дуже ефективною атакою на пониження рейтингів проти item-based систем.

Атаки з низьким рівнем знань використовують широковідомі об’єкти для наповнення профіля бота оцінками для них. Таким чином зловмисник може створити профіль схожий на середньостатистичного користувача, дослідивши оцінки лише широковідомих об’єктів.

Якщо зловмисник знає, який саме алгоритм використовує рекомендаційна система, він може зібрати більше інформації для атаки.

Таким чином атаки можна класифікувати на:

– Атаки з малою кількістю знань – цей тип атак не потребує детальних знань про розподіли оцінок у системі. Він вимагає системно-незалежних знань, які легко можна отримати за допомогою публічних джерел інформації.

– Атаки з великою кількістю знань – зловмиснику потрібно мати якнайбільше знань про алгоритми системи та розподіли оцінок у об'єктів системи. Наприклад, деякі атаки вимагають, щоб зловмисник знав середню оцінку і середнє квадратичне відхилення для кожного об'єкта системи.

Приладом атаки з великою кількістю знань є популярна атака.

Популярна атака (Popular Attack)

Припустимо, що система використовує стандартний user-based алгоритм колаборативної фільтрації, де подібність між користувачами визначається за допомогою кореляції Пірсона. Аналогічним чином, як і в атаці приєднання до більшості, множина I_f заповнюється з використанням широковідомих об'єктів системи. Однак це не гарантує високої схожості між профілем бота та справжніми профілями. Тому популярна атака використовує середні значення оцінок обраних широковідомих об'єктів для заповнення множини I_f. А для того, щоб урізноманітнити дані профілів, деяким випадковим чином обраним об'єктам ставить оцінки I_f або (r_{min} + 1), або r_{min}, залежно від того, чи є середня оцінка для об'єкту вищою чи нижчою. Така стратегія призведе до позитивних кореляцій між профілями ботів та автентичними профілями. Для визначення широковідомих об'єктів не потрібно багато знань, але для визначення середніх оцінок обраних об'єктів треба зібрати багато інформації. Популярну атаку можна легко налаштувати також для атак на пониження рейтингу. Цю атаку можна виявляти, якщо порівнювати профілі у системі – профілі ботів однієї бот-мережі будуть сильно схожими.

Атака зондом для зібрання інформації (Probe Attack)

Профілі ботів тим важче розпізнати, чим більш схожі їх оцінки на оцінки справжніх користувачів. Знання про реальні вподобання різних сегментів користувачів можна отримати із самої системи через атаку зондом. Для здійснення цієї атаки зловмисник створює насіннєвий профіль, а потім використовує його для одержання рекомендацій з системи. Ці рекомендації формуються на основі інформації системи про реальних користувачів, тому використання одержаних рекомендацій дозволить створити профілі ботів більш схожі на справжніх користувачів. Можна здійснювати зондування невеликої частини користувачів, щоб потім вплинути на малу групу, як у сегментній атаці, або великої частини – щоб одержати інформацію, наприклад, для середньої атаки. Зловмиснику потрібно використовувати лише невелику кількість насіннєвих профілів для того щоб рекомендаційна система сама надала йому потрібну інформацію у вигляді рекомендацій.

Проведені дослідження моделей інформаційних атак на рекомендаційні системи показали, що найбільш простими в реалізації та найменш ресурсозатратними є випадкова та середня атаки, а найбільш ефективною та непомітною, хоча й досить ресурсозатратною, є популярна атака. Інші досліджені моделі атак, по суті, являються їх модифікаціями, наприклад, для атаки лише певного сегменту (сегментна атака) або для атаки лише на пониження рейтингів (атака любов/ненависть), або для зменшення ресурсозатратності за рахунок зменшення непомітності бот-мережі (атака приєднання до більшості, атака любов/ненависть). Тому в програмній імітаційній моделі користувачів та об’єктів рекомендаційної мережі було вирішено моделювати бот-мережі на основі трьох основних моделей атак – випадкової, середньої та популярної атаки.

3.2. Основні підходи до виявлення інформаційних атак на рекомендаційні системи

Існуючі методи виявлення інформаційних атак на рекомендаційні системи базуються на встановленні факту наявності групи профілів користувачів, що здійснюють атаку, тобто, ідентифікації профілів ботів [2, 6-10]. Після виявлення профілів ботів, можна вилучити їх інформацію з бази даних, що використовується для формування списків рекомендацій. Таким чином поставлені ними оцінки та виконані дії (перегляди, коментарі тощо) не будуть впливати на рейтинги об'єктів системи.

В такому випадку виявлення атаки на рекомендаційну систему можна розглядати як задачу бінарної класифікації профілів системи [6, 7, 10] з двома можливими результатами для кожного профілю, а саме:

– Профіль справжнього користувача (Authentic);

– Профіль бота, створеного для атаки на систему (Attack).

Для створення такого класифікатору можна використовувати різні методи машинного навчання, які навчати на виборці, що містить як профілі аутентичних користувачів, так і профілі ботів.

Якщо класифікатор використовує алгоритм навчання з учителем, то на етапі навчання використовується анотований набір даних профілів, тобто, набір профілів, позначених як Authentic або Attack. Оскільки більшість моделей атак використовують групи профілів ботів, які працюють узгоджено, є корисним розглядати саме групи профілів разом під час класифікації.

Якість роботи такого класифікатору можна оцінювати за допомогою стандартних метрик, таких як точність позитивного прогнозу (3.1) та повнота (3.2):

$$precision = \frac{tp}{tp + fp}, \tag{3.1}$$

де *tp* – правильна класифікація профілю як Attack; *fp* – неправильна класифікація профілю як Attack.

$$recall = \frac{tp}{tp + fn}, \tag{3.2}$$

де *tp* – правильна класифікація профілю як Attack; *fn* – неправильна класифікація профілю як Authentic.

Також корисними будуть й інші метрики, такі як точність негативного прогнозу (3.3) та специфічність (3.4):

$$NPV = \frac{tn}{tn + fn}, \tag{3.3}$$

де *tn* – правильна класифікація профілю як Authentic; *fn* – неправильна класифікація профілю як Authentic.

$$specificity = \frac{tn}{tn + fp}, \tag{3.4}$$

де *tn* – правильна класифікація профілю як Authentic; *fp* – неправильна класифікація профілю як Attack.

Неправильна класифікація аутентичних профілів призводить до вилучення користної достовірної інформації із бази даних рекомендаційної системи, що може негативно вплинути на загальну якість її роботи. Один із способів оцінити цей вплив – обчислити MAE системи до та після виявлення атаки та нейтралізації профілів ботів.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\left|y(t) - \hat{y}(t)\right|. \tag{3.5}$$

Ефективність нейтралізації атаки можна оцінити за допомогою зсуву прогнозування рейтингів цільового об’єкту до та після виявлення атаки і вилучення профілів ботів з процесу обчислень рекомендаційної системи, наприклад, з використанням формули (2.20).

Виявлення профілю бота

Розподіл оцінок у профілі бота з високою ймовірністю буде відрізнятися від розподілу оцінок у профілях справжніх користувачів. Незважаючи на те, що зловмиснику вигідно створювати профілі ботів якомога більш схожими на профілі аутентичних користувачів системи, у нього ніколи не може бути достатньо інформації та ресурсів для повного усунення відмінностей між ботами та звичайними користувачами.

Ознаками профілю бота можуть бути наступні статистичні особливості [6, 7]: відхилення від середнього значення оцінок є більшим, ніж зазвичай; деяка група профілів, має вищу, ніж зазвичай, подобу до профілю, який перевіряється.

Відхилення від середнього значення оцінок можна оцінити за допомогою наступного показника – відхилення оцінок від середньої угоди (RDMA):

$$RDMA_u = \sum_{i=0}^{n_u} \frac{\left| \frac{r_{u,i} - \bar{r}_i}{l_i} \right|}{n_u}, \tag{3.6}$$

де n_u – кількість об’єктів, які оцінив користувач u; r_u – оцінка, яку поставив користувач u елементу i; l_i – кількість оцінок, виставлених об’єкту i всіма користувачами; $\bar{r}_i$ – середнє значення усіх оцінок об’єкту i.

Ступінь подібності з топ-сусідами дозволяє виявляти цілі групи ботів, оскільки профіль бота для деяких моделей атак може бути сильніше схожий на профілі найбільш схожих на нього користувачів, ніж це відбувається з профілями аутентичних користувачів:

$$DegSim_u = \sum_{v=1}^{k} \frac{sim_{u,v}}{k}, \tag{3.7}$$

де $sim_{u,v}$ – коефіцієнт подоби між користувачами u та v.

Виявлення групи профілів ботів

Як правило, ці алгоритми використовують кластеризацію профілів системи та намагаються відрізнити кластери профілів атаки від кластерів аутентичних

профілів.

Основні методи виявлення груп ботів передбачають розділення профілів користувачів на два кластери на основі їх коефіцієнтів подоби [7, 11]. Аналізуючи статистику кластерів приймається рішення про наявність нападу і, якщо так, то який кластер містить профілі атаки. Усі профілі кластера атаки ігноруються у процесі формування рекомендацій. Вважається, що напад відбувся, якщо різниця в статистичних особливостях профілів для двох кластерів досить велика. Кластер з меншим стандартним відхиленням визначається як кластер ботів. Однак, особливо для невеликих за розміром бот-мереж, значна частина кластеру, ідентифікованого як кластер ботів, може містити аутентичних користувачів.

Використання мереж репутації користувачів для визначення рівня довіри до даних їх профілів

Для боротьби з інформаційними атаками на рекомендаціні системи може бути корисним додавання параметру репутації для користувачів системи [5, 7, 12-15]. Такий параметр можна додати в рекомендаційну систему різними способами, наприклад, через створення репутаційної системи [14, 16, 17], коли користувачі можуть оцінювати рівень довіри один до одного, що часто застосовується на веб-сайтах дошок оголошень та маркетплейсів. Також іншим способом введення параметру репутації у рекомендаційну систему є автоматичне його обчислення на основі статистики дій користувача [12], якщо можна ідентифікувати деяку множину його дій як шкідливу чи корисну для системи.

При наявності параметру репутація, можна використовувати коефіцієнти репутації та зважувати вклад кожного користувача в прогноз рейтингів при формуванні рекомендацій.

Тож найпростіший спосіб визначення репутації користувача – зробити на веб-ресурсі можливість оцінювати його. Таким шляхом ідуть Інтернет-

магазини, дошки об’яв, ресурси з оренди житла, де можна виставляти оцінки та переглядати загальні рейтинги продавців та покупців.

Визначена таким чином репутація може використовуватися рекомендаційною системою, але вона сама по собі вразлива до інформаційних атак, таку репутацію можна підробити діями ботів. А також вона лише опосередковано може вказувати на нечесність користувача у виставлені оцінок об’єктам системи, адже така репутація є індикатором чесності/нечесності поведінки в інших діях.

Також цей параметр може визначатися за допомогою різних алгоритмів на основі статистики дій користувачів, наприклад, значення репутації підвищується, коли користувач оцінює об’єкт більш правдоподібно (для кластеру, у якому знаходиться) та зменшується, коли користувач неправдиво оцінює об’єкт (його оцінка протилежна до оцінок користувачів з його кластеру) [7]. В такій системі для користувача, що прагне максимізувати свій вплив на рекомендації іншим користувачам, доцільно чесно оцінювати об’єкти.

При використанні параметру репутації у рекомендаційних системах та врахуванні його при створенні рекомендацій боти будуть одержувати нижчі значення даного параметру, ніж аутентичні користувачі, тож дані їх профілів будуть мати менший вплив на формування списків рекомендацій або взагалі не будуть враховуватися.

3.3. Виявлення інформаційних атак на рекомендаційну систему на основі аналізу трендів рейтингів об’єктів системи

Наявність інформаційної атаки на рекомендаційну систему викликає зміну рейтингів деякої групи об’єктів (підвищує або знижує їх), але зміна рейтингів об’єктів не достатня причина, щоб вважати, що відбувається атака, так як рейтинги можуть змінюватися і внаслідок дій автентичних користувачів. Тому

можна виділити ряд додаткових ознак, що можуть вказувати на інформаційну атаку [18], зокрема:

– у об’єкта, на досліджуваному невеликому проміжку часу зросла кількість оцінок – це може вказувати на інформаційну атаку, адже, щоб зсунути рейтинги треба створити суттєву кількість ботів, які поставлять цільові оцінки об’єкту.

– об’єкту, на досліджуваному проміжку часу поставили цільові оцінки (найвищі або найнижчі у шкалі оцінювання, в залежності типу атаки, який підозрюється) – адже бот-мережа змінює рейтинги виставленням саме цільових оцінок; також, більш підозріло, якщо об’єкту виставили дуже багато цільових оцінок, а особливо якщо більше, ніж усіх інших оцінок в поточному проміжку часу, що може вказувати на поспіх атакуючого систему, коли йому треба якнайшвидше просунути свій контент.

– велика дисперсія оцінок об’єкту – природньо, що у об’єкту буде багато різних оцінок, адже при атаці намагаються змінити рейтинги, тобто, наприклад, до атаки було багато низьких оцінок, а бот-мережа виставляє багато високих оцінок.

– об’єкт багато разів потрапив у списки рекомендацій на досліджуваному проміжку часу, більше разів, ніж потрапляв раніше – якщо це атака, то вона успішна, а тільки успішні атаки нас і мають цікавити, тому що їх треба нейтралізувати, а неуспішні атаки можна і проігнорувати у випадку економії ресурсів системи.

У [18] запропонована множина показників, за значеннями яких можна визначити наявність чи відсутність інформаційної атаки на об’єкт рекомендаційної системи, яка має наступний вигляд:

$$Q_{a,i} = \{tr, pr, d_r, d_t, n_r, n_{tr}, n_{rec}\}, \quad (3.8)$$

де *tr* – тренд динаміки рейтингів об’єкту *i*, що може приймати наступні значення {–1, 0, 1} – відповідно «тренд зменшення рейтингу», «немає змін» та «тренд збільшення рейтингу»; *pr* – прогнозування тренду динаміки рейтингів

об’єкту i, наприклад, на основі показника Херста [19]; d_r – дисперсія рейтигів об’єкту i; d_t – дисперсія часу виставлення оцінок об’єкту i; n_r – кількість оцінок у об’єкту i на досліджуваному проміжку часу; n_{tr} – кількість цільових оцінок у об’єкту i на досліджуваному проміжку часу; n_{rec} – кількість потраплянь об’єкту i у списки рекомендацій користувачам на досліджуваному проміжку часу.

Будемо вважати, що атака на рекомендаційну систему відбувається при цілеспрямованій зміні рейтингів у одного або декількох об’єктів системи згуртованими діями групи профілів користувачів системи. При чому розмір шкоди від атаки не завжди залежить від кількості об’єктів. Успішна зміна рейтингів навіть одного об’єкту може мати великі наслідки, якщо об’єкт важливий і мова йде, наприклад, про соціальну, політичну чи медичну сферу тощо.

Якщо об’єкти системи можна ранжувати за ступенем їх важливості та потреби у захисті від інформаційних атак, то їм можна призначити відповідні коефіцієнти. І в першу чергу відслідковувати стан та динаміку рейтингів найбільш важливих об’єктів, ігноруючи або в останню чергу відслідковуючи стан рейтингів менш важливих об’єктів.

Усі показники з (3.8) визначити легко, окрім одного – тренду рейтингів. Оскільки визначенням трендів займається економічна наука, наприклад, для визначення змін курсу валют, акцій тощо – звернемося до її методів.

Для того, щоб визначити наявність та напрямок тренду у рейтингах об’єкту рекомендаційної системи можна скористатися одним з наступних методів технічного аналізу [20, 21, 22]:

- За ковзним середнім;
- За декількома ковзними середніми;
- За вершинами Зиґзаґу;
- За свідченнями ADX;
- За NRTR;
- За кольором свічок Heiken Ashi.

Оскільки найбільш універсальними є перші три методи визначення наявності та напрямку тренду, що дозволить адаптувати їх до задачі пошуку трендів у рейтингах об’єктів рекомендаційної системи, розглянемо їх детальніше нижче.

Визначення напрямку тренду за ковзним середнім

Один з найпростіших способів визначення наявності тренду і його напрямку – за ковзним середнім. Можна використовувати як одне ковзне середнє, так і цілий набір, який іноді називають "віялом".

Правило визначення тренду для одного ковзного середнього:

– тренд спрямований вгору, якщо на заданому проміжку часу останнє значення числового ряду знаходиться вище ковзного середнього;

– тренд спрямований вниз, якщо на заданому проміжку часу останнє значення числового ряду знаходиться нижче ковзного середнього.

Коли на заданому проміжку часу останнє значення числового ряду вище/нижче ковзного середнього, то наступне значення часто після цього розгортається в протилежному напрямку. Тобто, даний спосіб дає велику кількість хибних відповідей. Через це використання його як індикатора тренду вельми обмежене. Його можна використовувати тільки в якості самого грубого фільтру тренду.

Визначення напрямку тренду за декількома ковзними середніми

Щоб поліпшити якість визначення тренду за ковзним середнім можна, наприклад, використовувати два і більше ковзних середніх з різними періодами. Тоді правило визначення тренду для будь-якого числа (більше, ніж одного) ковзних середніх з різними періодами буде виглядати так:

– тренд спрямований вгору, якщо на заданому проміжку часу всі ковзні середні збудовані в правильному порядку підвищення до кінця числового ряду;

– тренд спрямований вниз, якщо на заданому проміжку часу всі ковзні середні збудовані в правильному порядку зниження до кінця числового ряду.

У даному методі число помилкових сигналів про зміну напрямку тренду

буде меншим, ніж у попередньому. Але збільшаться часові затрати на визначення тренду.

Визначення напрямку тренду за максимумами та мінімумами індикатору Зиґзаґ

У даному методі використовується правило Чарльза Доу [20]:

– тренд спрямований вгору, якщо кожний наступний локальний максимум графіку числового ряду вище попереднього локального максимуму і, при цьому, кожен наступний локальний мінімум графіку числового ряду також вище попереднього локального мінімуму;

– тренд спрямований вниз, якщо кожний наступний локальний мінімум графіку числового ряду нижче попереднього локального мінімуму і, при цьому, кожен наступний локальний максимум графіку числового ряду також нижче попереднього локального максимуму.

Локальні максимуми/мінімуми можна знаходити за вершинами індикатору Зиґзаґ.

Індикатор Зиґзаґ (ZigZag) – це індикатор тренду у технічному аналізі, що з'єднує локальні мінімуми і максимуми на графіку числового ряду і дозволяє фільтрувати шум. Існує багато різноманітних модифікацій індикатору ZigZag для аналізу фондових ринків, що враховують різні вподобання трейдерів.

Параметр мінімальної зміни значення часового ряду визначає кількість пунктів, на які значення повинне переміститися, щоб сформувати нову "Зіґ" або "Заґ" лінію. Таким чином, Зиґзаґ відображає тільки найбільш значущі зміни і розвороти.

