Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: **Методи кластеризації облікових записів користувачів для систем обміну повідомленнями**

Виконав(ла): студент(ка) 6 курсу, групи Сам-61
спеціальності 124 Системний аналіз
(шифр і назва спеціальності)

(підпис) Заверуха С.С. (прізвище та ініціали)

Керівник (підпис) Готович В.А. (прізвище та ініціали)

Нормоконтроль (підпис) Мацюк О.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Загородна Н.В. (прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 11 » грудня 2020 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 124 «Системний Аналіз»
(шифр і назва спеціальності)

студенту Заверусі Сергію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи кластеризації облікових записів користувачів для систем обміну повідомленнями

Керівник роботи Готович В.А., к.т.н., ст. викл.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *06* » *листопада* 2020 року № *4/7-830*

2. Термін подання студентом завершеної роботи 18 грудня 2020 року

3. Вихідні дані до роботи Наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
1 Кластеризація даних
2 Огляд відомих систем кластеризації користувачів
3 Практична реалізація кластеризації користувачів
4 Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Титульний слайд , Актуальність теми роботи, Мета дослідження , Предмет Об'єкт та наукова новизна дослідження, Завдання роботи, відміності кластеризації та класифікації, Базовий агломератний алгоритм кластеризації, Основні підходи до побудови кластерів, Відомі рішення, Основні проблеми існуючих рішень, Перелік використаних технологій , Архітектура, Метод Джаккара, java concurrency, Результати, Висновок

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека в надзвичайних ситуаціях	Стадник І.Я.		
Охорона праці	Дмитроца Л.П.		

7. Дата видачі завдання 21 вересня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	21.09.20-27.09.20	*Виконано*
2.	Підбір наукових джерел щодо інтелектуального аналізу	28.09.20-04.10.20	*Виконано*
	даних, кластеризації та класифікації		
3.	Переклад та опрацювання наукових джерел щодо	05.10.20-11.10.20	*Виконано*
	ієрархічної кластеризації		
4.	Виконання дослідження щодо частково розподіленого	12.10.20-18.10.20	*Виконано*
	Алгоритму ієрархічної кластеризації		
5.	Оформлення розділу «Кластеризація даних»	19.10.20-25.10.20	*Виконано*
6.	Оформлення розділу «Огляд відомих систем	26.10.20-01.11.20	*Виконано*
	Кластеризації користувачів»		
7.	Оформлення розділу «Практична реалізація»	02.11.20-08.11.20	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	09.11.20-15.11.20	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	16.11.20-22.11.20	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	23.11.20-29.11.20	*Виконано*
11.	Нормоконтроль	30.11.20-05.12.20	*Виконано*
12.	Перевірка на плагіат	25.12.2020	*Виконано*
13.	Попередній захист кваліфікаційної роботи	14.12.20	*Виконано*
14.	Захист кваліфікаційної роботи	21.12.2020	

Студент _____ Заверуха С.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи кластеризації облікових записів користувачів для систем обміну повідомлень// Дипломна робота освітнього рівня "Магістр" // Заверуха Сергій Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група САм-61 // Тернопіль, 2020 // С. – 88, рис. – 30, табл. – 0, додат. – 5, бібліогр. – 46.

Ключові слова: МЕТОДИ КЛАСТЕРИЗАЦІЇ, ІЄРАРХІЧНА КЛАСТЕРИЗАЦІЯ, РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ.

У кваліфікаційній роботі магістра проведено дослідження методів ієрархічної кластеризації, а також розроблено пришвидшений метод ієрархічної кластеризації шляхом використання засобів багатопотокового програмування.

У першому розділі було зроблено короткий огляд технологій добування даних, розглянуто цілі і властивості кластерів. Крім цього, було розглянуто відмінності між кластеризацією та класифікацією.

В другому розділі був проведений огляд основних методів побудови ієрархічних кластерів, основні відмінності з центроїдними та статистичними моделями. Визначено слабкі сторони ієрархічної моделі та представлено спосіб вирішення проблем з швидкістю виконання.

Третій розділ містить вимоги до створюваного програмного продукту та короткий огляд використовуваних інструментів.

В четвертому розділі показано процес розробки розподіленої логіки ієрархічної кластеризації. Представлено результати тестування і виконання створеної моделі на сучасних багатопотокових системах.