Головний недолік цього способу визначення тренду – в реальному часі неможливо зрозуміти утворився вже екстремум або ще ні.

Цей недолік робить даний спосіб малоцінним для практичного використання в реальному часі. Зате він дуже корисний при технічному аналізі зібраних раніше даних з метою пошуку закономірностей і для оцінки якості роботи системи.

У [2] пропонується для визначення наявності тренду рейтингу об’єкту використати декілька ковзних середніх. Також пропонується здійснювати прогнозування динаміки трендів рейтингів об’єктів системи у найближчому майбутньому. Прогноз збереження виявлених трендів у майбутньому може бути одною з ознак успішної атаки. Для прогнозування динаміки трендів рейтингів об’єктів було використано *R/S-аналіз* [21].

Алгоритм R/S-аналізу складається з наступних кроків [21]:

1. Дано початковий часовий ряд S_t. Розраховуємо логарифмічне відношення:

$$N_t = \ln \frac{S_t}{S_{t-1}} \tag{3.9}$$

2. Розділимо ряд N на A суміжних періодів довжиною n. Позначимо кожен період як I_a, де $a = 1, 2, \ldots, A$. Визначимо для кожного I_a середнє значення:

$$E(I_a) = \frac{1}{n} \sum_{k=1}^{n} N_{k,a} \tag{3.10}$$

3. Розраховуємо відхилення від середнього значення для кожного періоду I_a:

$$X_{k,a} = \sum_{i=1}^{k} \left(N_{i,a} - E(I_a) \right) \tag{3.11}$$

4. Розрахуємо розмах в межах кожного періоду:

$$R_{I_a} = \max\left(X_{k,a}\right) - \min\left(X_{k,a}\right) \tag{3.12}$$

5. Розрахуємо стандартне відхилення для кожного періоду I_a:

$$S_{I_a} = \sqrt{\frac{1}{n} \sum_{k=1}^{n} \left(N_{k,a} - E(I_a) \right)^2} \tag{3.13}$$

6. Кожен R_{I_a} ділимо на S_{I_a}. Далі розраховуємо середнє значення *R/S*:

$$R/S(n) = \frac{\sum_{a=1}^{A} R_{I_a} / S_{I_a}}{A} \tag{3.14}$$

7. Збільшуємо n і повторюємо кроки 2-6 до тих пір, поки $n \le N/2$.

8. Будуємо графік залежності $\log(R/S(n))$ від $\log(n)$ і за допомогою методу найменших квадратів знаходимо регресію виду: $\log(R/S(n)) = H \cdot \log(n) + c$, де H – показник Херста.

9. Далі перевіряємо отриманий результат на значимість. Для цього перевіряємо гіпотезу про те, що аналізована структура є нормально-розподіленою. Якщо *R/S* є випадковими змінними, нормально розподіленими, тоді можна припустити, що H також розподілені нормально. Асимптотичною границею для незалежного процесу є показник Херста рівний 0.5. Еніс і Ллойд [23], а також Петерс [21] запропонували використовувати такі очікувані показники *R/S*:

$$E(R/S(n)) = \frac{n-0.5}{n} \cdot \left(n \cdot \frac{\pi}{2}\right)^{-0.5} \cdot \sum_{r=1}^{n-1} \sqrt{\frac{n-r}{r}} \qquad (3.15)$$

Для n спостережень знаходимо очікуваний показник Херста $E(H)$.

10. Розраховуємо очікувану дисперсію показника Херста за формулою:

$$Variance(H) = \frac{1}{N} \qquad (3.16)$$

де H – показник Херста; N – число спостережень у вибірці.

11. Перевіряємо значимість отриманого коефіцієнта Херста шляхом оцінки кількості стандартних відхилень, на які H перевершує $E(H)$. Значущим вважається результат, коли показник значущості по модулю більше 2.

Інтерпретація показників індексу Херста [19]:

– $H = 0.5$ – процес з відсутністю пам'яті, певного тренду немає.

– $H > 0.5$ – процес характеризується персистентністю – має тенденцію до збереження тренду.

– $H < 0.5$ – процес характеризується антіперсистентністю – будь-яку тенденцію прагне змінити протилежна.

Значення показника Херста природних процесів групуються поблизу наступних значень [19]: 0.72-0.73.

При дослідженні стану рекомендаційної системи є сенс звертати увагу на об’єкти, у яких $H > 0.5$ на досліджуваному проміжку часу, такі об’єкти будуть змінювати свої рейтинги відповідно певного тренду на протязі довгого проміжку часу, отже, серед них можуть бути цілі успішних інформаційних атак, особливо якщо $H > 0.73$.

У [18] було розроблено метод виявлення об’єктів інформаційної атаки бот-мережі ін’єкцією профілів у рекомендаційній системі, що складається з наступних етапів:

Етап 1. Формуємо множину об’єктів I для перевірки, вона може містити всі об’єкти рекомендаційної системи або тільки критично важливі об’єкти, що потребують захисту від інформаційних атак.

Етап 2. Визначаємо за допомогою декількох ковзних середніх (або інших методів визначення тренду) для кожного об’єкту з множини I показник tr на проміжку часу τ.

Етап 3. Визначаємо за допомогою R/S-аналізу для кожного об’єкту з множини I індекс Херста H на проміжку часу τ.

Етап 4. Визначаємо для кожного об’єкту з множини I на проміжку часу τ дисперсію оцінок d_r та дисперсію часових проміжків між виставленням цільових оцінок d_t, а також їх середньостатистичні значення у системі $d_{\tau,cep}$ та $d_{t,cep}$.

Етап 5. Визначаємо для кожного об’єкту з множини I на проміжку часу τ кількість виставлених йому цільових n_{tg} та усіх оцінок n_r, а також середньостатистичну кількість цільових $n_{tg,cep}$ та усіх оцінок $n_{r,cep}$ для об’єктів системи.

Етап 6. Визначаємо для кожного об’єкту з множини I на проміжку часу τ кількість потраплянь у списки рекомендацій n_{rec}, а також середньостатистичну кількість потраплянь у списки рекомендацій $n_{rec,cep}$ для всіх об’єктів системи.

Етап 7. Визначаємо наявність та тип атаки за наступними правилами:

Правило 1. Якщо у об’єкта наявні будь-які 5 ознак з даних: тренд на зростання рейтингу $tr_\tau = 1$, $H > 0.73$, $d_\tau \le d_{\tau,cep}$, $d_t \le d_{t,cep}$, $n_r > n_{r,cep}$, $n_{tg} > n_{tg,cep}$, $n_{rec} > n_{rec,cep}$, то вважаємо, що існує висока ймовірність наявності атаки на підвищення рейтингу для цього об’єкту.

Правило 2. Якщо у об’єкта наявні будь-які 5 ознак з даних: тренд на зменшення рейтингу: $tr_\tau = -1$, $H > 0.73$, $d_\tau \le d_{\tau,cep}$, $d_t \le d_{t,cep}$, $n_r > n_{r,cep}$, $n_{tg} > n_{tg,cep}$, $n_{rec} < n_{rec,cep}$, то вважаємо, що існує висока ймовірність наявності атаки на зниження рейтингу для цього об’єкту.

З об’єктів, у яких наявні ознаки інформаційної атаки, можна сформувати множину I_g. Це дозволить при пошуку бот-мереж методами кластеризації та статистичного аналізу профілів користувачів перевіряти не всі профілі системи, а тільки ті, які взаємодіяли з об’єктами множини I_g. Після знаходження бот-мережі можна точніше визначити I_g, виходячи з аналізу діяльності ідентифікованих профілів ботів.

3.4. Виявлення та нейтралізація мережі ботів у рекомендаційній системі

Якщо у системі виявлені об’єкти з аномальною зміною трендів рейтингів, що гіпотетично могло виникнути внаслідок атаки бот-мережі, логічно наступним кроком зробити спробу виявлення профілів користувачів, що вплинули своїми діями на цю зміну та спробувати з’ясувати чи поєднані вони у скоординовану мережу або мережі.

Природно, що на зміну трендів рейтингів об’єктів вплинули усі користувачі, що поставили їм оцінки рівні цільовим – високі, якщо у об’єкту зросли рейтинги, або низькі, якщо у об’єкту знизилися рейтинги. Множину таких користувачів легко виявити простими запитами до бази даних рекомендаційної системи. А от визначити, які профілі користувачів дійсно є частиною бот-мережі та виконують скоординовані дії з іншими її учасниками, а

які просто поставили оцінку, що відповідала їх вподобанням – значно складніше.

Задачу виділення бот-мережі серед профілів користувачів системи можна звести до задачі пошуку підграфу у соціальному графі рекомендаційної системи, вершинами якого будуть профілі користувачів пов’язані між собою деякими спільними діями, що мали вплив на зміну трендів рейтингів усіх або більшості об’єктів з множини ймовірних цілей інформаційної атаки [1, 7, 24, 25]. Тому перед розглядом методу виявлення мережі ботів наведемо існуючі методи графової кластеризації та проведемо тестування їх роботи на даних рекомендаційної мережі.

Методи кластеризації графів, які також називають методами пошуку співтовариств (якщо їх застосовують до соціальних мереж) за принципом роботи можна поділити на наступні: засновані на оптимізації модулярності, засновані на спектральних особливостях графу та засновані на оцінці ентропії системи [26-30]. За результатами роботи методи пошуку співтовариств можна поділити на такі, що розбивають граф на кластери, які не перетинаються (наприклад, Edge Betweenness, Label Propagation, FastGreedy, WalkTrap, Infomap, Leading Eigenvector, MultiLevel тощо), та на ті, що розбивають граф на кластери, які перетинаються (наприклад, k-Clique Perlocation, BigCLAM, DEMON, CONGO тощо) [31].

Для розділення користувачів рекомендаційної системи на аутентичних та ботів слід застосувати методи, що розбивають граф на кластери, які не перетинаються, адже в даному разі профіль не може потрапляти одразу до двох категорій. Було проведено дослідження найбільш відомих та часто використовуваних методів даного типу, наведене нижче.

Методи кластеризації графів, засновані на оптимізації модулярності

Переважна більшість алгоритмів кластеризації графів заснована на оптимізації модулярності [30, 31].

Модулярність – деяка числова характеристика, яка описує вираженість

структури кластерів в певному графі [29, 30, 31]. Для оцінки модулярності можна використовувати формулу:

$$Q = \frac{1}{2n_e}\sum_{ij}(A_{ij} - \frac{d_i d_j}{2n_e})\delta(C_i, C_j), \quad (3.17)$$

де n_e – кількість ребер у графі; A – матриця суміжності графу; d_i – кількість ребер, суміжних з вершиною i; d_j – кількість ребер, суміжних з вершиною j; $\delta(C_i, C_j)$ – дельта-функція, рівна одиниці, якщо $C_i = C_j$ та нулю в іншому випадку.

Дана величина дорівнює різниці між кількістю ребер всередині кластера при поточному розбитті і кількістю ребер, якби вони були випадково згенеровані [30].

Значення модулярності показує вираженість кластерів, вона буде:

– рівна одиниці для повного графу, в якому всі вершини помістили в один кластер;

– рівна нулю для розбиття на кластери, при якому кожна вершина знаходиться в окремому кластері;

– для невдалих розбиттів модулярність може приймати негативне значення.

По-суті, за допомогою значення модулярності можна оцінити якість розбиття графів на кластери. Якісне розбиття характеризується тим, що кількість внутрішніх зв’язків всередині кожного кластера має бути більша, ніж кількість його зовнішніх зв’язків.

Значення модулярності характеризує не те, наскільки для даного розбиття внутрішньо-кластерні зв'язки більш щільні, ніж міжкластерні, а те, наскільки вони більш щільні в порівнянні з деякою початковою щільністю. Тому відбувається порівняння з «нульовою гіпотезою», яка полягає у тому, що дуги розподілені випадково, тобто, немає закономірностей в розподілі щільності дуг всередині графу.

Принцип алгоритмів кластеризації графів, заснованих на оптимізації

модулярності, полягає у тому, що на кожному кроці алгоритму кожній вершині графу деяким чином ставиться у відповідність деякий кластер, обчислюється значення модулярності та здійснюється перерозподіл вершин графу між кластерами таким чином, щоб збільшити значення модулярності. Робота таких алгоритмів припиняється тоді, коли вже не можна покращити значення модулярності.

Розглянемо деякі найбільш відомі методи кластеризації графів на основі оптимізації модулярності.

Метод FastGreedy – полягає в жадібній оптимізації модулярності [31, 32]. На першому кроці методу кожній вершині графу ставиться у відповідність окремий кластер, а потім об'єднуються такі пари кластерів, об’єднання яких призводить до максимального збільшення модулярності. При цьому об'єднуються тільки інциденті пари вершин, тому що інакше модулярність не може збільшитися. Ітерації об’єднання кластерів продовжуються поки продовжує збільшуватися значення модулярності після них.

Метод Louvain (або **Multilevel**) – заснований на багаторівневій оптимізації функції модулярності [31, 33]. Більш якісно розбиває граф на кластертери порівняно з попереднім методом. Даний метод складається з двох частин. Перша частина, по суті, ідентична методу FastGreedy. Друга частина методу полягає у наступному: створюється новий граф з метавершинами у вигляді знайдених кластерів і ребрами з сумарною вагою всіх ребер, що йдуть від одного кластера до іншого (також створюються петлі з сумарними вагами зв'язків всередині кластера). Такий граф називається метаграф. Алгоритм знову запускається на новому графі. Це один з найбільш широковідомих методів за рахунок своєї швидкості роботи.

Метод Leading Eigenvector – спектральний метод, заснований на власних векторах матриці модулярності [31], яка визначається наступним способом:

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2n_e}, \tag{3.18}$$

де A – матриця суміжності графу; d_i – кількість ребер, суміжних з вершиною i; d_j – кількість ребер, суміжних з вершиною j; n_e – кількість ребер у графі.

Для даної матриці знаходиться перший власний вектор (з максимальним власним числом). Ті вершини, у яких відповідне значення менше нуля, належать одному кластеру, а де більше нуля – іншому. Подібним чином можливий поділ на більшу кількість кластерів.

Дані методи, засновані на оптимізації модулярності, є досить ефективними, але не найпростішими у реалізації.

Методи кластеризації графів, засновані на розмітці графів

Для пошуку кластерів у графі можна використовувати різні методи розмітки та розфарбовування графів. В основі даних методів лежить ідея, що вершина належить до того кластеру, до якого належить найбільша кількість її сусідніх вершин. В даних методах дуже важливий порядок перебору вершин. Методи розмітки графу допомагають перебрати вершини певним чином та з врахуванням їх сусідства визначитися з їх приналежністю до певних кластерів.

Розглянемо найвідоміший метод кластеризації графів з даної групи методів – LabelPropagation.

Метод LabelPropagation – розбиває граф на кластери наступним чином: кожна вершина у графі відноситься до того кластеру, якому належить більшість її сусідів, якщо ж таких кластерів декілька, то вибирається випадково один з них [30, 31, 33].

Розглянемо принцип роботи даного методу. У початковий момент часу всім вершинам ставиться у відповідність окремий кластер. Кожна вершина одержує мітку або колір відповідного кластеру. Потім для кожної вершини перевіряється, до яких кластерів належать її сусіди та здійснюється перерозподіл приналежності до кластерів. Через випадковості важливо на кожній ітерації змінювати порядок обходу вершин. Алгоритм закінчує роботу, коли уже нема чого змінювати – всі вершини відносяться до тих кластерів, що і більшість їх сусідів. Для покращення результатів можна застосувати наступну

хитрість – запустити алгоритм декілька разів, кожного разу зберігати результат його роботи та обрати найкращий варіант розбиття графу. Головна перевага даного алгоритму – майже лінійна складність. Недоліком алгоритму є те, що на зашумлених графах часто відбувається об'єднання всіх вершин в один кластер або невелику кількість кластерів.

Методи кластеризації графів, засновані на випадкових блуканнях

Для розбиття графу на кластери можна застосовувати алгоритми випадкового блукання.

Випадкове блукання – математична модель процесу випадкових змін – кроків в дискретні моменти часу. При цьому передбачається, що зміна на кожному кроці не залежить від попередніх змін і від часу.

Розглянемо найвідоміші методи кластеризації графів, засновані на випадкових блуканнях.

Метод Walktrap – використовує ідею про те, що короткі випадкові блукання не призводять до виходу з поточної спільноти (кластеру) [31, 32]. Відстань між вершинами або між групами вершин розраховують на основі ймовірності досяжності шляху від однієї вершини до іншої в процесі випадкового блукання. Даний показник є великим, якщо вершини знаходяться в різних кластерах в графі, і маленьким, якщо вони знаходяться в одному кластері. Далі здійснюється ієрархічна кластеризація агломеративним способом на основі методу Уорда: вершини об'єднуються у кластери на основі вибору найменшого середнього квадрата відстаней між ними. Після того, як вершина приєднана до якого-небудь кластеру, відстані між вершинами і кластерами перераховуються.

Метод Infomap – заснований на понятті інформаційних потоків в мережах, кодуванні і стисненні інформації [31]. В даному методі застосовується підхід, що базується на випадкових блуканнях та кодах Хаффмана. У кожної вершини є певна ймовірність її відвідування. За допомогою кодів Хаффмана, відповідно до цих ймовірностей, можна закодувати шлях блукання. Ця послідовність

матиме деяку довжину. Якщо використовувати ієрархічне кодування, можна скоротити довжину послідовності. Infomap ґрунтується на жадібному способі мінімізації довжини коду блукання.

Методи розбиття графів на кластери, що перетинаються

Розглянуті вище методи дозволяють розбити граф на кластери, що не перетинаються. В той же час для рішення деяких практичних задач може виникнути необхідність розбити граф на кластери, що можуть перетинатися між собою. В контексті задачі пошуку бот-мереж серед профілів користувачів рекомендаційної системи такі методи можуть знадобитися для виділення різних бот-мереж (серед множини профілів ботів), у яких є спільні та відмінні об’єкти у списках цілей для атак.

Розглянемо деякі методи, що дозволяють вирішити цю задачу.

Метод Clique perlocation method (CPM) – призначений для розбиття графу на кластери, що перетинаються, використовуючи особливості структури графу, а саме наявності *клік*. Починає роботу з пошуку всіх клік розміру l, після чого будується новий граф, вершинами якого є знайдені кліки [31, 32]. Ребро утворюється у разі, якщо перетин вершин-клік складається з $(l-1)$ вершин початкового графу. Компоненти зв'язності нового графу і будуть визначати знайдені кластери. Перевагою методу є його інтуїтивність та простота для розуміння. Недоліком методу є непридатність його використання на графах з дуже великою кількістю вершин. Слід пояснити термін кліка у теорії графів. ***Кліка*** – підмножина вершин неорієнтованого графу, будь-які дві з яких з'єднані ребром (рис. 3.3).