ANNOTATION

Methods for clustering user accounts for messaging systems // Diploma thesis Master degree // Zaverukha Sergiy Serhiyovych // Ternopil' Ivan Pul'uj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science, group SAm-61 // Ternopil, 2020 // Pages – 88, Fig. – 30, Tables – 0, Annexes – 5, References – 46.

Keywords: METHODS OF CLUSTERIZATION, HIERARCHIC CLUSTERIZATION, DISTRIBUTED CALCULATIONS.

In the qualification work of the master the research of methods of hierarchical clustering is carried out, and also the accelerated method of hierarchical clustering by use of means of multithreaded programming is developed.

In the first section, a brief overview of data mining technologies was made, and the goals and properties of clusters were considered. In addition, the differences between clustering and classification were considered.

The second section reviews the main methods of constructing hierarchical clusters, the main differences with centroid and statistical models. The weaknesses of the hierarchical model are identified and a way to solve speed problems is presented.

The third section contains the requirements for the created software product and a brief overview of the tools used.

The fourth section shows the process of developing a distributed logic of hierarchical clustering. The results of testing and execution of the created model on modern multithreaded systems are presented.

ЗМІСТ

ДОДАТКИ

ВСТУП

Актуальність теми роботи. Ієрархічна кластеризація є надзвичайно точним методом поділу інформації на окремі зв’язні групи, проте основними проблемами даного методу є непомірне використання оперативної пам’яті та велика часова складність алгоритму.

Метою дослідження є спроба зменшити використання пам’яті і зменшення часової складності алгоритму.

Для вирішення даної проблеми необхідно розв’язати такі **задачі:**

- здійснити аналіз існуючих методів кластеризації;
- проаналізувати створення n-вимірних векторів;
- проаналізувати методи порівняння n-вимірних векторів;
- проаналізувати методи швидкої побудови матриць подібностей;
- проаналізувати методи створення потокобезпечних структур та компонентів;
- здійснити вибір оптимального інструментарію для вирішення поставлених завдань.

Об’єкт дослідження – методологія побудови ієрархічних дерев з використанням засобів паралельного програмування.

Предмет дослідження – сукупність теоретично-методичних основ і утилітарних проблем побудови інформаційних систем з підтримкою кластеризації користувачів.

Наукова новизна – розробка системи кластеризації користувачів інформаційних систем на основі n-вимірних векторів інтересів.

Практичне значення одержаних результатів – можливість подолання проблеми надмірного використання пам’яті та зменшення часової складності.

Апробація результатів роботи. Результати кваліфікаційної роботи представлено на двох наукових конференціях:

- IX Міжнародна науково-технічна конференція молодих учених та студентів «Актуальні задачі сучасних технологій», на тему “Використання бінарних n-вимірних векторів для встановлення міри подібності користувачів інформаційних систем”;
- VIII науково-технічна конференція «Інформаційні моделі, системи та технології», на тему “Використання засобів багатопотокового програмування для пришвидшення побудови матриці подібностей n-вимірних векторів”.

1 КЛАСТЕРИЗАЦІЯ ДАНИХ

1.1 Інтелектуальні технології добування даних

Видобування даних – це процес автоматичного пошуку корисної інформації у великих сховищах даних. Методи видобутку даних застосовуються для пошуку нової корисної інформації в великому наборі даних, котрі без використання методів видобування даних можуть залишитись невідомими. Вони також можуть прогнозувати результати майбутніх спостережень опираючись на раніше отримані факти [1].

Варто зазначити, що не всі завдання пошуку інформації вважаються інтелектуальним аналізом даних. Прикладом даних завдань є пошук веб-сторінок з допомогою пошукового додатку чи використання системи керування базами даних для пошуку окремих записів. Проте це, методи інтелектуального аналізу використовуються для поліпшення систем пошуку інформації.

Видобуток даних є важливим елементом в процесі пошуку нових знань в базі даних. Даний процес представляє собою перетворення необроблених даних в корисну інформацію (рисунок 1.1). Даний процес складається з ряду етапів трансформації, від попередньої обробки даних до подальшої обробки результатів видобутку даних.