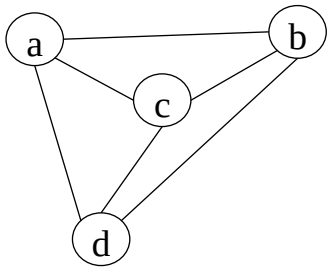


Рис. 3.3. Приклад структури типу *кліка* у графі

Метод Cluster Affiliation Model for Big Networks – ймовірнісна генеративна модель, що зводить задачу виділення кластерів до задачі факторизації невід'ємних матриць [31, 32]. Застосовується до дводольного графу, в одній частині якого знаходяться кластери, а в іншій – вершини, причому кожна вершина $v \in V$ не просто належить кластеру $c \in C$, а належить йому з деякою невід'ємною вагою. Задача оптимізації полягає у визначенні методом максимізації правдоподібності для даного графу оптимальної матриці зв'язків приналежності до кластерів. Для того, щоб відповісти на питання належить, чи не належить певний об'єкт до певного кластеру, до зв'язків подоби застосовується відсікання за пороговим значенням.

Метод Democratic Estimate of the Modular Organization of a Network (DEMON) – полягає у тому, що для кожної вершини графу будується его-мережа, потім для кожної такої его-мережі окремо застосовується алгоритм Label Propagation, у результаті якого отримуються розбиття на кластери $C_i(u_i)$, для кожного користувача u_i [31]. Усі такі покриття потім об'єднуються у загальне покриття C, яке на початку ініціалізувалося порожнім. Під час такого об'єднання два кластери об'єднуються в один тільки в тому разі, якщо не більше ε відсотків меншого з них не міститься в більшому з них, наприклад, для $\varepsilon = 0$ об'єднання буде відбуватися тільки, коли один з кластерів повністю міститься в іншому, а для $\varepsilon = 1$ об'єднання буде відбуватися завжди.

Розглянемо метод виявлення та нейтралізації мережі ботів у рекомендаційній системі на основі графової кластеризації та аналізу дій користувачів, запропонований у [18].

У вище було розглянуто метод визначення множини ймовірних цілей інформаційної атаки мережі ботів за допомогою аналізу трендів рейтингів об'єктів рекомендаційної системи. Позначимо таку множину можливих цілей атаки G.

Після того, як виявлена атака на рекомендаційну систему та сформована множина ймовірних цілей ботів G, логічним буде дослідити профілі всіх

користувачів, які вплинули на зміну трендів рейтингів об'єктів з цієї множини *G*, що і пропонується зробити у [18] для пошуку серед них профілів ботів.

Для вирішення задачі ідентифікації профілів ботів у [18] було запропоновано метод заснований на графовій кластеризації та аналізі дій користувачів, з використанням коефіцієнтів «недовіри», що складається з наступних етапів:

Етап 1. Формуємо множину підозрілих профілів користувачів *S*, в яку поміщаємо профілі, що ставили цільові оцінки r_t об'єктам з множини *G*.

Етап 2. Привласнюємо кожному користувачу з множини *G* мітку *:suspicious* та коефіцієнт недовіри, розрахований за наступною формулою:

$$k_{d,i} = \sum_{j \in G} \frac{E_{r_{t,i,j}}}{n_g}, \tag{3.19}$$

де $E_{r_{t,i,j}}$ – наявність цільової оцінки r_t від користувача *i* об'єкту *j*, що належить множині ймовірних цілей атаки *G*, приймає значення 1, якщо цільова оцінка є та 0 – при відсутності такої оцінки від користувача *i* об'єкту *j*; n_g – кількість об'єктів у множині можливих цілей атаки *G*.

Етап 3. Для кожної пари користувачів i_1 та i_2 з множини *S*, де $k_{d,i1} \geq q$ та $k_{d,i2} \geq q$, створюємо ребро між ними з міткою *:BotNet.*

Етап 4. Виконуємо графову кластеризацію для підграфу, що містить вершини з мітками *:User* і *:suspicious* та ребра з міткою *:BotNet.* Результати роботи такої кластеризації будуть наступними – усі боти потраплять до одного великого кластеру, якщо бот-мережа одна, або до декількох великих кластерів – якщо бот-мереж декілька; аутентичні користувачі потраплять у різні кластери, кожний з таких кластерів буде містити одного користувача або невелику кількість користувачів. Можливі й випадки, коли до кластеру з ботами потрапить деяка кількість аутентичних користувачів або деяка група схожих між собою та активних аутентичних користувачів утворить окремий кластер у разі, якщо їх дії зсунули рейтинги деяких об'єктів.

Етап 5. Визначаємо найбільші кластери, що складаються з $(N_{cr} - e)$ користувачів, де N_{cr} – мінімальна кількість користувачів, що може вплинути на результати роботи рекомендаційної системи (залежить від параметрів конкретної системи), e – приблизне значення похибки при розділенні профілів користувачів на кластери. Такий кластер (або кластери) вважаємо можливою бот-мережею (бот-мережами). У користувачів, що не потрапляють до даних кластерів прибираємо ребра з міткою *:BotNet*. Користувачів, що потрапили до підграфу *BotNet* треба додатково перевірити, проаналізувавши статистичні характеристики їх профілів, наприклад, за допомогою запропонованого у попередньому розділі способу з використанням нейронних мереж [34]. Також для додаткової перевірки профілів з підграфу *BotNet* можна здійснити пошук у них певних характерних для ботів ознак, наприклад, погано заповнених анкетних даних або надзвичайно високої активності (характерні особливості ботів залежать від конкретної системи і можуть ставати відомими в процесі збору статистичних даних під час її роботи). Одними з загальних ознак ботів для багатьох систем можуть бути: відмінність значення дисперсії оцінок та дисперсії часових інтервалів між виставленнями оцінок у профілях ботів від середньостатистичних значень відповідних дисперсій у профілях користувачів системи. Для конкретної системи ознаками ботів можуть бути: особливості реєстрації профілю, особливості наповнення профілю особистою інформацією, стиль написання та зміст коментарів, список друзів користувача тощо. Після перевірки статистичних даних окремих профілів користувачів, які потрапили у підграф *BotNet*, треба його скоректувати, видаливши з нього користувачів, розпізнаних за статистичними даними як аутентичні. Якщо є кластер, у якому всі користувачі визнані автентичними – слід перестати вважати його бот-мережею.

Етап 6. Коректуємо множину G після аналізу оцінок користувачів з підграфу *BotNet*. Слід перевірити, яким об’єктам користувачі з бот-мережі скоординовано виставляли цільові оцінки. Видаляємо з G об’єкти, які не

одержували взагалі або одержали незначний процент цільових оцінок від користувачів ідентифікованих як боти. Додаємо до множини *G* об'єкти, які одержали цільові оцінки від усіх ботів (або великого проценту ботів).

У [18] було запропоновано спосіб тестування методів виявлення ботів у рекомендаційних системах. Набори даних для експериментів генерувалися у програмній імітаційній моделі рекомендаційної системи запропонованій у [35], яка буде розглянута у наступному розділі. Формат та статистичні особливості даних генерувалися максимально наближеними до відповідних характеристик відкритого набору даних MovieLens Datasets [36]. Інформаційні атаки моделювалися за допомогою популярної моделі атаки [7, 37, 38]. У якості алгоритму графової кластеризації використовувався алгоритм Label Propagation.

У проведеній в [18] серії експериментів було згенеровано 10% ботів, усі інші користувачі системи – аутентичні. У різних експериментах у ботів була різна кількість цілей для атаки: 1, 5, 10, 15, 20, 25, 30, 35, 40 та 45 цілей. Також серед аутентичних користувачів було згенеровано 20% профілів з високим рівнем активності у рекомендаційній системі, які виставляли значно більше оцінок, ніж середньостатистичні користувачі. Об'єкти системи були згенеровані таким чином, щоб відноситися за своїми властивостями до одного з 19 кластерів.

Алгоритм пошуку ботів для досліджуваної рекомендаційної системи у [18] мав наступні параметри:

– $q = 0.05$ – тобто, користувач виставив цільові оцінки не менше 5% об'єктів, які визначені підсистемою інформаційної безпеки як ймовірні цілі атаки. Порогове значення невелике, тому що невідомо наскільки вірно розпізнані цілі атаки, можливо там багато об'єктів, у яких змінилися рейтинги природним чином.

– Відсіювання профілів користувачів з підграфу *BotNet* здійснюється на основі значень дисперсії оцінок та дисперсії часових інтервалів між виставленням оцінок. Аутентичними користувачами, які помилково потрапили

до підграфу *BotNet*, вважаються такі, дані з профілів яких відповідають наступному правилу:

$$|D_r - D_{r,avr}| < 0.15 \text{ AND } (D_{t,r} > 72 \text{ AND } D_{t,r} < 600),$$

де D_r – дисперсія оцінок у профілі користувача, $D_{r,avr}$ – усічене середнє дисперсії оцінок у профілі користувачів системи (відсікалося 30% крайніх значень, тобто, гранично можливий процент ботів у розглядуваній системі), $D_{t,r}$ – дисперсія часових інтервалів між виставленнями оцінок. Тобто, дисперсія оцінок у профілях аутентичних користувачів несуттєво відрізняється від середньостатистичної дисперсії (атакуючий систему це значення знати не може, він може лише приблизно його оцінити, тому не може вірно відтворити дану характеристику при створенні профілів ботів). А перевірка дисперсії часових інтервалів між виставленнями оцінок має наступний сенс – слід вважати підозрілими користувачів, які роблять занадто однакові або занадто різні інтервали між виставленнями оцінок, в той же час дисперсія часових проміжків між виставленнями оцінок у автентичних користувачів знаходяться, як правило, в певному діапазоні значень.

Контрольні питання

1. Яка основна мета інформаційних атак на рекомендаційні системи?
2. Що таке атака ін’єкцією профілів?
3. Як відрізняється поведінка ботів від звичайних користувачів рекомендаційної системи?
4. Які існують базові моделі атак на рекомендаційні системи?
5. Які існують основні підходи до виявлення інформаційних атак на рекомендаційні системи?
6. Як можна виявити та нейтралізувати мережу ботів, що здійснює атаку на рекомендаційну систему?
7. Як можна використати динаміку трендів рейтингів об’єктів для

виявлення атаки на рекомендаційну систему?

8. Які бувають способи визначення трендів рейтингів об’єктів рекомендаційної системи?

9. Як можна використати методи кластеризації графів для виявлення мереж ботів у рекомендаційних системах?

10. Які бувають методи кластеризації графів?

Список літератури до розділу

1. Мелешко Є.В., Хох В.Д., Улічев О.С. Дослідження відомих моделей атак на рекомендаційні системи з колаборативною фільтрацією // Збірник наукових праць Системи управління, навігації та зв’язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – №. 5 (57). – С. 67-71.

2. Chirita P.A., Nejdl W., Zamfir C. Preventing shilling attacks in online recommender systems // In Proceedings of the ACM Workshop on Web Information and Data Management. – 2005. – P. 67-74.

3. Kumari T., Punam B. A Comprehensive Study of Shilling Attacks in Recommender Systems // IJCSI International Journal of Computer Science Issues, Vol. 14, Issue 4. – 2017. – DOI: https://doi.org/10.20943/01201704.4450

4. Mobasher B., Burke R., Bhaumik R., Williams C. Effective attack models for shilling item-based collaborative filtering system // In Proceedings of the WebKDD Workshop, held in conjuction with ACM SIGKDD 2005, Chicago, Illinois. – 2005. – 8 p.

5. Mobasher B., Burke R., Bhaumik R.,Williams C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness // ACM Transactions on Internet Technology, Vol. 7(4). – 2007. – 41 p. – [Electronic resource] – Access mode: https://doi.org/10.1145/1278366.1278372

6. O’Mahony M.P., Hurley N.J., Silvestre G.C.M. Promoting recommendations: An attack on collaborative filtering // DEXA, Lecture Notes in

Computer Science, Vol. 2453. – 2002. – P. 494-503.

7. Ricci F., Rokach L., Shapira B., Kantor P.B. (Editors) Recommender Systems Handbook // Boston: Springer. – 2011. – 842 p. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-0-387-85820-3

8. Williams A. C., Mobasher B., Burke R. Defending recommender systems: detection of profile injection attacks // Service Oriented Computing and Applications. – 2007. – P. 157–170.

9. Zhou W., Wen J., Koh Y.S., Alam S., Dobbie G. Attack detection in recommender systems based on target item analysis // International Joint Conference on Neural Networks (IJCNN 2014), Beijing. – 2014. – P. 332-339. – [Electronic resource] – Access mode: https://ieeexplore.ieee.org/document/6889419

10. Zhou W., Wen J., Qu Q., Zeng J., Cheng T. Shilling attack detection for recommender systems based on credibility of group users and rating time series // PLoS ONE 13(5): e0196533. – 2018. – [Electronic resource] – Access mode: https://doi.org/10.1371/journal.pone.0196533

11. Mobasher B., Burke R.D., Sandvig J.J. Model-based collaborative filtering as a defense against profile injection attacks // In Proceedings of the 21st national conference on Artificial intelligence, Vol. 2 (AAAI'06). AAAI Press. – 2006. – P. 1388-1393. – [Electronic resource] – Access mode: https://dl.acm.org/doi/10.5555/1597348.1597409

12. Mohammadi V., Rahmani A.M., Darwesh A.M., Sahafi A. Trust-based recommendation systems in Internet of Things: a systematic literature review // Human-centric Computing and Information Sciences. – 2019. – 61 p. – [Electronic resource] – Access mode: https://doi.org/10.1186/s13673-019-0183-8

13. Ozsoy M.G., Polat F. Trust based recommendation systems // Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. – 2013. – P. 1267-1274.

14. Traag V.A. Algorithms and Dynamical Models for Communities and Reputation in Social Networks // Springer International Publishing. – 2014. – P. 229.

– [Electronic resource] – Access mode: https://doi.org/10.1007/978-3-319-06391-1

15. Мелешко Є.В., Чабан О.О., Міхав В.В. Способи побудови рекомендаційних систем для соціальних мереж з врахуванням репутації користувачів // Збірник тез Всеукраїнської науково-практичної конференції «Перспективні напрямки інформаційних і комп'ютерних систем та мереж, комп'ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті», м. Кропивницький, 13-14 листопада 2019 р. – Кропивницький: ЦНТУ. – 2019. – С. 86-87.

16. Jøsang A., Ismail R., Boyd C. A survey of trust and reputation systems for online service provision // Decision Support Systems, Volume 43, Issue 2. – 2007. – P. 618-644. DOI: https://doi.org/10.1016/j.dss.2005.05.019

17. Шингалов Д.В., Мелешко Є.В., Босько В.В. Дослідження моделей репутації користувачів соціальної мережі // Збірник тез II Міжнародної науково-практичної конференції "Інформаційна безпека та інформаційні технології", м. Кропивницький, 2-3 квітня 2020 року. – Кропивницький: ЦНТУ. – 2020. – С. 29.

18. Мелешко Є. В. Методологія забезпечення стійкості рекомендаційних систем до дестабілізуючих факторів у комп'ютерних мережах: дис. ... докт. техн. наук з комп'ютерних систем та компонентів: 05.13.05 – Комп'ютерні системи та компоненти / Є. В. Мелешко; Черкаський держ. технологічний ун-т. – Черкаси, 2020. – 323 с.

19. An accurate algorithm to calculate the Hurst exponent of self-similar processes / M. Fernández-Martínez, M. A. Sánchez-Granero, J. E. Trinidad Segovia, I. M. Román-Sánchez. Physics Letters A. 2014. Vol. 378, no. 32–33, P. 2355-2362. DOI: 10.1016/j.physleta.2014.06.018.

20. Murphy J. J. Technical Analysis of the Futures Markets: A Comprehensive Guide to Trading Methods and Applications. – Prentice Hall Press. – 1986. – 556 p.

21. Peters E. E. Fractal Market Analysis: Applying Chaos Theory to Investment and Economics. – Wiley. – 1994. – 336 p.

22. Frost A. J., Prechter R. R. Elliott Wave Principle: Key To Market Behavior. – New Classics Library. – 2005. – 244 p.

23. Anis A.A., Lloyd E.H. The expected value of the adjusted rescaled Hurst range of independent normal summands // Biometrica 63. – 1976. – P. 283-298.

24. Мелешко Є.В. Аналіз структури соціальної мережі з точки зору інформаційної безпеки // Збірник тез XVIII міжнародного науково-практичного семінару "Комбінаторні конфігурації та їх застосування", м. Кіровоград, 15-16 квітня 2016. – Кіровоград: Кіровоградський національний технічний університет. – 2016. – С. 93-97.

25. Мелешко Є.В. Дослідження методів динамічного аналізу віртуальних соціальних мереж з точки зору інформаційної безпеки // Матеріали Всеукраїнської науково-практичної конференції "Кібербезпека в Україні: правові та організаційні питання", м. Одеса, 21 жовтня 2016 р. – Одеса: ОДУВС. – 2016. – С. 154-155.

26. Mislove A., Marcon M., Gummadi K. P., Druschel P., Bhattacharjee B. Measurement and analysis of online social networks // In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07), ACM, New York, NY, USA. – 2007. – P. 29-42. DOI: https://doi.org/10.1145/1298306.1298311

27. Fortunato S. Community detection in graphs // Physics Reports, Vol. 486, Issues 3-5. – 2010. – P. 75-174. DOI: https://doi.org/10.1016/j.physrep.2009.11.002

28. Мелешко Є.В. Методи кластеризації графів для побудови рекомендаційних систем соціальних медіа // Збірник тез VII Міжнародної науково-практичної конференції «Обробка сигналів і негаусівських процесів», присвячена пам’яті професора Кунченка Ю.П., м. Черкаси, 23-24 травня 2019 р., – Черкаси: ЧДТУ. – 2019. – С. 100-102.

29. Мелешко Є.В. Розробка програмного забезпечення для виділення співтовариств у соціальній мережі // Збірник тез III Всеукраїнської науково-практичної конференції «Перспективні напрямки сучасної електроніки, інформатики і комп’ютерних систем», м. Дніпро, 21-23 листопада 2018 р. –

Дніпро: ДНУ. – 2018. – С. 42-43.

30. Foreman J. W. Data Smart: Using Data Science to Transform Information into Insight. – Wiley. – 2013. – 432 с.

31. Tantipathananandh C., Berger-Wolf T. Y. Finding Communities in Dynamic Social Networks // in IEEE 11th International Conference on Data Mining, Vancouver, BC, Canada. – 2011. P. 1236-1241. – DOI: https://doi.org/10.1109/ICDM.2011.67

32. Aggarwal C. C., Zhai C. X. Mining Text Data. – Springer New York. – 2012. – P. 524. – DOI: https://doi.org/10.1007/978-1-4614-3223-4

33. Neo4j Documentation // Official website of the graph database Neo4j. – [Electronic resource] – Access mode: https://neo4j.com/docs/

34. Meleshko Ye., Drieiev O., Drieieva H. Method of identification bot profiles based on neural networks in recommendation systems // Advanced Information Systems. – 2020. – Vol. 4, No. 2 – P. 24-28.