Рисунок 1.1 – Процес добування інформації з даних

Вхідні дані можуть зберігатися у різних форматах (електронні таблиці, реляційні таблиці та інші.) та можуть знаходитись у централізованому сховищі даних або розподілятися між кількома базами даних. Метою попередньої обробки є перетворення необроблених вхідних даних у відповідний формат для подальшого аналізу. Етапи попередньої обробки даних включають злиття даних з декількох джерел, очищення даних для усунення шуму та повторюваних спостережень, а також вибір записів та функцій, які мають відношення до задачі аналізу даних. Через безліч способів збирання та зберігання даних, попередня обробка даних є найбільш складним кроком у загальному процесі виявлення нових знань.

Отже, видобування даних - це технологія, яка поєднує традиційні методи аналізу даних із складними алгоритмами для обробки великих обсягів даних. Дана технологія представляє новий підхід для вивчення та аналізу нових типів даних та аналізу старих типів даних новими способами [2].

1.2 Визначення кластерного аналізу

Кластерний аналіз групує об'єкти даних лише на основі інформації, що міститься в даних, що описує об'єкти та їх взаємозв'язки. Мета полягає в тому, щоб об’єкти в межах групи були подібними (або пов’язаними) один з одним і відрізнялися від (або не пов’язані) об’єктів з інших груп. Чим більша подібність (або однорідність) всередині групи і чим більша різниця між групами, тим краща або виразніша кластеризація [3].

У багатьох додатках поняття кластеру не є чітко визначеним. Для розуміння складності визначення, що являє собою кластер, варто розглянути рисунок 1.2, який показує двадцять точок та три різні способи поділу їх на кластери. Форми маркерів вказують на приналежність до кластера. Рисунки 1.2 (b) та 1.2 (d) поділяють дані відповідно на дві та шість частин. Поділ кожного з двох більших скупчень на три підгрупи є особливістю сприйняття людини

зображення об’єктів на декартовій площині. Також, даний набір точок можна поділити на чотири кластери, як зображено на рисунку 1.2 (c). Цей рисунок ілюструє, що визначення кластера є неточним і що найкраще визначення залежить від природи даних та бажаних результатів.

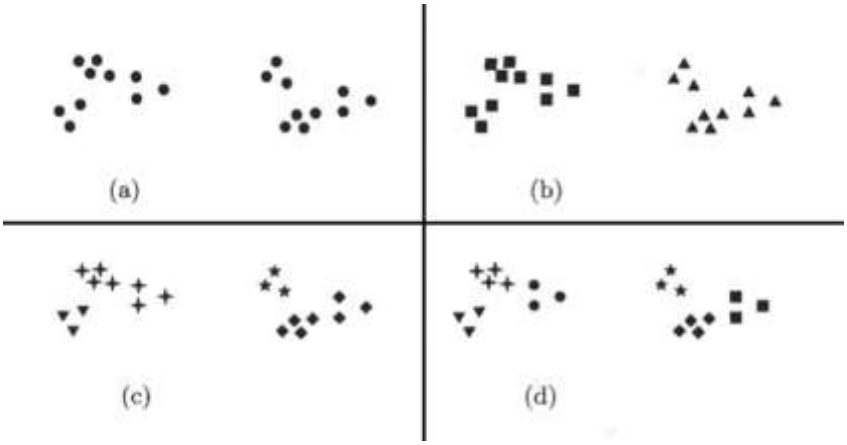

Рисунок 1.2 – Розбиття набору даних на довільну кількість кластерів

Кластерний аналіз пов’язаний з іншими методами, які використовуються для поділу об’єктів даних на групи. Наприклад, кластеризацію можна розглядати як форму класифікації, оскільки вона здійснює маркування об'єктів мітками класу (кластера). Класифікація в широкому розумінні є класифікацією під наглядом; тобто, новим, не позначеним об’єктам присвоюється мітка класу за допомогою моделі, розробленої з об’єктів із відомими мітками класів. З цієї причини кластерний аналіз іноді називають класифікацією без нагляду. Коли термін класифікація використовується без будь-якої кваліфікації в рамках аналізу даних, він зазвичай відноситься до контрольованої класифікації [4].

Крім того, терміни “сегментація” та “розбиття” іноді використовуються як синоніми кластеризації, ці терміни часто використовуються для підходів поза традиційними межами кластерного аналізу. Наприклад, термін розбиття часто використовується у зв'язку з методами, які поділяють графи на підграфи і які не пов'язані з кластеризацією. Сегментація часто стосується поділу даних на групи за допомогою простих методів; зображення можна розділити на сегменти лише

на основі інтенсивності пікселів та кольору, або людей можна розділити на групи залежно від їх доходу. Тим не менше, деякі завдання з розділення графіків та сегментації зображень та ринку пов'язані з кластерним аналізом.