35. Meleshko Ye. Computer model of virtual social network with recommendation system // Innovative technologies and scientific solutions for industries. – 2019. – №2(8). – P. 80-85.

36. Harper F.M., Konstan J.A. The MovieLens Datasets: History and Context // ACM Transactions on Interactive Intelligent Systems (TiiS). – 2015. – 19 p. – [Electronic resource] – Access mode: https://doi.org/10.1145/2827872

37. Gunes I., Kaleli C., Bilge A., Polat H. Shilling attacks against recommender systems: a comprehensive survey // Artificial Intelligence Review, Vol. 42. – 2014. – P. 767-799. – [Electronic resource] – Access mode: https://doi.org/10.1007/s10462-012-9364-9

38. Kaur P., Goel S. Shilling attack models in recommender system // International Conference on Inventive Computation Technologies (ICICT), Coimbatore. – 2016. – P. 1-5. – [Electronic resource] – Access mode: https://ieeexplore.ieee.org/document/7824865/

РОЗДІЛ 4.
МОДЕЛЮВАННЯ РОБОТИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ У СКЛАДНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

4.1. Складні мережі та їх властивості

Складні мережі (complex networks) – це стохастичні мережі з нетривіальною топологією, зокрема, вони відрізняються від класичних стохастичних мереж наявністю невеликої кількості вузлів з великим числом зв'язків (такі вершини називаються хабами) [1-4]. Більшість реальних мереж – складні. Складні мережі прийнято ділити на: технічні мережі (наприклад, комп'ютерні мережі, транспортні мережі), біологічні мережі (наприклад, мережі метаболізму, екологічні мережі), соціальні мережі (наприклад, мережі друзів, мережі цитування, мережі телефонного зв'язку) тощо. Найкраще досліджені складні мережі як модель соціальних мереж.

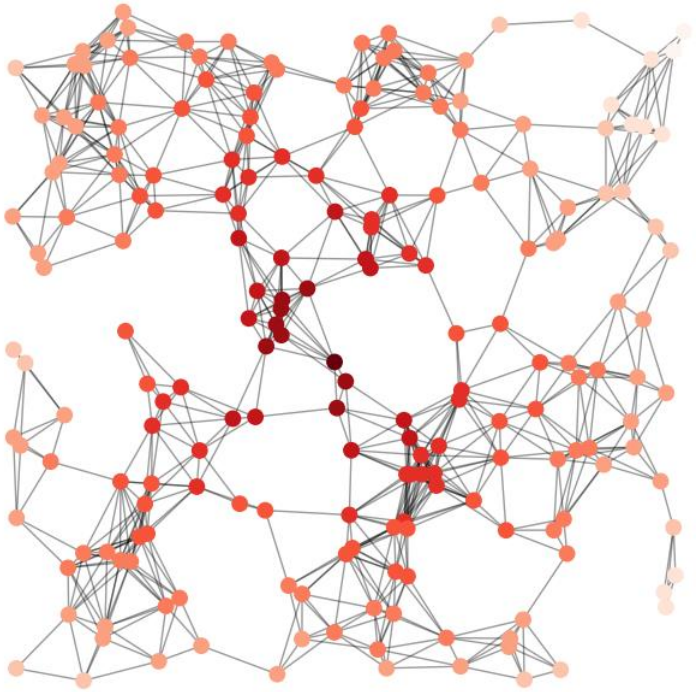


Рис. 4.1 – Приклад структури складної мережі [5]

У складних мереж, що відображають соціальні зв'язки, є наступні основні

властивості [1-11]:

1. *Безмасштабність.* Розподіл степенів вузлів (vertex degree, кількості зв'язків у вузлів) за степеневим розподілом.

2. *"Тісний світ"* (small-world network). Невеликий діаметр мережі.

3. Високий *коефіцієнт кластеризації* та високий *коефіцієнт транзитивності.* Якщо в соціальній мережі є учасники *A*, *B* та *C*, і є соціальні зв'язки між *A* та *B*, а також між *A* та *C*, то досить висока ймовірність, що у *B* та *C* також є соціальні зв'язки.

4. *Гігантська зв'язна компонента*. Тобто, більше 80% вузлів пов'язані між собою.

5. Присутні ієрархічні зв'язки.

6. Присутні складні кластерні утворення (*кліки*, *клани* тощо).

7. *Асортативність.* В широкому розумінні асортативність – це виникнення зв'язків між вершинами, які чимось схожі між собою. У вузькому розумінні асортативність – це виникнення зв'язків між вершинами з великою кількістю зв'язків.

Розглянемо основні відомі моделі генерації стохастичних та складних мереж.

Відомою моделлю генерації складних мереж є **модель Барабаши-Альберт (Barabasi-Albert model)** [1, 2, 6, 12]. Автори даної моделі показали, що для виникнення безмасштабних мереж необхідна наявність двох умов:

1. *Ріст.* Починаючи з невеликого числа n_0 вузлів, на кожній новій часовій ітерації додається один новий вузол з n зв'язками (де $n \leq n_0$), які з'єднують новий вузол з n різними уже існуючими вузлами.

2. *Бажане приєднання* (Preferencial attachment). Ймовірність P, з якою новий вузол утворить зв'язок з деяким уже існуючим вузлом i, тим вища, чим більше зв'язків у i-го вузла, та визначається за формулою:

$$P_i = \frac{k_i}{\sum_j k_j} \ , \qquad (4.1)$$

де k_i – степінь *i*-го вузла, а в знаменнику підраховується сума всіх степенів існуючих у мережі вузлів.

Цим принципом можна пояснити причини виникнення степеневого закону у соціальних мережах, асортативності та малого діаметру мережі.

Перевагами даної моделі є те, що мережа, яку вона генерує володіє властивостями розрідженості, "тісного світу", безмасштабності. Недоліками моделі є те, що результуючий граф сильно залежить від початкового параметру n_0, а також є складність з бажаним приєднанням у випадковому виборі вершин.

Модель Ердеша-Ран'ї (Erdős-Renyi model) [13]. Нехай є множина вершин $V_n = \{v_1, v_2, ..., v_n\}$, а в графі не буде петель, кратних ребер і орієнтації, тому потенційних ребер буде C_n^2. Вершини з'єднуються попарно з ймовірністю $p \in [0;1]$, незалежно від інших вершин. У даній моделі відсутнє бажане приєднання. Дана модель дозволить створити стохастичний граф, але він не буде мати важливих властивостей складних мереж, а саме степеневого закону розподілу степенів вершин та високого коефіцієнту кластеризації.

Модель Болобаша-Ріордана (Bollobas-Riordan model) [14, 15]. Спочатку будується множина випадкових графів $\{G_1^n\}$, в якій у графу з номером *n* число вершин та ребер рівне *n*. Потім ця множина перетворюється в множину $\{G_k^n\}$, в якій у графу з номером *n* число вершин рівне *n*, а число ребер рівне *kn*, $k \in N$. Дана модель генерує складні мережі та добре збігається з емпіричними даними.

Дані моделі дозволяють моделювати структуру соціального графу. Для дослідження соціальних процесів необхідно моделювати динамічну складну мережу. Моделлю динамічної мережі може бути **динамічний граф** [3].

Динамічний граф D, представляє собою послідовність класичних графів G_k, перехід між якими описується різними графовими операціями $\varphi(G_k) = G_{k+1}$. Графові операції можна поділити на базові та складні [3]. До базових операцій відносяться операції:

– додавання/видалення ребра;

– додавання/видалення вершини.

Будь-яку складну графову операцію можна описати послідовністю базових графових операцій.

Операція, що здійснює перехід від графу G_k до графу G_{k+1} може бути як базовою так і складною. Для побудови динамічного графу можна використати множину графових операцій $\Phi = \{\varphi^t\}$. Послідовність графів $G_1, G_2, G_3, \ldots G_m$, називається траєкторією динамічного графу.

Для прогнозування та моделювання змін у динамічному графі можуть використовуватися ієрархічні, ймовірнісні та реляційні моделі, моделі засновані на властивостях соціальних мереж та моделі засновані на властивостях учасників мережі [10, 11, 16-18].

Проведене дослідження показало, що для моделювання соціального графу рекомендаційної системи найкраще підійдуть моделі, що будуть розроблятися на базі відомої моделі Барабаши-Альберт, оскільки вона дозволяє відтворити найбільшу кількість властивостей соціальних мереж, зокрема, степеневий закон розподілу степенів вершин, високі коефіцієнти кластеризації та транзитивності, асортативність, малий діаметр мережі, що є важливим для моделювання структури зв'язків у рекомендаційній системі. А для симуляції інформаційних процесів слід застосовувати динамічний граф.

4.2. Моделювання рекомендаційної системи у централізованій складній комп'ютерній мережі

Розглянемо програмну імітаційну модель поведінки користувачів віртуальної соціальної мережі з рекомендаційною системою, запропоновану у [3] для тестування рекомендаційних систем, зокрема, з метою визначення та порівняння показників точності та стійкості різних методів їх побудови.

Для розробки методів моделювання структури зв'язків у соціальній мережі

з рекомендаційною системою було взято за основу, але модифікувати з врахуванням специфіки задачі, принципи, на яких базується модель Барабаши-Альберт (а саме, «ріст» та «бажане приєднання»), так як вона проста в реалізації та дозволяє створити стохастичний граф з властивостями соціальних мереж.

Для моделювання соціальної мережі з рекомендаційною системою необхідно додати у модель мережі, крім користувачів, об’єкти та рекомендації. Граф соціальної мережі повинен бути динамічним для моделювання поведінки користувачів, появи нового контенту та процесу створення і пропонування рекомендацій, а також для моделювання та відслідковування змін у мережі після надання рекомендацій користувачам.

Модель соціальної мережі з рекомендаційною системою [3] було створено з використанням графової бази даних Neo4j [19].

Соціальна мережа була представлена у вигляді стохастичного графу, у якого у якості вершин були:

– користувачі соціальної мережі;

– пости (інформаційні блоки) користувачів у соціальній мережі.

У якості ребер були відношення: "друзі"; "підписники"; "опублікування посту"; "пост переглянуто"; "лайк посту"; "подоба між користувачами"; "подоба між постами" та "пост рекомендовано".

Зовнішній вигляд зрізу графу розробленої моделі соціальної мережі з рекомендаційною системою представлено на рис. 4.2.

Генерація структури зв’язків соціальної мережі здійснювалася на основі модифікованої моделі Барабаши-Альберт. Для генерації мережі її граф був поділений на наступні підграфи: Users-Friends, Users-Followers, Users-Similarity, Posts-Published, Posts-Viewed, Posts-Liked (або Posts-Rated), Posts-Similarity та Posts-Recommended.

Підграфи Users-Friends, Users-Followers, Posts-Published, Posts-Viewed та Posts-Liked (або Posts-Rated) створюються генератором графу соціальної

мережі.

Підграфи Users-Similarity, Posts-Similarity та Posts-Recommended створюються рекомендаційною системою у процесі формування рекомендацій.

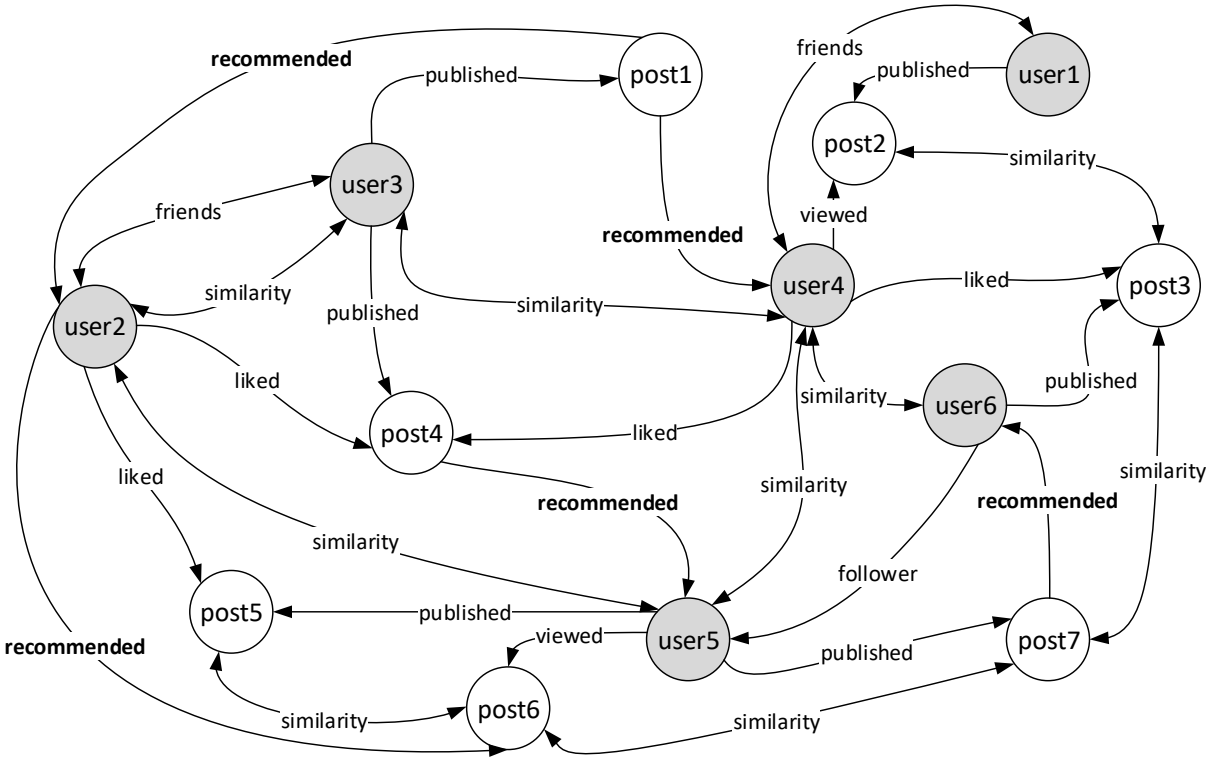


Рис. 4.2. Приклад зрізу графу запропонованої моделі соціальної мережі з рекомендаційною системою

Етапи запропонованого у [3] методу генерації структури зв'язків соціальної мережі з рекомендаційною системою:

1 етап. Генерується неорієнтований підграф Users-Friends на основі моделі Барабаши-Альберт.

2 етап. Генерується орієнтований підграф Users-Followers на основі моделі Барабаши-Альберт.

3 етап. Підграфи Users-Friends та Users-Followers об'єднуються у загальний граф.

4 етап. Генерується орієнтований підграф Posts-Published на основі модифікованої моделі Барабаши-Альберт. На першій ітерації випадковим чином обираються n_0 користувачів, які "створюють" m_0 постів. Потім на кожній новій ітерації додається новий пост, ймовірність його опублікування деяким користувачем залежить від кількості друзів та підписників у цього користувача та кількості уже опублікованих постів, і визначається за формулою:

$$P_i = \frac{k_{1i} + k_{2i} + k_{3i}}{\sum_j (k_{1j} + k_{2j} + k_{3j})}, \quad (4.2)$$

де k_{1i} – кількість друзів у i-го вузла, k_{2i} – кількість підписників у i-го вузла, k_{3i} – кількість постів у i-го вузла, а в знаменнику підраховується сума всіх цих значень для усіх існуючих у мережі вузлів.

Для кожного посту генерується набір ключових слів (або властивостей), які вибираються з заданого набору, для подальшої можливості моделювати роботу рекомендаційної системи.

5 етап. Підграф Posts-Published приєднується до загального графу.

6 етап. Генерується орієнтований підграф Posts-Viewed на основі модифікованої моделі Барабаши-Альберт. На першій ітерації випадковим чином обираються n_0 користувачів, які "переглядають" m_0 випадковим чином обраних постів. Потім на кожній новій ітерації додається новий перегляд випадкового посту, ймовірність того, що деякий пост буде переглянутий, залежить від кількості друзів та підписників у автора посту та кількості уже опублікованих ним постів, а також від кількості попередніх переглядів даного поста, та визначається за формулою:

$$P_i = \frac{q_{1i} + q_{2i} + q_{3i}}{\sum_j (q_{1j} + q_{2j} + q_{3j})}, \quad (4.3)$$

де q_{1i} – кількість друзів у автора i-го поста, q_{2i} – кількість підписників у автора i-го поста, q_{3i} – кількість переглядів у i-го поста, а в знаменнику підраховується сума всіх цих значень для усіх існуючих у мережі вузлів.

7 етап. Підграф Posts-Viewed приєднується до загального графу.

8 етап. Генерується підграф Posts-Liked (або Posts-Rated) на основі модифікованої моделі Барабаши-Альберт. На першій ітерації випадковим чином обираються n_0 користувачів, які "ставлять" m_0 лайків (або оцінок) випадковим постам. Потім на кожній новій ітерації додається новий лайк випадковому посту, ймовірність того, що деякий пост отримає лайк, залежить від кількості друзів та підписників у автора посту та кількості уже опублікованих ним постів, а також від кількості переглядів даного поста та кількості попередніх лайків даного поста, і визначається за формулою:

$$P_i = \frac{q_{1i} + q_{2i} + q_{3i} + q_{4i}}{\sum_j (q_{1j} + q_{2j} + q_{3j} + q_{4j})}, \quad (4.4)$$

де q_{1i} – кількість друзів у автора i-го поста, q_{2i} – кількість підписників у автора i-го поста, q_{3i} – кількість переглядів у i-го поста, q_{4i} – кількість лайків (або оцінок) у i-го поста, а в знаменнику підраховується сума всіх цих значень для усіх існуючих у мережі вузлів.

9 етап. Підграф Posts-Liked (або Posts-Rated) приєднується до загального графу.

10 етап. Підграфи Users-Similarity, Posts-Similarity та Posts-Recommended генеруютья алгоритмами обраної рекомендаційної системи та приєднуються до загального графу.

Приклад частини соціальної мережі, одержаної в результаті моделювання зображено на рис. 4.3.

З метою тестування розробленої системи згенеровані графи соціальної мережі були експортовані у .csv файли з бази даних Neo4j, а їх параметри досліджені у програмному забезпеченні Gephi [20].

Gephi – це інтерактивна платформа з відкритим кодом для аналізу та візуалізації даних представлених у вигляді графів, що дозволяє досліджувати всі види мереж, складних систем, статичних та динамічних графів.

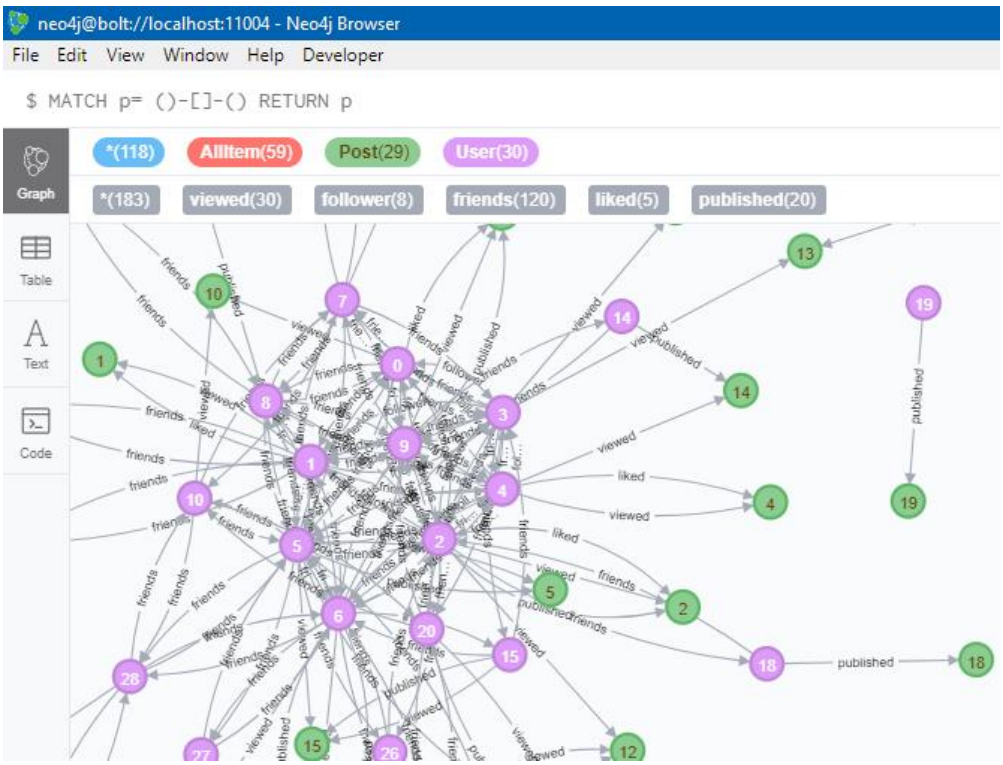


Рис. 4.3. Приклад частини соціальної мережі одержаної в результаті моделювання (скриншот з менеджера СУБД Neo4j Desktop)

Середні значення параметрів згенерованих у запропонованій програмній імітаційній моделі соціальних графів рекомендаційної системи, які були обчислені за допомогою Gephi:

– Середній степінь вузлів: 5.7.

– Діаметр мережі: 4.0.

– Щільність графу: 0.22.

– Середній коефіцієнт кластеризації: 0.61.

– Середня довжина шляху: 1.98.

Отже, при генерації структури зв’язків соціальних мереж розробленим методом було отримано графи, що були сильно розріджені (мали низьку щільність), діаметр мережі був у середньому рівним 4 (що відповідає сучасним віртуальним соціальним мережам, наприклад, у мережі Facebook цей показник 4.74), коефіцієнт кластеризації був досить високим, візуально на графі

спостерігалися різні кластерні утворення (кліки та клани), середня довжина шляху була невисока. Все це відповідає параметрам реальних соціальних мереж. Цей метод також можна використати як основу для моделювання структури зв'язків користувачів та об'єктів і контент-орієнтованого веб-сайту.

У [3] було здійснено моделювання поведінки користувачів на основі генерації динамічного графу Users-Ratings-Items. Модель розроблена таким чином, щоб одержувати набір даних схожий на MovieLens datasets [21].

Етапи запропонованого методу моделювання структури зв'язків між елементами та поведінки користувачів рекомендаційної системи:

1 етап. Ініціалізація параметрів системи, зокрема, вибір кількості користувачів та об'єктів, кількості можливих властивостей у них, проценту активних користувачів, проценту популярних об'єктів, кількості часових ітерацій, після яких модель завершить свою роботу тощо. При необхідності моделювати інформаційну атаку обирається тип атаки, кількість ботів та кількість цілей атаки. Відбувається створення набору можливих властивостей для елементів системи, реалізованих у моделі прихованими факторами, та генерація шаблонів кластерів елементів на основі цих властивостей.

2 етап. Створення «Зерна» соціального графу рекомендаційної системи – до графу додається початкова кількість користувачів та об'єктів, деякій кількості об'єктів виставляються оцінки. Початкова кількість користувачів, початкова кількість об'єктів та щільність графу – налаштовувані параметри. Кожному новому користувачу та об'єкту системи привласнюється значення зміщення, певний кластер та відповідний йому вектор значень прихованих факторів, що визначає ступінь його приналежності до кожної з можливих властивостей для елементів системи. Оцінки у системі виставляються на основі ступеня співпадіння прихованих факторів користувача та об'єкта, а також показників їх зміщень. Ймовірність перегляду об'єкту всередині «Зерна» залежить від заданої щільності графу. Ймовірність виставлення оцінки переглянутому об'єкту для аутентичних користувачів залежить від прихованих

факторів користувача (що визначають його вподобання), прихованих факторів об’єкту (що визначають його властивості), зміщення користувача (яке вказує на характерне для користувача систематичне заниження або завищення оцінок), зміщення об’єкту (яке вказує на якість об’єкту, що викликає завжди одержання оцінок вище/нижче, ніж у схожих за властивостями об’єктів), а також випадкового зміщення (яке виникає з ймовірністю 0-3% і є налаштовуваним параметром). У «Зерні» профілі ботів відсутні, вони починають приєднуватися до мережі на 3 етапі.

3 етап. На кожній ітерації часу моделі до графу приєднується певна кількість користувачів та фільмів. Ця кількість визначається випадковим чином та лежить у межах від 0 до N, де N менше загальної кількості елементів системи відповідного типу. Також на кожній ітерації часу моделі відбувається вибір деякої кількості пар користувачів та об’єктів для виставлення оцінок. Ймовірність перегляду об’єкту може визначатися на основі принципу «бажаного приєднання» з моделі Барабаши-Альберт. Ймовірність виставлення оцінки переглянутому об’єкту для аутентичних користувачів залежить від прихованих факторів користувача (що визначають його вподобання), прихованих факторів об’єкту (що визначають його властивості), зміщення користувача (яке вказує на характерне для користувача систематичне заниження або завищення оцінок), зміщення об’єкту (яке вказує на якість об’єкту, що викликає завжди одержання оцінок вище/нижче, ніж у схожих за властивостями об’єктів), а також випадкового зміщення (яке виникає з ймовірністю 0-3% і є налаштовуваним параметром). Ймовірність перегляду об’єкту та виставлення йому оцінки для ботів визначається використаною для їх створення моделлю атаки.

4 етап. Зупинка роботи імітаційної моделі, збереження згенерованого набору даних у файл для подальшого використання у методах формування та тестування списків рекомендацій.

Структурна схема розробленої програмної моделі зображена на рис. 4.4.

Рис. 4.4. Структурна схема програмної імітаційної моделі користувачів та об’єктів рекомендаційної системи

З рисунку видно, що програмна імітаційна модель складається з наступних елементів: «Модель користувача системи», «Модель об’єкту системи» та «Модель інформаційних процесів у рекомендаційній системі».

Модель користувача системи має наступні параметри:

– Id – ідентифікаційний номер користувача, містить цифру;

– Активний – приймає значення True – якщо користувач активний (ставить більше оцінок, ніж інші) та False – якщо рівень активності звичайний;

– Бот – приймає значення True – якщо користувач бот (профіль приймає участь в атаці на рекомендаційну систему) та False – в протилежному випадку;

– Тип – приймає значення «Звичайний», якщо користувач не бот, інакше містить назву застосованої на рекомендаційну систему атаки;

– Характер – містить тип характеру користувача стосовно стилю виставлення оцінок: «Звичайний», «Частіше ставить хороші оцінки», «Ставить тільки хороші оцінки». «Звичайний» ставить і позитивні, і негативні оцінки. Інші типи користувачів можуть не виставляти оцінку, якщо оцінили об’єкт негативно;

– Час реєстрації – містить час реєстрації користувача у системі;

– Час останньої активності – містить час останньої активності користувача;

– Час зупинки активності – час, коли користувач перестав користуватися системою;

– Приховані фактори – список випадкових змінних, які приймають значення від -1 до 1, довжиною K, що моделює вплив тих чи інших характеристик фільму на судження про нього користувача;

– Зсув користувача – випадкова величина в діапазоні від -1.0 до 1.0, що моделює схильність користувача занижувати чи завищувати оцінки фільмам;

– Час останньої зміни вподобань – використовується при необхідності моделювання змін вподобань користувачів у часі, містить час останнього перезапису списку прихованих факторів користувача.

У розробленому програмному забезпеченні об’єктами системи були

фільми. Модель об’єкту системи має наступні параметри:

– Id – ідентифікаційний номер об’єкту, містить цифру;

– Популярний – приймає значення True – якщо об’єкт популярний (отримує більше оцінок, ніж інші) та False – якщо рівень популярності звичайний;

– Тип для ботів – приймає значення «Цільовий» – якщо боти повинні змінити рейтинг об’єкту в певну сторону та False – в протилежному випадку;

– Тип атаки – приймає значення «Немає», якщо об’єкт не цільовий, або вказує тип атаки «На зменшення рейтингу», «На збільшення рейтингу»;

– Час додавання – містить час додавання об’єкту до бази даних рекомендаційної системи;

– Приховані фактори – список випадкових змінних, які приймають значення від -1 до 1, довжиною K, що моделює вираженість тих чи інших характеристик у даному об’єкту, які впливають на рівень інтересу до нього користувача;

– Зсув об’єкту – випадкова величина в діапазоні від -1.0 до 1.0, що моделює загальну якість об’єкту, що впливає на оцінки користувачів і призводить до отримання частіше низьких оцінок через низьку якість або частіше високих оцінок через високу якість об’єкту.

Параметри користувачів та об’єктів системи встановлюються під час створення та ініціалізації відповідних екземплярів об’єктів, а деякі з них можуть змінюватися під час роботи програми.

Модель інформаційних процесів у системі має наступні параметри:

– Матриця рейтингів – містить матрицю суміжності графу, у якого вершинами являються користувачі та об’єкти, а ребрами оцінки поставлені користувачами об’єктам.

– Матриця часу – містить матрицю суміжності графу, у якого вершинами являються користувачі та об’єкти, а ребрами час виставлення оцінок поставлених користувачами об’єктам.

– Набір можливих оцінок – містить набір оцінок, які користувачі можуть виставляти об’єктам, в розроблюваній моделі набір оцінок представлений наступним списком [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0], що означає можливість поставити оцінку у вигляді кількості зірочок, зірочок максимум 5, можна вибирати половинку зірочки.

– Список користувачів – містить список користувачів системи, екземплярів класу Користувач.

– Список об’єктів – містить список об’єктів системи, екземплярів класу Об’єкт.

– Початковий час роботи моделі – містить час у форматі Unix time stamp.

– Поточний час роботи моделі – містить час у форматі Unix time stamp, що вказує на час останньої дії у системі.

– Список ботів – містить список ботів у розроблюваній моделі.

– Список цілей – містить список цілей атаки ботів у розроблюваній моделі.

– Кількість кластерів – містить задану кількість кластерів, на яку можна буде розділити елементи системи на основі даних про їх приховані фактори.

– Шаблони кластерів – містить список шаблонів для моделювання прихованих факторів користувачів/об’єктів. Всього у моделі було створено 19 випадкових шаблонів для генерації елементів, що можуть належати 19 різним кластерам. Також в моделі можна обрати опцію моделювання певного проценту елементів, що не належать до згенерованих кластерів.

Було реалізовано наступні функції для моделювання поведінки користувачів і об’єктів та інформаційних процесів рекомендаційної системи:

– Запуск моделі – запускає процес моделювання користувачів та об’єктів рекомендаційної системи.

– Збереження даних моделі у файл – зберігає усі дані та параметри розробленої програмної імітаційної моделі користувачів, об’єктів та процесів рекомендаційної системи у файл.

– Генерація елементів системи – створення заданої кількості користувачів

та об’єктів системи та привласнення їм параметрів.

– Генерація шаблонів кластерів елементів – дозволяє згенерувати шаблони прихованих факторів елементів для створення в подальшому набору елементів, що відносяться до певних кластерів.

– Генерація прихованих факторів елементів на основі шаблонів кластерів – на вході одержує шаблон кластера, на виході надає список прихованих факторів, що випадковим чином на деякі величини відрізняються від даних у шаблоні, таким чином, щоб не виходити за межі кластеру, але мати свої унікальні значення факторів.

– Генерація прихованих факторів елементів за межами кластерів – генерація списку прихованих факторів елемента випадковим чином без використання шаблонів.

– Генерація «Зерна» соціального графу – створює початковий соціальний граф рекомендаційної системи на основі заданої початкової кількості користувачів, об’єктів та щільності графу.

– Ітеративна побудова моделі – дозволяє моделювати зміну часу.

– Одна ітерація моделі – всередині даної функції викликаються всі функції, що реалізують поведінку користувачів та роботу рекомендаційної системи в поточний момент часу.

– Генерація поточного часу роботи моделі – генерація часового проміжку між двома подіями у системі – випадкової величини, що лежить в заданому діапазоні.

– Додавання користувача до системи – моделювання процесу реєстрації користувача у рекомендаційній системі, користувач отримує час реєстрації та можливість переглядати об’єкти та ставити їм оцінки.

– Додавання об’єкту до системи – моделювання процесу додавання об’єкту до бази даних рекомендаційної системи, об’єкт одержує час додавання до бази даних та можливість одержувати перегляди та оцінки.

– Визначення ймовірності перегляду об'єкту – на основі принципу

«Бажаного приєднання» визначається ймовірність перегляду певного об’єкту певним користувачем.

– Визначення ймовірності появи оцінки – на основі принципу «Бажаного приєднання» визначається ймовірність виставлення оцінки певному об’єкту певним користувачем.

– Генерація оцінки на основі прихованих факторів відповідного об’єкту та користувача. Оцінка для пари користувач-об’єкт визначається за наступними формулами:

$$d_{u,m} = \frac{\sum_{i=0}^{n} \left| f_{u,i} - f_{m,i} \right|}{n}, \quad (4.5)$$

$$r_{u,m} = \Psi\left(5d_{u,m} + b_u + b_m\right), \quad (4.6)$$

де $d_{u,m}$ – дистанція між користувачем u та об’єктом m у багатомірному просторі прихованих факторів, може приймати значення від 0 до 1; n – кількість прихованих факторів у системі; $f_{u,i}$ – i-тий прихований фактор користувача u; $f_{m,i}$ – i-тий прихований фактор об’єкту m; b_u – зсув користувача в оцінюванні об’єктів (рівень вимогливості до контенту); b_m – зсув об’єкту у одержанні оцінок (рівень якості контенту); $\Psi()$ – функція, що перетворює одержане дробове число у дискретне число з набору оцінок [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0], наприклад, якщо число лежить в діапазоні від (4.000 – k) до (4.500 – k), де k деяке невелике число (у моделі було взято k = 0.05), то воно перетворюється на оцінку 4.0.

– Рішення поставити оцінку та її коригування. Після того як користувач «переглянув» об’єкт, і на основі прихованих факторів, було визначено, яка оцінка для даного об’єкту відповідає його вподобанням, використовується дана функція, що визначає факт того, що користувач прийме рішення поставити переглянутому об’єкту оцінку. А також у даній функції можливе незначне коригування оцінки на основі випадкових чинників. За допомогою даної

функції ймовірності появи оцінок можна наблизити до частот появи оцінок у наборі даних MovieLens datasets, наведених у таблиці 4.1.

Таблиця 4.1. Частота появи різних оцінок у відкритому наборі даних MovieLens datasets

Оцінка	**Частота появи**
0.5	0.015558
1.0	0.032405
1.5	0.015509
2.0	0.067723
2.5	0.048238
3.0	0.201993
3.5	0.119742
4.0	0.268933
4.5	0.083401
5.0	0.146498

Також в запропонованій моделі [3] розроблені генератори оцінок ботів для різних видів атак.

– Генерація оцінки для випадкової атаки – створює оцінки для пар бот-об’єкт, де бот здійснює випадкову атаку на рекомендаційну мережу. Генерація оцінки для випадкової атаки на підвищення рейтингу відбувається відбувається наступним чином:

$$r_{u,m} = \begin{cases} \text{randomPattern}(), \textit{якщо об'єкт} - \text{"}\textit{звичайний}\text{"} \\ 5.0, \textit{якщо об'єкт} - \text{"}\textit{цільовий}\text{"} \end{cases}, \qquad (4.7)$$

де randomPattern() – функція, що генерує випадкові оцінки з заданими значеннями ймовірності появи.

– Генерація оцінки для середньої атаки – створює оцінки для пар бот-об’єкт, де бот здійснює середню атаку на рекомендаційну мережу. Генерація оцінки для середньої атаки на підвищення рейтингу відбувається наступним чином:

$$r_{u,m} = \begin{cases} \textit{average}\,\text{Pattern}(), \textit{якщо об'єкт} - \text{"}\textit{звичайний}\text{"} \\ 5.0, \textit{якщо об'єкт} - \text{"}\textit{цільовий}\text{"} \end{cases}, \qquad (4.8)$$

де *average*Pattern() – функція, що генерує для випадково обраного об’єкту його середньостатистичну оцінку.

– Генерація оцінки для популярної атаки – створює оцінки для пар бот-об’єкт, де бот здійснює популярну атаку на рекомендаційну мережу. Генерація оцінки для популярної атаки на підвищення рейтингу відбувається відбувається наступним чином:

$$r_{u,m} = \begin{cases} popular\,\text{Pattern}(), \textit{якщо об'єкт} - \text{"}\textit{звичайний}\text{"} \\ 5.0, \textit{якщо об'єкт} - \text{"}\textit{цільовий}\text{"} \end{cases}, \qquad (4.9)$$

де *popular*Pattern() – функція, що генерує для випадково обраного об’єкту з множини популярних об’єктів його середньостатистичну оцінку.

– Визначення ймовірності зміни вподобань – використовується при необхідності моделювання змін вподобань користувачів системи у часі, визначає ймовірність того, що у поточний момент часу вказаний користувач змінить свої вподобання, якщо відомий час, коли він останній раз змінював вподобання.

– Зміна вподобань користувача – здійснює заміну прихованих факторів вказаного користувача.

Соціальний граф рекомендаційної системи у програмній імітаційній моделі має наступний формат: вершини – користувачі та об’єкти, ребра – зв’язки типу «оцінив», «подоба», «рекомендовано» тощо. І вершини, і ребра містять набори параметрів, що відповідають наявній про них інформації. Наприклад, ребро «оцінив» містить значення оцінки і часову мітку, а вершина типу користувач містить усі поля даних, що відповідають параметрам моделі користувача у системі.

На рис. 4.5-4.7 наведено приклади частин (зрізів соціального графу) бази даних програмної імітаційної моделі рекомендаційної системи.

На рис. 4.5 наведено виконання наступного запиту до СУБД Neo4j Desktop:

```
match p=(u1:User)-[r1:Rated{goal: "testing"}]->(m1:Movie)
```

```
return p LIMIT 20
```

Цей запит виводить на екран зріз графу з вершинами представленими користувачами і фільмами та ребрами представленими оцінками, віднесеними за часом виставлення до тестового набору даних. У запиті встановлено ліміт виведення на екран у кількості 20 вершин графу.