Отже, основна відмінність кластеризації і класифікації[5] це наявність об’єктів з відомими мітками та задана кількість бажаних кластерів.

1.3 Задачі та сфери застосування кластеризації даних

Кластерний аналіз поділяє дані на групи (кластери), які є значущими, корисними або мають дані обидві риси одночасно. Якщо метою є значущі групи, то кластери повинні фіксувати природну структуру даних. Однак у деяких випадках кластерний аналіз є лише корисною відправною точкою для інших цілей, таких як узагальнення даних. Будь то для розуміння чи корисності, кластерний аналіз вже давно відіграє важливу роль у найрізноманітніших галузях: психологія та інші соціальні науки, біологія, статистика, розпізнавання образів, пошук інформації, машинне навчання та видобуток даних [4].

Класи або концептуально значущі групи об’єктів, що мають спільні характеристики, відіграють важливу роль у тому, як люди аналізують і описують світ. Людський розум вміє ділити об'єкти на групи (кластеризація) та віднести певні об'єкти до цих груп (класифікація). Прикладом цього є те, як відносно маленькі діти можуть швидко позначити об'єкти на фотографії як будівлі, транспортні засоби, людей, тварин, рослини тощо. У контексті розуміння даних, кластери є потенційними класами, а кластерний аналіз - це вивчення методів автоматичного пошуку класів [5].

Біологи витратили багато років на створення таксономії (ієрархічної класифікації) усього живого: царства, роду, класу, порядку, сім’ї, роду) та видів. Таким чином, можливо не дивно, що більша частина перших робіт з кластерного аналізу прагнула створити дисципліну математичної систематики, яка могла б автоматично знаходити такі класифікаційні структури. Зовсім недавно біологи

застосували кластеризацію для аналізу великої кількості генетичної інформації, яка зараз доступна. Наприклад, кластеризація була використана для пошуку груп генів, які мають подібні функції.

Мережа інтернет складається з мільярдів веб-сторінок, і результати запиту до пошукової системи можуть повернути тисячі сторінок. Кластеризація може бути використана для групування цих результатів пошуку в невелику кількість кластерів, кожен з яких фіксує певний аспект запиту. Наприклад, запит "фільм" може повернути веб-сторінки, згруповані за такими категоріями, як огляди, трейлери, зірки та кінотеатри. Кожну категорію (кластер) можна розділити на під категорії (під кластери), створюючи ієрархічну структуру, яка надалі допомагає користувачеві досліджувати результати запиту.

Кластерний аналіз забезпечує абстракцію від окремих об’єктів даних до кластерів, в яких ці об’єкти даних знаходяться. Крім того, деякі методи кластеризації характеризують кожен кластер з точки зору прототипу кластера; тобто об’єкт даних, який є представником інших об’єктів у кластері. Ці кластерні прототипи можуть бути використані як основа для ряду методів аналізу даних або обробки даних. Тому в контексті корисності кластерний аналіз - це вивчення методів пошуку найбільш репрезентативних кластерних прототипів.

Багато методів аналізу даних, такі як регресія або PCA, мають часову чи просторову складність O (n^2) або вище (де n - кількість об'єктів), таким чином використання великих наборів даних є не практичним. Однак замість того, щоб застосовувати алгоритм до всього набору даних, його можна застосувати до зменшеного набору даних, що складається лише з прототипів кластера. Залежно від типу аналізу, кількості прототипів та точності, з якою прототипи представляють дані, результати можуть бути порівнянними з тими, які були б отримані, якби всі дані могли бути використані.

Також кластерні прототипи також можуть бути використані для стиснення даних. Зокрема, створюється таблиця, яка складається з прототипів для кожного кластера, в такому випадку кожному прототипу присвоюється ціле число, яке є

його позицією (індексом) у таблиці. Кожен об'єкт представлений індексом прототипу, пов'язаного з його кластером. Цей тип стиснення відомий як векторне квантування і часто застосовується до зображень і звуку , де присутня велика кількість подібні об’єктів один до одного, втрата інформації є прийнятною, і бажане суттєве зменшення обсягу даних.