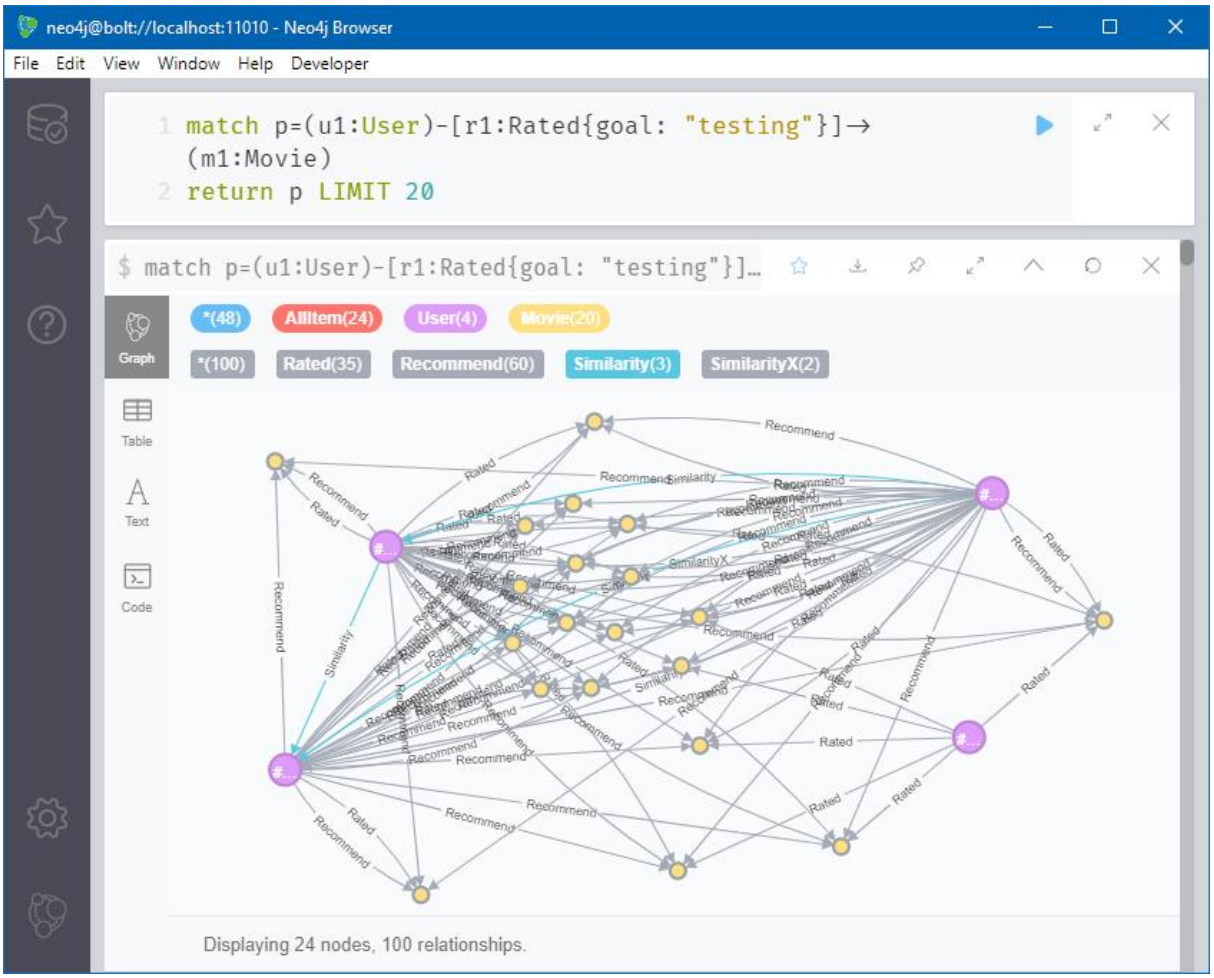


Рис. 4.5. Приклад частини бази даних запропонованої програмної імітаційної моделі рекомендаційної системи №1 (скриншот з менеджера СУБД Neo4j Desktop)

На рис. 4.5 видно частину даних з записами 4 користувачів, що оцінили деяку множину фільмів. Для відображення даних по запиту СУБД Neo4j обрала випадковим чином 24 вершини графу серед користувачів та фільмів. Також були відображені ребра :Rated та :Recommend, що містять поставлені

користувачами оцінки у тестовій вибірці та рекомендації спрогнозовані рекомендаційною системою на основі робочої вибірки даних.

На рис. 4.6 наведено виконання наступного запиту до СУБД Neo4j Desktop:

```
match p=(u1:User)-[r1:Rated{goal: "testing"}]-(m1:Movie)-[r2:Recommend]-(u1:User)
return r1.rating, r2.Recommend
```

Цей запит виводить на екран таблицю з порівнянням прогнозованих та реальних оцінок користувачів у розробленій програмній імітаційній моделі.

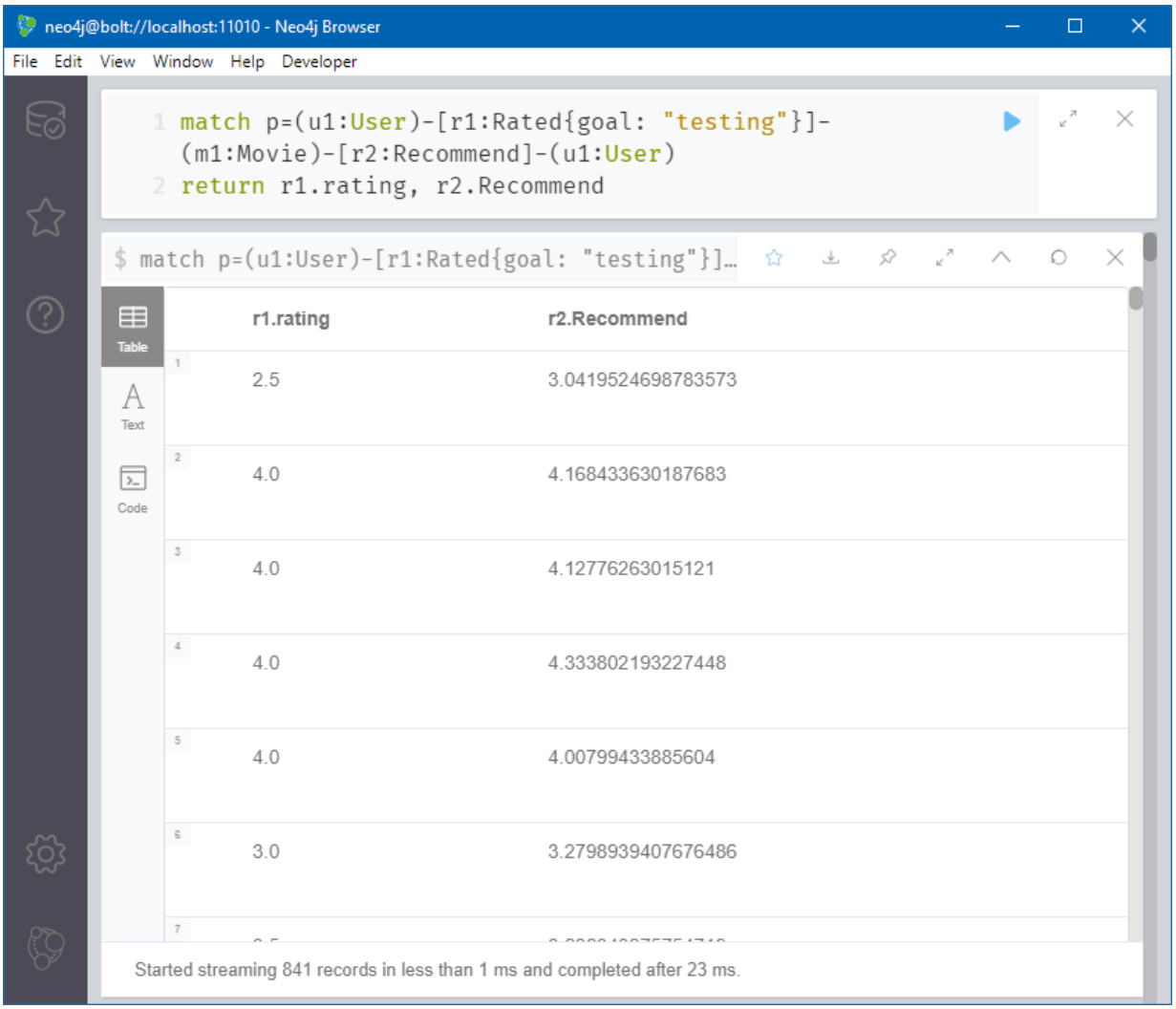


Рис. 4.6. Приклад частини бази даних запропонованої програмної імітаційної моделі рекомендаційної системи №2 (скриншот з менеджера СУБД Neo4j Desktop)

На рис. 4.7 наведено виконання наступного запиту до СУБД Neo4j Desktop:

```
match p=(u1:User{Bot:1})-[r:Similarity]-(u2:User{Bot:1})
where r.SimilarityCoefficient > 0.6
return p
```

Цей запит виводить на екран згенерованих у програмній імітаційній моделі ботів, коефіцієнти подоби яких між собою більші, ніж 0.6.

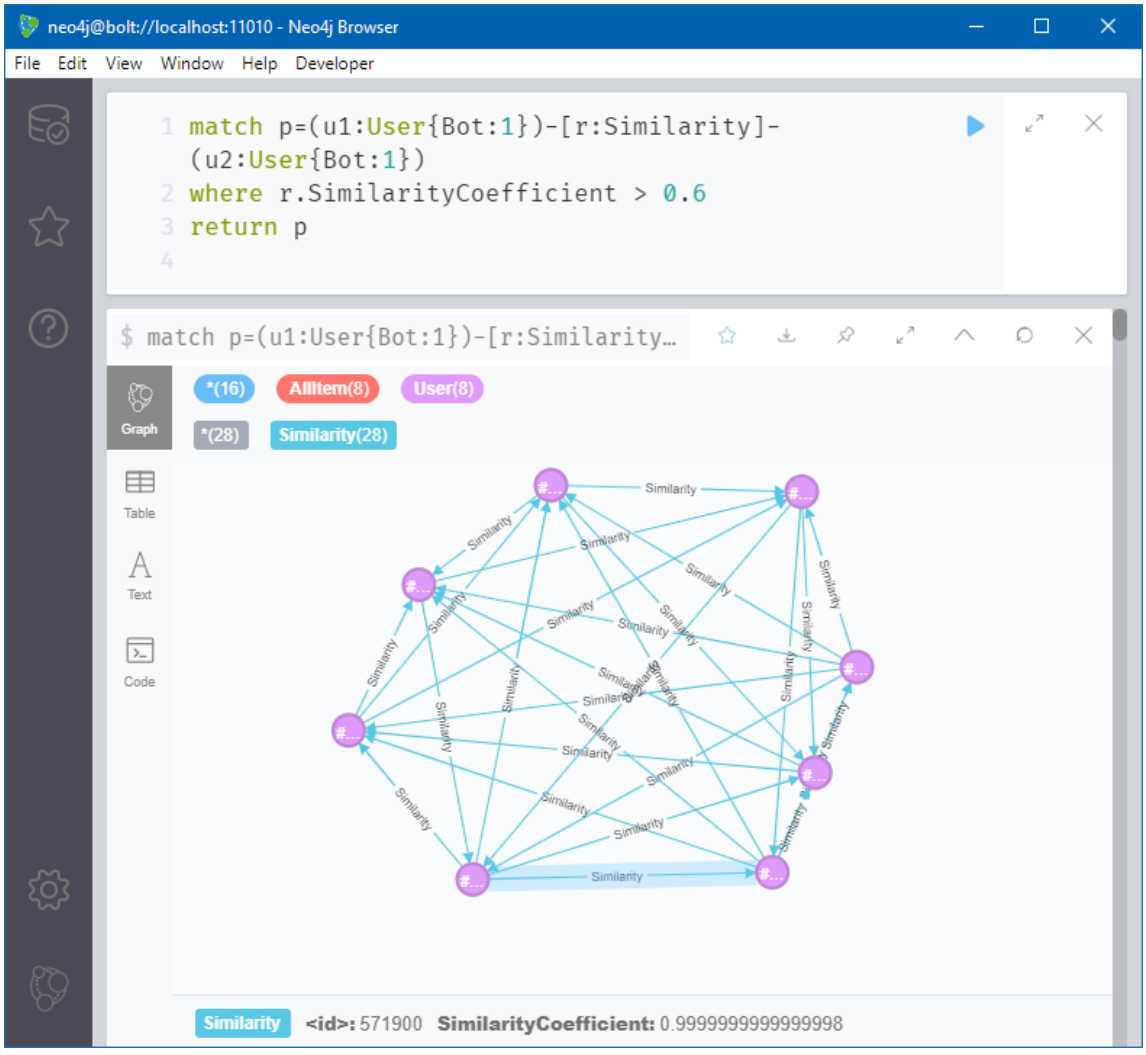


Рис. 4.7. Приклад частини бази даних запропонованої програмної імітаційної моделі рекомендаційної системи №3 (скриншот з менеджера СУБД Neo4j Desktop)

Отже, розглянутий метод програмного імітаційного моделювання користувачів та об’єктів рекомендаційної системи дозволяє генерувати набори даних для тестування алгоритмів роботи рекомендаційних систем. Розроблений метод дозволяє моделювати поведінку як звичайних користувачів, так і ботів, що дає можливість створювати набори даних для тестування стійкості рекомендаційних систем до інформаційних атак, а також ефективності методів виявлення та нейтралізації бот-мереж. Структура зв’язків між користувачами та об’єктами рекомендаційної системи моделюється за допомогою теорії складних мереж. Інформаційні атаки ботів моделюються на основі відомих моделей атак ін’єкцією профілів на рекомендаційні системи.

4.3. Моделювання рекомендаційної системи у децентралізованій складній комп’ютерній мережі

В деяких видах peer-to-peer мереж класичні методи роботи рекомендаційних систем не зможуть працювати без певної адаптації та змін внаслідок особливостей архітектури цих мереж.

P2P (peer-to-peer) – одноранговi комп’ютерні мережі, які засновані на принципі рівноправності вузлів мережі і характеризуються тим, що будь-які вузли можуть з’єднуватися між собою для надання послуг один одному, на відміну від традиційної архітектури, у якій лише окрема категорія учасників (сервери), може надавати певні сервіси іншим.

Методи організації P2P мереж можна розділити на наступні три види [22-32]:

– Централізовані одноранговi мережі, наприклад, BitTorrent [31].

– Неструктуровані децентралізовані одноранговi мережі, цей тип мереж використовує алгоритм пошуку Flooding, типовий представник – Gnutella [32].

– Структуровані децентралізовані одноранговi мережі, найчастіше будуються на основі розподілених хеш-таблиць, наприклад, на основі

алгоритму Kademlia [29] або Chord [30].

BitTorrent – пірінговий (P2P) мережевий протокол для кооперативного обміну файлами через мережу Інтернет.

Flooding – метод пошуку ресурсів в однорангових децентралізованих неструктурованих мережах, у якому пошуковий запит передається від сусіда до сусіда, поки ресурс не буде знайдено або поки не буде опитано усі вузли.

Gnutella – повністю децентралізована файлообмінна мережа, використовує для роботи однорангову комп'ютерну мережу, побудовану з користувачів мережі Інтернет.

Chord – метод маршрутизації в однорангових децентралізованих структурованих комп'ютерних мережах, реалізований на основі розподіленої хеш-таблиці. Chord визначає, як саме ключі призначаються вузлам, і як вузол може виявити значення для цього ключа, спочатку виявивши вузол, відповідальний за цей ключ.

Kademlia – метод маршрутизації в однорангових децентралізованих структурованих комп'ютерних мережах, реалізований на основі розподіленої хеш-таблиці. Протокол Kademlia визначає структуру мережі, що регулює зв'язок між вузлами, та спосіб обміну інформацією в ній.

Для централізованих однорангових мереж можливе застосування стандартних методів побудови рекомендаційних систем, оскільки у таких мережах є сервер, що зберігає посилання на файли та статистику. Складнощі виникають при намаганні створити рекомендаційні системи для децентралізованих однорангових мереж, тому що треба враховувати їх архітектуру і адаптувати відомі методи та підходи до неї. Зокрема, в децентралізованих мережах уся інформація розподілена по усім комп'ютерам мережі (це і файли, і таблиці маршрутизації, і статистика тощо), тому виникає проблема збору інформації для рекомендаційної системи та розгляд можливості зібрати не всю інформацію, а мінімально потрібну, щоб не опитувати всі комп'ютери у мережі.

Розглянемо принципи роботи однорангових децентралізованих структурованих комп'ютерних мереж. Найчастіше для індексації та маршрутизації вони використовують розподілені хеш-таблиці (Distributed hash table, DHT). Для простоти будемо називати далі такі комп'ютерні мережі – P2P DHT мережі.

Узагальнені принципи роботи P2P DHT мереж наступні [22-24, 29, 30]:

1. *Ідентифікатори комп'ютерів.* Комп'ютери мають ідентифікатори $h \in H$, що змінюються у діапазоні $[0, 2^m\text{-}1]$. Вони визначається деякою хеш-функцією або залежить від черговості підключення комп'ютерів до мережі. Наприклад, 1-й комп'ютер у мережі отримує ідентифікатор $h_0=0$. i-тий комп'ютер одержує ідентифікатор $h_i=i$ або один з вільних ідентифікаторів, що звільнилися внаслідок виходу з мережі деяких комп'ютерів, що раніше були приєднані. Оскільки в децентралізованих мережах комп'ютери можуть приєднуватися та від'єднуватися від мережі, тобто, учасники постійно змінюються, задача виявлення уже невикористовуваних індексів і надання їх новим комп'ютерам є важливою та по-різному вирішується у різних алгоритмах. Як правило, не тільки при приєднанні нового комп'ютера до мережі, а й з деякою періодичністю відбувається перевірка доступності вузлів та оновлення таблиць маршрутизації.

2. *Ідентифікатори файлів.* Файли також мають ідентифікатори $h \in H$, що змінюються у такому ж діапазоні як і у комп'ютерів $[0, 2^m\text{-}1]$. Вони визначаються деякою хеш-функцією. Використовується так зване узгоджене хешування, що на відміну від лінійного передбачає зміну в середньому тільки K/n ключів, де K – число ключів, а n – число слотів. Популярним є застосування алгоритму хешування SHA-1 в P2P DHT мережах, в такому разі m=160, що дозволяє створити досить велику множину ідентифікаторів.

3. *Зберігання файлів.* Файли чи їх частини (при розподіленому зберіганні) або шляхи до них (при розподіленому індексуванні) зберігаються на комп'ютерах таким чином, щоб ідентифікатор комп'ютера та ідентифікатор

файлу (його частини) співпадали. Або, при відсутності у мережі комп'ютера з потрібним ідентифікатором, вони повинні бути максимально близькі з наявних варіантів за деякою метрикою відстані. Надалі для простоти будемо розглядати розподілене зберігання файлів. Для забезпечення надійності зберігання інформації кожен файл дублюють на декілька комп'ютерів, для цього обирається q наявних у мережі комп'ютерів, ідентифікатори яких найближчі за обраною метрикою відстані до ідентифікатора файлу. При зміні складу учасників мережі, за необхідності, відбувається перерозподіл частини файлів між комп'ютерами.