1.4 Цілі і властивості кластерів

Кластеризація спрямована на пошук корисних груп об’єктів (кластерів), де корисність визначається цілями аналізу даних. Існує кілька різних понять кластеру, які виявляються корисними на практиці.

Добре відокремлений кластер - це сукупність об'єктів, у яких кожен об'єкт знаходиться ближче (або більше схожий) з будь-яким іншим об'єктом кластера, ніж з будь-яким об'єктом, що не входить в кластер. Іноді поріг використовується, щоб вказати, що всі об'єкти кластера повинні бути досить близькими (або подібними) один до одного. Це ідеалістичне визначення кластера виконується лише тоді, коли дані містять природні кластери, досить далекі один від одного (рисунок 1.3). Дану ситуацію можна представити як добре відокремлених скупчень, що складається з двох груп точок у двовимірному просторі. Відстань між будь-якими двома точками в різних групах більше, ніж відстань між будь-якими двома точками в межах групи. Добре відокремлені скупчення не повинні бути кулястими, але можуть приймати будь-яку форму.

Рисунок 1.3 – Добре відокремлені кластери

Кластер на основі прототипу - це сукупність об’єктів, у яких кожен об’єкт знаходиться ближче (більш схоже) до прототипу, який визначає кластер, ніж до прототипу будь-якого іншого кластера. Для даних з безперервними атрибутами прототипом кластера часто є центроїд, тобто середнє (середнє) усіх точок кластера. Коли центроїд (рисунок 1.4) не має значення, наприклад, коли дані мають категоріальні атрибути, прототип часто є найбільш репрезентативною точкою кластера. Для багатьох типів даних прототип можна розглядати як найбільш центральну точку, в такому випадку кластери, що базуються на прототипах, називаються як кластери, засновані на центрі. Не дивно, що такі скупчення, як правило, бувають кулястими.

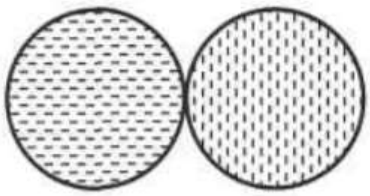

Рисунок 1.4 – Центральні кластери (кластери засновані на центрі)

Кластер на основі графа. Якщо дані представлені у вигляді графа, де вузли є об'єктами, а посилання представляють зв'язки між об'єктами, тоді кластер можна визначити як пов'язаний компонент; тобто групу об’єктів, які зв’язані між собою, але не мають зв’язку з об’єктами поза групою. Важливим прикладом кластерів на основі графів є кластери на основі суміжності, де два об'єкти з'єднані лише в тому випадку, якщо вони знаходяться на певній відстані один від одного. Це означає, що кожен об'єкт кластера, що базується на суміжності, знаходиться ближче до якогось іншого об'єкта кластера, ніж до будь-якої точки іншого кластера. Дане визначення кластера є корисним, коли кластери є нерегулярними або переплетеними, але можуть мати проблеми, коли присутній шум, оскільки невеликий міст точок може об'єднати два окремі кластери.

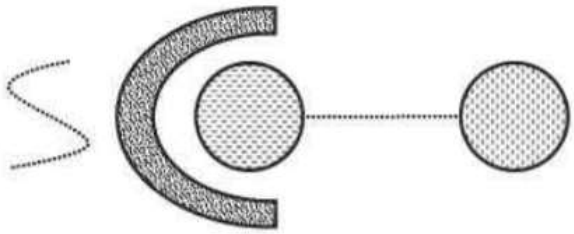

Рисунок 1.5 – Кластери на основі суміжностей

Кластер на основі щільності являє собою щільну область об’єктів, яка оточена областю низької щільності. Рисунок 1.6 показує деякі кластери на основі щільності даних, створених додаванням шуму до даних рисунка 1.5 . Два кругові скупчення не об’єднані, як на рисунку 1.5, оскільки міст між ними зникає в шум. Аналогічно, крива, яка присутня на рисунку 1.5 , також переходить у шум і не утворює скупчення на рисунку 1.6. Визначення кластера на основі щільності часто застосовується, коли кластери є нерегулярними або переплетеними, а також коли присутній шум і викиди. На відміну від цього, визначення кластера на основі суміжності не буде добре працювати для даних на Рисунку 1.6, оскільки шум, як правило, утворює сітки між кластерами.