4. *Таблиця маршрутизації.* Зберігання таблиці маршрутизації також є розподіленим. Кожен комп'ютер містить інформацію потрібну для доступу (мережеві адреси) найближчих N сусідів $X = \{x_{i-n}, \dots x_{i+n}\}$ рис. 4.8.

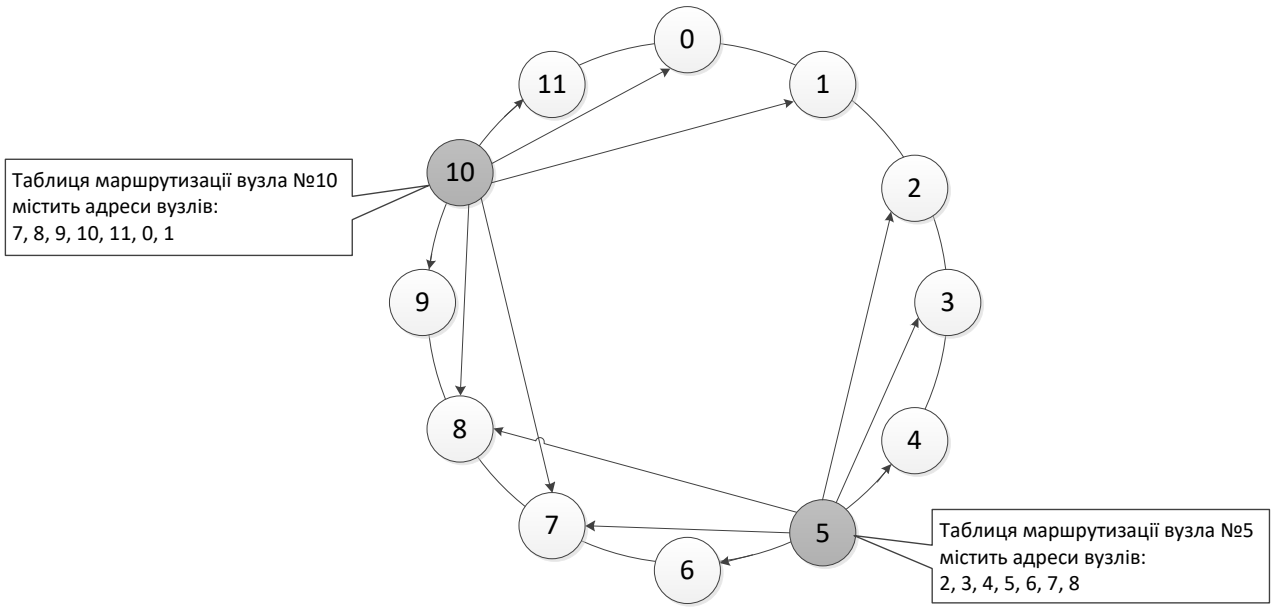


Рис. 4.8. Схематичне зображення принципу наповнення таблиць маршрутизації в однорангових децентралізованих структурованих комп'ютерних мережах: таблиці маршрутизації показані для комп'ютерів №5 та №10 при зберіганні 6 найближчих сусідів у мережі з 12 вузлів за принципами алгоритму Chord

Визначення множини X, а також розмір N цієї множини, у кожному конкретному методі здійснюється по-різному. Наприклад, в алгоритмі Chord [30] таблиця маршрутизації комп'ютера містить $2r+1$ записи, тобто інформацію про сам комп'ютер, а також про r комп'ютерів, що мають найближчі менші номери серед наявних у мережі комп'ютерів, та про r комп'ютерів, що мають найближчі більші номери серед наявних комп'ютерів. Комп'ютери об'єднуються в кільце, тобто, останній комп'ютер посилається на перший. Що викликає можливі проблеми з неправильним замиканням кільця, виникненням декількох кілець та розривом кільця при помилках в оновленні таблиць маршрутизації. В алгоритмі Kademlia [29] таблиця маршрутизації зберігається у вигляді так званих K-bucket-тів. У кожному K-bucket-ті вузла зберігається інформація про K вузлів мережі, чия відстань до нього знаходиться в межах інтервалу [2^i, 2^(i+1)), у якості метрики відстані використовується операція XOR. Якщо, наприклад, для створення ідентифікаторів застосовується хеш-функція SHA-1, то кількість K-bucket-тів на кожному вузлі буде 160. K – загальносистемне число, наприклад 20. Кожен K-bucket – це список, що містить не більше K-записів; тобто для мережі з K=20 кожен вузол матиме списки, що містять до 20 вузлів для певної відстані від себе. На практиці виходить, що кожен вузол зберігає інформацію про вузли, з якими будь-коли взаємодіяв, протягом певного часу (наприклад, час життя запиту 24 години). Таким чином розмір таблиці маршрутизації змінюється динамічно, а деякі K-bucket-ти будуть порожніми в деякий момент часу.

5. *Пошук файлів.* Файли у мережі шукаються за їх хеш-значеннями. Комп'ютер, який здійснює пошук файлу, вибирає в своїй таблиці маршрутизації j комп'ютерів, ідентифікатори яких найближчі за обраною метрикою відстані до ідентифікатора шуканого файлу. Кожен комп'ютер з множини вибраних $S = \{s_0, s_1, \dots s_j\}$ перевіряє чи міститься безпосередньо на ньому шуканий файл, і якщо ні, то пересилає пошукове повідомлення уже j найпідходящим комп'ютерам зі своєї таблиці маршрутизації, і так поки файл не

буде знайдений. Чим більше значення *s*, тим менша ймовірність відмови та вищий рівень інформаційної безпеки, але й вище навантаження на мережу. Можна було б пересилати запит тільки на один комп'ютер з найближчим до шуканого ідентифікатором, але може виявитися, що він на даний час відсутній у мережі. Або ж цим комп'ютером володіє зловмисник, що перенаправить запит не на пошук та завантаження потрібного файлу, а на завантаження вірусу, в той же час при направленні запиту на декілька комп'ютерів можна буде обрати ту відповідь, яка прийде від більшості, що значно зменшить ймовірність інформаційної атаки.

6. *Основні запити до мережі.* Методи роботи P2P DHT мереж повинні містити також наступні функції:

– Перевірка наявності сусідніх вузлів у мережі для оновлення таблиць маршрутизації;

– Пошук вузлів за ідентифікатором;

– Пошук файлів за ідентифікатором.

Зокрема, в протоколі алгоритму Kademlia для їх реалізації наявні наступні 4 типи запитів: 1) PING – необхідний для перевірки існування конкретного вузла у мережі; 2) STORE дає змогу розмістити інформацію на заданому вузлі; 3) FIND_VALUE – дозволяє знайти значення за ключем; 4) FIND_NODE – використовується для пошуку найближчих K вузлів до заданого ідентифікатора (схожий на FIND_VALUE, тільки ніколи не повертає значення, завжди вузли).

З врахуванням особливостей архітектури P2P DHT мереж у [33] було запропоновано метод роботи рекомендаційних систем для них.

Для реалізації рекомендаційної системи у P2P DHT мережі найкраще підійдуть підходи засновані на колаборативній фільтрації, адже для неї достатньо знати тільки реакцію користувача на контент, що може бути представлена оцінками, лайками або самим фактом завантажень файлів. Вміст файлів, їх назви та інформацію про користувачів знати не обов'язково. Таким чином можна працювати лише з ідентифікаторами файлів, ідентифікаторами

користувачів та статистикою завантаження файлів, це все можна отримати при роботі з децентралізованою структурованою одноранговою комп'ютерною мережею.

Для представлення даних рекомендаційної системи у пам'яті у [33] було використано дві структури даних: розгорнуті зв'язні списки та хеш-таблиці із відкритою адресацією. Розгорнуті зв'язні списки були обрані після проведення експерименту з використанням програмної імітаційної моделі, де тестувалися різні методи зберігання даних рекомендаційної системи [34]. Хеш-таблиці обрані для проміжних обчислень.

Розгорнутий зв'язний список – це зв'язний список, який складається із блоків з наперед визначеною кількістю елементів. Така структура дає можливість поєднати ефективне розширення зв'язних списків та переваги послідовного доступу до елементів блоку. У [33] було використано цю структуру даних для збереження інформації про вподобання користувачів.

Для реалізації децентралізованої рекомендаційної системи у [33] використовувалося два типи списків:

1) асоційовані з користувачем – містять список об'єктів, які він вподобав;

2) асоційовані з об'єктом – містять список користувачів, які вподобали цей об'єкт.

Таким чином, використовуються дві копії інформації про вподобання, що збільшує витрати пам'яті, але дає можливість уникати перегляду усіх вподобань для генерації рекомендації та дає можливість врахувати архітектуру P2P DHT мереж. Зокрема, кожен комп'ютер може містити список вподобань асоційований зі своїм користувачем та списки вподобань асоційовані з об'єктами, які на ньому розташовані. Ця інформація доступна для збору кожному комп'ютеру, для цього треба тільки зберігати статистику дій відповідного користувача та статистику завантажень наявних файлів. В такому разі під вподобанням буде розумітися факт завантаження файлу. Якщо впровадити у P2P DHT мережу можливість оцінювання файлів, схема буде

трохи складнішою. Доступ до потрібної інформації, яка не розміщена на комп'ютері користувача, якому треба сформувати рекомендації, може бути отримана запитами до інших комп'ютерів за ідентифікаторами, організованими розподіленою хеш-таблицею P2P DHT мережі.

Також у [33] було використано хеш-таблиці з відкритою адресацією для збереження проміжних значень, необхідних для генерації рекомендацій, в тому числі для підрахунку кількості входжень елементів. Для цього було реалізовано операцію інкременту, яка працює наступним чином:

1. Здійснюється пошук елементу за ключем.
2. Якщо елемент не існує, створюється новий елемент зі значенням 1.
3. Якщо елемент існує і має значення -1, то його значення не змінюється, інакше – збільшується на 1.

Такий підхід дає можливість заборонити інкрементувати певні елементи.

Запропонований у [33] метод роботи рекомендаційної системи складається з наступних етапів:

Етап 1. Створити хеш-таблицю з ідентифікаторами користувачів H_u та хеш-таблицю з ідентифікаторами об'єктів H_f. Одержати доступ до розгорнутих списків, асоційованих з користувачами L_u та асоційованих з об'єктами L_f, що містять історію дій (завантаження/вподобання).

Етап 2. Переглянути список вподобань L_{u0} користувача U_0, для якого треба створити рекомендації. Для кожного вподобаного користувачем U_0 об'єкту створити у хеш-таблиці об'єктів H_f елемент зі значенням -1.

Етап 3. Для кожного користувача, який вподобав один з об'єктів, що в H_f мають значення -1, інкрементувати відповідний елемент у хеш-таблиці H_u.

Етап 4. Для кожного користувача U_i з хеш-таблиці користувачів H_u, елемент якого не містить -1, переглянути відповідний список вподобань L_{ui} та інкрементувати елементи у хеш-таблиці об'єктів H_f, якщо деякий користувач вподобав відповідний об'єкт.

Етап 5. Створити масив M_{rec} для збереження списку рекомендацій. Для кожного елементу з хеш-таблиці об’єктів H_f, значення якого не -1, здійснити спробу вставити цей елемент до масиву M_{rec} у порядку спадання значень. Обрати *TopN* рекомендацій з масиву M_{rec} sдля показу користувачу $U_{0.}$

На початку роботи запропонований метод позначає цільового користувача U_0 (якому треба сформувати рекомендації) як недоступного для участі у проміжних обчисленнях, шляхом занесення у хеш-таблицю користувачів елемента, ключ якого – це ідентифікатор цього користувача, а значення дорівнює -1.

Після цього виконується обхід списку вподобань цільового користувача. Кожен вподобаний об’єкт заноситься до хеш-таблиці об’єктів зі значенням -1, щоб уникнути повторної рекомендації уже вподобаних об’єктів. Якщо система допускає повторні рекомендації, то цей етап може бути змінений.

Далі для кожного вподобаного об’єкту вибирається список користувачів, які його вподобали. Для кожного користувача із цього списку виконується інкрементація відповідного значення у хеш-таблиці користувачів. Це значення буде використано для визначення ступеня подібності користувачів. Після завершення обходу хеш-таблиця користувачів фільтрується, щоб виключити користувачів, вподобання яких не будуть використовуватися при формуванні рекомендації.

Під час наступного етапу виконується обхід усіх елементів хеш-таблиці користувачів. Для кожного з цих користувачів вибирається відповідний список вподобань. Для кожного елементу з цього списку виконується інкрементація відповідного значення у хеш-таблиці об’єктів. На основі цього значення визначається, чи потрапить об’єкт до рекомендації.

На останньому етапі з хеш-таблиці об’єктів відбираються елементи із найбільшими значеннями та передаються як рекомендації для цільового користувача.

4.4. Зберігання даних моделі рекомендаційної системи у формі бінарних діаграм рішень

Бінарні діаграми рішення (або діаграми рішень на основі логіки) можуть бути корисними у моделюванні деяких складних мереж. Ці діаграми використовуються для представлення логічних рішень, де кожен вузол мережі має два стани – істину чи брехню (0 або 1). У складних мережах, де безліч параметрів взаємопов’язані, бінарні діаграми можуть допомогти спростити аналіз логічних залежностей та виявити основні вузли та шляхи, через які дані чи рішення проходять через мережу.

Однак, якщо мережа включає складні взаємодії або потребує врахування множини різних станів, бінарні діаграми можуть бути обмежені у своєму застосуванні. У таких випадках можуть бути корисними інші методи, такі як графи, імовірнісні моделі або нейромережні підходи.

Використання бінарних діаграм рішень виправдане, коли:

– Необхідна простота моделі та мінімізація обчислювальних витрат.

– Завдання зводиться до аналізу найпростіших логічних залежностей.

– Взаємозв'язки між компонентами мережі можуть бути модельовані через бінарні змінні (наприклад, стани увімкнено/вимкнено, підключено/відключено тощо).

Для більш складних мережевих моделей, скоріш за все, доведеться використовувати універсальні інструменти для аналізу мережевих структур.

Дані рекомендаційної системи зручно представляти у вигляді графу. Один з найбільш поширених способів зберігання графів у комп’ютерних програмах – використання матриці суміжності [36]. Матриця суміжності графу A містить значення булевого типу, тобто вона є матричним представленням булевої функції F, яка описує наявність дуги між двома вузлами. Якщо орграф G помічений, то необхідно також зберегти його вагу.

Представлення булевої функції як таблиці істинності чи досконалої

кон’юнктивної\диз’юнктивної нормальної форми потребує $\Omega(2^{2m+l})$ пам’яті та вимагає значних обчислень для визначення значення функції. У [33] досліджується використання бінарних діаграм рішень для зберігання графу даних рекомендаційної системи.

Бінарні діаграми рішень (БДР) – це економна форма представлення булевих функцій у вигляді орієнтованого ациклічного графу. Вершини графу представляють аргументи функції, листки – її двійкові значення [37]. Для додавання і вилучення ребер та зміни ваги ребер необхідно мати можливість редагувати дані графу. БДР дають можливість зберігати дані у стисненому вигляді та швидко отримувати значення функції за її параметрами, але редагування БДР вимагає складних обчислень. При представленні булевих функцій у формі БДР стало можливим розв’язувати багато проблем, які при традиційних представленнях структур нерозв’язні через значну розмірність таких представлень і складність операцій над ними. БДР можуть успішно застосовуватися фактично в кожній галузі, де потрібно обробляти дискретні структури даних.

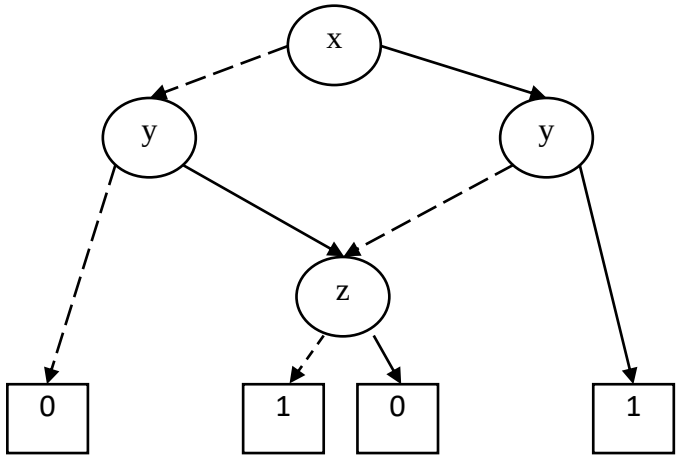


Рис. 4.9. Приклад бінарної діаграми рішень

Адаптація БДР для представлення орієнтованих графів з даними рекомендаціійної системи наведена нижче.

Коли дані рекомендаційної системи зберігаються у вигляді графу, то як правило вони мають наступний формат:

– вершини графу – користувачі та об’єкти (контент, товари, тощо) системи.

– ребра графу – дії користувачів по відношенню до об’єктів (перегляди, оцінки, тощо), відношення подоби, зв’язки рекомендацій типу користувачу-рекомендовано-об’єкт, тощо.

У [33] була досліджена можливість зберігати граф рекомендаційної системи у пам’яті у вигляді бінарної діаграми рішень. Для проведення експерименту по вивченню потреб в оперативній пам’яті було розроблено програмну модель спрощеної рекомендаційної системи, в якій було виділено три основні сутності – агент, сесія та предмет. Рекомендаційна система отримує наступні параметри: n_a – кількість агентів, n_s – кількість сесій, n_i – кількість предметів, n_{al} – максимальна кількість вподобань агента, n_{sl} – максимальний розмір сесії.

Рекомендаційна система в запропонованій у [33] програмній моделі працює за наступним алгоритмом:

1. Для кожного з n_a агентів випадковим чином генерується від 1 до n_{al} вподобань (рис. 4.10). При цьому унікальність вподобань не перевіряється, тому реальна кількість вподобань може виявитися менше.

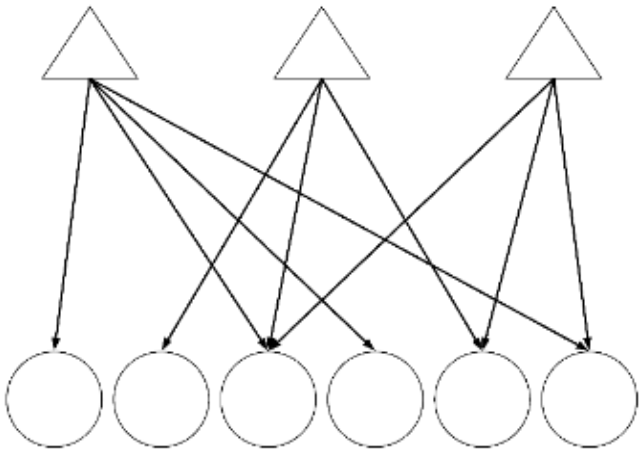


Рис. 4.10. Створення вподобань для агентів
(трикутники – агенти, круги – предмети)

2. Створюється n_s сесій. До кожної сесії закріплюється випадковим чином обраний агент. Потім серед вподобань цього агента випадковим чином обирається від 1 до $min(n_{al}, n_{sl})$ вподобань, які копіюються до сесії (рис. 4.11).