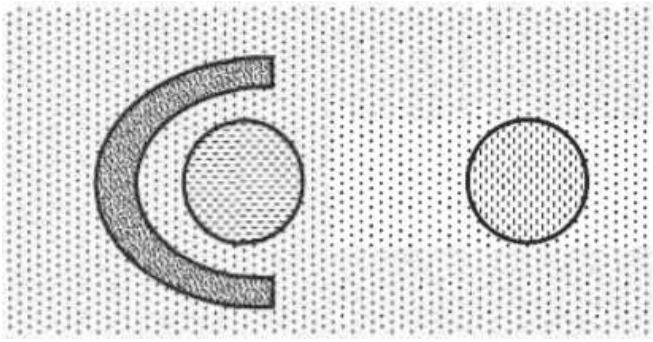

Рисунок 1.6 – Кластер на основі щільності

Shared-Property (концептуальні кластери). Більш загально, ми можемо визначити кластер як набір об'єктів, що мають спільну властивість. Це визначення охоплює всі попередні визначення кластера; наприклад, об’єкти в центральному кластері діляться властивістю, що всі вони найближчі до одного

центроїда. Однак підхід до спільних властивостей також включає нові типи кластерів. Для розуміння даного підходу необхідно розглянути кластери, зображені на рисунку 1.7.

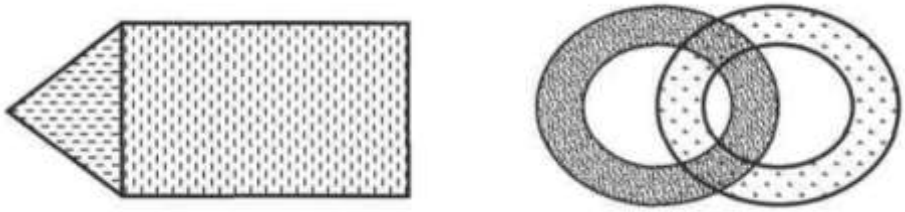

Рисунок 1.7 – Концептуальні кластери

Трикутна область (скупчення) примикає до прямокутної, і є два переплетені кола (скупчення). В обох випадках алгоритму кластеризації потрібна дуже конкретна концепція кластера для успішного виявлення цих кластерів. Процес пошуку таких кластерів називається концептуальною кластеризацією. Однак складне уявлення про кластер заведе нас у область розпізнавання шаблонів що є доволі складним завданням.

1.5 Методи кластерного аналізу

В кластерному аналізу є три простих методи кластеризації, котрі варто розглянути в першу чергу, щоб представити багато концепцій, залучених до кластерного аналізу.

- K-середніх. Це методика розділової кластеризації на основі прототипу, яка намагається знайти вказану користувачем кількість кластерів (K), які представлені в вигляді центроїдів;
- агломератна ієрархічна кластеризація. Цей підхід кластеризації відноситься до набору тісно пов'язаних методів кластеризації, котрі реалізують ієрархічну кластеризацію, починаючи з кожної точки як одиночний кластер, а потім неодноразово об'єднуючи два найближчі кластери, поки не залишиться

єдиний, всеосяжний кластер. Деякі з цих методів мають природну інтерпретацію з точки зору кластеризації на основі графів, тоді як інші мають інтерпретацію з точки зору підходу на основі прототипу;

- DBSCAN. Це алгоритм кластеризації на основі щільності, який створює кластеризацію з розділами, в якій кількість кластерів автоматично визначається алгоритмом. Точки в регіонах з низькою щільністю класифікуються як шумові та опущені; таким чином, DBSCAN не забезпечує повної кластеризації.

1.6 Кластерна еквівалентність

У контрольованій класифікації оцінка отриманої результуючої моделі є невід'ємною частиною процесу розробки класифікаційної моделі, і існують загальновизнані заходи та процедури оцінки, наприклад, точність та перехресна перевірка, відповідно. Однак за своєю суттю кластерна оцінка не є загальновживаною частиною кластерного аналізу. Тим не менше, кластерна оцінка або валідація кластера, як її традиційно називають, є важливою, для перевірки точності моделі, тому варто розглянути деякі найпоширеніші та найпростіші підходи.

Існує певна імовірність плутанини щодо того, чому необхідна оцінка кластера. Оскільки кластерний аналіз є частиною дослідницького аналізу даних та проводиться неодноразово. Тому, оцінка здається надмірно складним доповненням, для неформального процесу кластеризації. Крім того, існує ціла низка різних типів кластерів - у якомусь сенсі кожен алгоритм кластеризації визначає свій тип кластера, тому кожна ситуація може вимагати різного виміру оцінки. Наприклад, кластери K-середніх можуть оцінюватися з точки зору залишкової суми квадратів, але для кластерів на основі щільності, які не обов'язково повинні бути центроїдами, SSE в більшості випадків не буде працювати.

1.7 Ієрархічна кластеризація

Ієрархічні методи кластеризації є другою важливою категорією методів кластеризації. Як і у випадку з K-засобами, ці підходи порівняно з багатьма алгоритмами кластеризації є відносно старими, але вони все ще користуються широким використанням. Існує два основних підходи для створення ієрархічної кластеризації: агломеративний та розподільний.

Агломеративний алгоритм починається з точок як окремих кластерів і на кожному кроці об’єднуються найближчі пару кластерів. Це вимагає визначення поняття близькості кластера.

На початку роботи розподільного алгоритму є один всеохоплюючий кластер і на кожному кроці даний ділиться кластер, поки не залишаться лише одиночні кластери окремих точок. У даному випадку необхідно вирішити, який кластер розділити на кожному кроці і як це зробити.

Для наборів двовимірних точок, ієрархічну кластеризацію також можна графічно представити за допомогою вкладеної діаграми кластера. На рисунку 1.8 показаний приклад цих двох типів фігур для набору з чотирьох двовимірних точок.

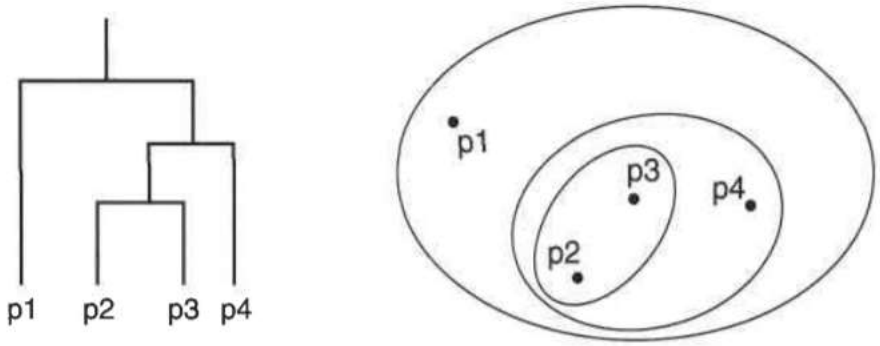

Рисунок 1.8 – Дендрограма (зліва) і діаграма кластера (справа)

Ієрархічна кластеризація часто відображається графічно за допомогою деревоподібної діаграми, званої дендрограмою, яка відображає як відносини кластер-підкластер, так і порядок об'єднання кластерів (агломератний вигляд) або розбиття (роздільний).

1.8 Базовий агломератний ієрархічний кластерний алгоритм

Багато агломератних ієрархічних методів кластеризації є варіаціями одного і того же підходу: починаючи з окремих точок як кластерів, послідовно об'єднують два найближчі кластери, поки не залишиться лише один кластер.

Алгоритм що реалізує даний метод, складається з наступних кроків:

1. За необхідності обчислюється матриця близькості.
2. Мітка.
3. Об’єднуються найближчі два кластери.
4. Оновлюється матриця близькості, щоб відобразити близькість між новим кластером та початковими кластерами.
5. Переходимо на крок 2 що містить мітку, до тих пір поки не залишиться один кластер

Ключовою операцією вище наведеного алгоритму є обчислення близькості між двома кластерами, і саме визначення близькості кластера відрізняє різні агломератні ієрархічні прийоми. Близькість кластера зазвичай визначається з урахуванням певного типу кластера. Наприклад, багато агломератних ієрархічних методів кластеризації, таких як MIN, MAX та Group Average, походять із графічного подання кластерів. MIN визначає близькість кластера як близькість між найближчими двома точками, що знаходяться в різних кластерах, або використовуючи графічні терміни, найкоротший край між двома вузлами в різних підмножинах вузлів. Що формує кластери на основі суміжності, як показано на рисунку 1.5. Альтернативним варіантом є використання методу MAX, що приймає близькість між найвіддаленішими двома точками в різних