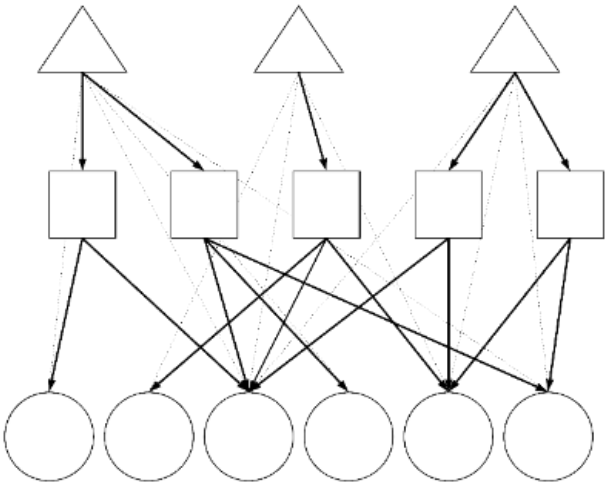


Рис. 4.11. Створення сесій та вподобань для них
(трикутники – агенти, квадрати – сесії, круги – предмети; сесія містить вподобання агента, які він зробив за одне відвідування ресурсу)

3. Випадковим чином обирається контрольна сесія, для якої буде сформовано рекомендацію.

4. Визначаються усі предмети, що належать до контрольної сесії (рис. 4.12).

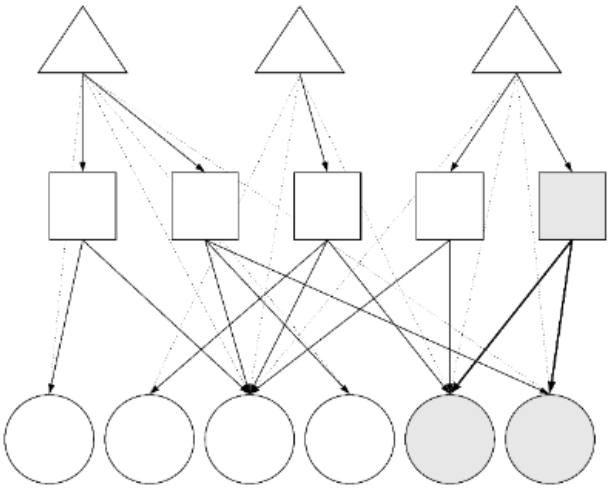


Рис. 4.12. Визначення контрольної сесії
(трикутники – агенти, квадрати – сесії, круги – предмети)

5. Здійснюється пошук усіх сесій, вподобання яких мають перетин із вподобаннями контрольної сесії (рис. 4.13). На цьому етапі є можливість відфільтрувати сесії за розміром перетину.

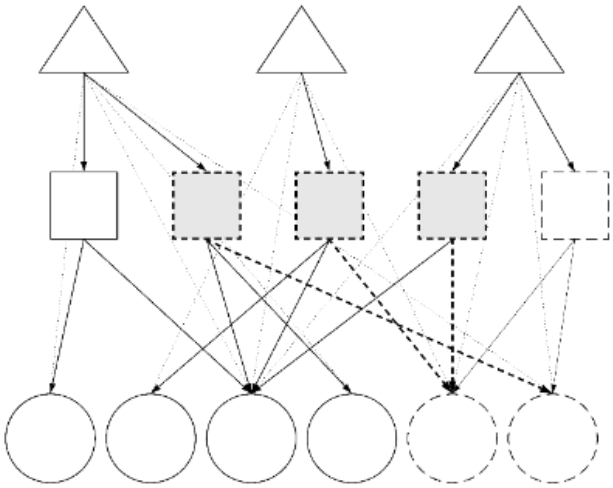


Рис. 4.13. Пошук сесій, які мають перетин із контрольною сесією (трикутники – агенти, квадрати – сесії, круги – предмети)

6. Визначаються предмети, які буде рекомендовано. Здійснюється пошук усіх предметів, які належать хоча б одній з відібраних сесій, але не належать до контрольної сесії (рис. 4.14). На цьому етапі є можливість відфільтрувати предмети за кількістю закріплених сесій.

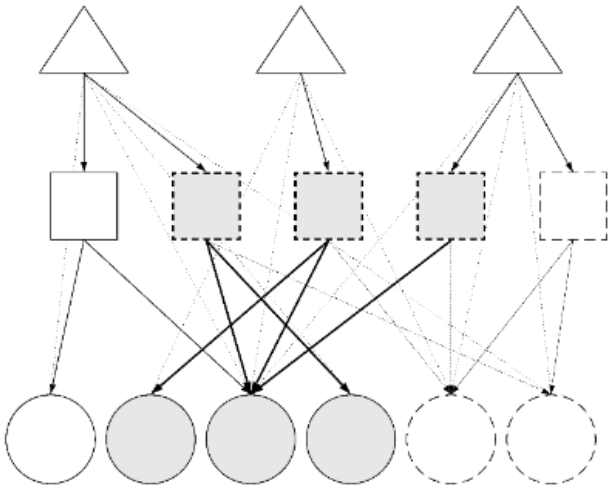


Рис. 4.14. Вибірка предметів для формування рекомендації (трикутники – агенти, квадрати – сесії, круги – предмети)

Етапи роботи запропонованого у [33] методу моделювання рекомендаційної системи полягають у наступному:

Етап 1. Ініціалізація рекомендаційної системи: задається кількість агентів, предметів та сесій, а також розмір сесії та максимальна кількість вподобань.

Етап 2. Для кожного агента випадковим чином генерується від 1 до n_{al} вподобань.

Етап 3. Створюється n_s сесій. До кожної сесії закріплюється випадковим чином обраний агент. Потім серед вподобань цього агента випадковим чином обирається від 1 до k вподобань, які копіюються до сесії.

Етап 4. Випадковим чином обирається контрольна сесія, для якої буде сформовано рекомендацію.

Етап 5. Визначаються усі предмети, які належать до контрольної сесії.

Етап 6. Здійснюється пошук усіх сесій, вподобання яких мають перетин із вподобаннями контрольної сесії. На цьому етапі є можливість відфільтрувати сесії за розміром перетину.

Етап 7. Визначаються предмети, які буде рекомендовано. Здійснюється пошук усіх предметів, які належать хоча б одній із відібраних сесій, але не належать до контрольної сесії. На цьому етапі є можливість відфільтрувати предмети за кількістю закріплених сесій.

Накопичення змін до БДР

Якщо алгоритм передбачає періоди інтенсивного редагування графу, можна тимчасово зберігати зміни у структурі даних, більш пристосованій до внесення змін. Можна застосувати ідею “гарячого” (хеш-таблиця) та “холодного” (БДР) сховищ. У хеш-таблицю заносяться параметри функції, на яких необхідно змінити значення, та саме значення. При досягненні певного розміру або по завершенню логічного блоку операцій із хеш-таблиці формується коригуюча БДР, яка потім зливається із основною. При читанні значень необхідно спочатку перевірити існування значення у хеш-таблиці, і у випадку його відсутності читати із БДР. Складність використання цього

підходу полягає у виборі булевої операції, яка буде використовуватися при злитті БДР. Одним з найкращих підходів є використання операції XOR стає, особливо коли необхідно зберігати інформацію не лише про факт наявності дуги між вузлами, а й зберігати вагу цієї дуги. Для цього можна виділити додаткову групу змінних, яка відповідатиме за представлення ваг у БДР.

Бінарні діаграми рішень мають як переваги, так і недоліки у порівнянні з хеш-таблицями, бітовими масивами та зв'язними списками. БДР дають можливість шукати дані за частковими ключами, що додатково дозволяє зберігати зв’язки більшої розмірності. Бінарні діаграми зберігають дані з меншими витратами пам’яті, ніж хеш-таблиці, завдяки чому у оперативній пам’яті одночасно можна розмістити інформацію про більшу кількість вподобань. БДР програють у швидкодії, проте нами було застосовано кілька оптимізацій:

– підтримка унікальності фрагментів БДР обчислуювально дуже дорога, тому вона була відділена від процедури редагування і виконується кілька разів за час генерації тестових даних;

– видалення надлишкових фрагментів з БДР спричиняє розрідженість вузлів у пам’яті, через що зростає час доступу до вузла. Тому після видалення надлишкових вузлів вони ущільняються шляхом перенесення у нову область пам’яті;

– ущільнення вузлів дає можливість звільнити певну кількість пам’яті, проте виявилося, що звільнення пам’яті у цій ситуації неефективне – наступні операції редагування вимагають створення нових вузлів, через що потрібно буде повторно переносити вузли між областями пам’яті;

– для здійснення пошуку у БДР її необхідно проіндексувати відповідно до ключа та параметрів пошуку, а обхід усіх вузлів БДР – це дорога операція. Для прискорення пошуку було покращено процедуру індексування, завдяки чому пошук з вказаною першою частиною ключа не вимагає повторної індексації.

Контрольні питання

1. Що таке складні мережі? Які їх властивості?
2. Які існують моделі складних мереж?
3. Як за допомогою складних мереж можна моделювати та тестувати рекомендаційні системи?
4. Які існують підходи до моделювання рекомендаційних систем у централізованих складних комп’ютерних системах?
5. Які існують підходи до моделювання рекомендаційних систем у децентралізованих складних комп’ютерних системах?
6. Чи доцільно використовувати бінарні діаграми рішень для моделювання рекомендаційних систем?
7. Як можна зберігати дані рекомендаційних систем у бінарних діаграмах рішень?
8. Як можна моделювати атаки на рекомендаційні системи?
9. Як можна моделювати поведінку користувачів у рекомендаційних системах?
10. Як можна здійснювати тестування рекомендаційних систем на основі програмних імітаційних моделей?

Список літератури до розділу

1. Agrawal R., Srikant R. Fast Algorithms for mining association rules // Proc. Int. Conf. on Very Large Databases. – 1994. – P. 487-499.
2. Barabási A.-L. Network science // Cambridge University Press. – 2018. – 475 p. – [Electronic resource] – Access mode: http://networksciencebook.com/
3. Мелешко Є. В. Методологія забезпечення стійкості рекомендаційних систем до дестабілізуючих факторів у комп’ютерних мережах: дис. ... докт. техн. наук з комп’ютерних систем та компонентів: 05.13.05 – Комп'ютерні

системи та компоненти / Є. В. Мелешко; Черкаський держ. технологічний ун-т. – Черкаси, 2020. – 323 с.

4. Пасічник В.В., Іванущак Н.М. Дослідження та моделювання складних мереж // Східно-Європейський журнал передових технологій, Вип. 2, № 3(44). – 2010. – С. 43-48.

5. Random Geometric Graph // NetworkX documentation – URL: https://networkx.org/documentation/stable/auto_examples/drawing/plot_random_geometric_graph.html#sphx-glr-auto-examples-drawing-plot-random-geometric-graph-py

6. Barabási A.-L., Albert R. Emergence of scaling in random networks (англ.) // Science, Vol. 286, No. 5439. – 1999. – P. 509-512. – [Electronic resource] – Access mode: https://doi.org/10.1126/science.286.5439.509

7. Barabási L.-A., Albert R., Jeong H. Diameter of the world-wide web // Nature, Vol. 401. – 1999. – P. 130-131.

8. Barabási L.-A., Albert R., Jeong H. Scale-free characteristics of random networks: the topology of the world-wide web // Physica, A281. – 2000. – 69-77.

9. Haidai B., Artiukh R., Malyeyeva O. Analysis and modelling the preferences of social networks users // Innovative technologies and scientific solutions for industries, No 1 (3). – 2018. – P. 5-12. – [Electronic resource] – Access mode: DOI: https://doi.org/10.30837/2522-9818.2018.3.005

10. Traag V.A. Algorithms and Dynamical Models for Communities and Reputation in Social Networks // Springer International Publishing. – 2014. – P. 229. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-3-319-06391-1

11. Watts D.J., Strogatz S.H. Collective dynamics of “small-world” networks // Nature, Vol. 393(6684). – 1998. – P. 440-442. – [Electronic resource] – Access mode: https://www.nature.com/articles/30918

12. He J., Chu W.W. A Social Network-Based Recommender System (SNRS) // In: Memon N., Xu J., Hicks D., Chen H. (eds) Data Mining for Social Network Data.

Annals of Information Systems, vol 12. Springer, Boston, MA. – 2010. – 32 p. – [Electronic resource] – Access mode: https://doi.org/10.1007/978-1-4419-6287-4_4

13. Erdös P. and Rényi A. On the evolution of random graphs // Publication of the Mathematical Institute of the Hungarian Academy of Sciences, Vol. 5. – 1960. – P. 17-61.

14. Bollobás B., Borgs C., Chayes T., Riordan O.M. Directed scale-free graphs // ProceedingSODA '03 Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. – 2003. – P. 132-139.

15. Bollobás B., Riordan O. Mathematical results on scale-free random graphs // Handbook of graphs and networks. – Weinheim: Wiley-VCH. – 2003. – P. 1-34.

16. Li Q.L. Nonlinear Markov processes in big networks // Special Matrices, Vol. 4(1). – 2016. – P. 202-217.

17. Newman M., Barabási A.-L., Watts D. J. The Structure and dynamics of networks // Princeton University Press. – 2006. – P. 592.

18. Шингалов Д.В., Мелешко Є.В., Улічев А.С. Дослідження Баєсових мереж довіри як засобів для моделювання динамічних процесів у складних мережах // Збірник тез XVII Міжнародної науково-практичної конференції «Математичне та програмне забезпечення інтелектуальних систем», м. Дніпро, 20-22 листопада 2019 р. – Дніпро: ДНУ. – 2019. – С. 284-285.

19. Neo4j Documentation // Official website of the graph database Neo4j. – [Electronic resource] – Access mode: https://neo4j.com/docs/

20. Wiki for project Gephi on GitHub // Web-site GitHub. – 2019. – [Electronic resource] – Access mode: https://github.com/gephi/gephi/wiki

21. Harper F.M., Konstan J.A. The MovieLens Datasets: History and Context // ACM Transactions on Interactive Intelligent Systems (TiiS). – 2015. – 19 p. – [Electronic resource] – Access mode: https://doi.org/10.1145/2827872

22. Riposo Ju. Diffusion on the Peer-to-Peer Network // LAP LAMBERT Academic Publishing. – 2022. – 100 p.

23. Koo S.G.M. Multimedia Content Distribution Using Peer-to-Peer Overlay Networks: The Design and Analysis of the Next Generation Peer-to-Peer Networks // VDM Verlag Dr. Müller. – 2008. – 88 p.

24. Milojicic D.S., Kalogeraki V., Lukose R., Nagaraja K., Pruyne J., Richard B., Rollins S., Xu Z. Peer-to-peer computing // Technical Report HPL-2002-57, HP Labs. – 2002. – 51 p. – URL: https://www.cs.kau.se/cs/education/courses/dvad02/p2/seminar4/Papers/HPL-2002-57R1.pdf

25. Zeinalipour-Yazti D., Kalogeraki V., Gunopulos D. Information retrieval techniques for peer-to-peer networks // Com-puting in Science & Engineering, Vol. 6, No. 4, pp. 20-26. – 2004. – DOI: 10.1109/MCSE.2004.12

26. Lua E.K., Crowcroft J., Pias M., Sharma R., Lim S. A Sur-vey and Comparison of Peer-to-Peer Overlay Network Schemes // IEEE Communications survey and tutorial. – 2004. – URL: https://snap.stanford.edu/class/cs224w-readings/lua04p2p.pdf

27. Kalogeraki V., Gunopulos D., Zeinalipour-Yazti D. A Local Search Mechanism for Peer-to-Peer Networks // Proc. of CIKM'02, McLean VA, USA, 2002. – URL: http://alumni.cs.ucr.edu/~csyiazti/downloads/papers/cikm02/cikm02.pdf

28. Zeinalipour-Yazti D. Information Retrieval in Peer-to-Peer Systems // M.Sc Thesis, Dept. of Computer Science, University of California Riverside. – 2003. – URL: http://alumni.cs.ucr.edu/~csyiazti/papers/msc/html/

29. Kademlia: A Design Specification – 2010. – URL: https://xlattice.sourceforge.net/components/protocol/kademlia/specs.html

30. Stoica I. , Morris R., Karger D.R., Kaashoek M.F., Balakrishnan H. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications // ACM SIGCOMM Computer Communication Review, Vol. 31(4). – 2001. DOI: 10.1145/964723.383071

31. The BitTorrent Protocol Specification – 2017. – URL: http://www.bittorrent.org/beps/bep_0003.html

32. Gnutella Protocol Development – 2003. – URL: https://rfc-gnutella.sourceforge.net

33. Міхав В. В. Модель та методи збору і обробки даних для рекомендаційних систем у peer-to-peer комп'ютерних мережах : дис. ... д-ра філософії з комп'ютерної інженерії: 123 Комп'ютерна інженерія / В. В. Міхав; Черкаський держ. технологічний у-т. – Черкаси, 2023. – 179 с.

34. Міхав В.В., Мелешко Є.В., Якименко М.С., Бащенко Д.В. Методи зберігання даних рекомендаційної системи на основі зв'язних списків // Системи управління, навігації та зв'язку – Полтава: ПНТУ, 2021. – Т. 4(66). – С. 59-62. – DOI: https://doi.org/10.26906/SUNZ.2021.4.059

35. Harper F.M., Konstan J.A. The MovieLens Datasets: History and Context // ACM Transactions on Interactive Intelligent Systems (TiiS). – 2015. – 19 p. – URL: https://doi.org/10.1145/2827872

36. Knuth D. E. The Art of Computer Programming. Vol. 1: Fundamental Algorithms. 3rd ed. – Boston: Addison-Wesley. – 1997. – 672 p.

37. Knuth D. E. The Art of Computer Programming. Vol. 4A: Combinatorial Algorithms – Boston: Addison-Wesley. – 2011. – 912 p.

НАВЧАЛЬНЕ ВИДАННЯ

МЕЛЕШКО Єлизавета Владиславівна,
МІХАВ Володимир Володимирович

РЕКОМЕНДАЦІЙНІ СИСТЕМИ У СКЛАДНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Навчальний посібник

Редактор Міхав В.В.
Комп'ютерна графіка Мелешко Є.В.
Обкладинка Мелешко Є.В.
Комп'ютерний набір та верстка Мелешко Є.В.

Видавець ЦНТУ
25000, м. Кропивницький, просп. Університетський, 8
тел.: (0522) 55-92-66