кластерах як близькість кластера, або використовуючи терміни графіків, найдовший край між двома вузлами в різних підмножинах вузлів. (Якщо наша близькість - це відстані, то методи, MIN та MAX, є короткими та сугестивними. Однак для знаходження схожості, методи можуть задаватись іншими способами при умові що віддалені точки вказують на віддалені кластери. З цієї причини нам необхідно використовувати альтернативні методи, одинарне посилання та повне посилання відповідно.) Даний підхід, заснований на графіках, середня техніка групи, визначає близькість кластера як середню попарну близькість (середню довжину ребер) усіх пар точок з різних кластерів. Рисунок 1.9 ілюструє ці три підходи.

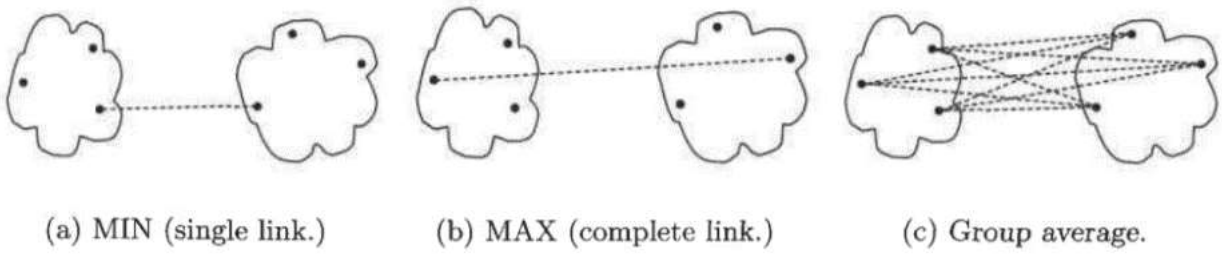

Рисунок 1.9 – Графічне визначення наближення(близькості) кластерів

В окремих випадках ми використовуємо подання даних на основі прототипу, в якому кожен кластер представлений центроїдом, різні визначення близькості кластера є більш природними. При використанні центроїдів близькість кластера зазвичай визначається як близькість між центроїдами кластеру. Альтернативний рішенням є використання метода Уордтса, що розглядає кластер як його центроїд, а відстань між кластерами є середнє квадратичне відхилення, що виникає в результаті злиття двох кластерів. Метод Уорда як і метод K-середніх намагається мінімізувати суму квадратних відстаней точок від їх кластерних центроїдів.

Базовий агломератний ієрархічний алгоритм кластеризації використовує матрицю близькості. Даний крок вимагає збереження $\frac{1}{2}m^2$ елементів (якщо

матриця близькості симетрична), де m - кількість точок даних. Простір, необхідний для відстеження кластерів, пропорційний кількості кластерів, що дорівнює m-1, за винятком одноелементних кластерів. Отже, загальна складність простору становить O (m^2).

Для обчислення матриці близькості потрібен час O (m^2). Після цього кроку існує m-1 ітерацій, оскільки на початку існує m кластерів, та під час кожної ітерації об'єднуються два кластери. Якщо пошук матриці близькості виконується лінійним способом то для виконання і-ї ітерації потрібен час O ($(m - i + 1)^2$), який пропорційний поточній кількості кластерів у квадраті. Наступний крок вимагає лише O ($m - i + 1$) для оновлення матриці близькості після злиття двох кластерів Без змін це призведе до часової складності O (m^3). Якщо відстані від кожного кластера до всіх інших кластерів зберігаються як відсортований список (або купа), можна зменшити вартість пошуку двох найближчих кластерів до O ($m - i + 1$). Однак, через додаткову складність збереження даних у відсортованому списку або купі, загальний час, необхідний для ієрархічної кластеризації становить O ($m^2 \log m$).

Просторова та часова складність ієрархічної кластеризації сильно обмежує розміри наборів даних, які можна обробити.

1.9 Підходи до побудови ієрархічних кластерів

Щоб проілюструвати поведінку різних алгоритмів ієрархічної кластеризації, необхідно використати зразкові дані, які складаються з 6 двовимірних точок, які показані на рисунку 1.10.

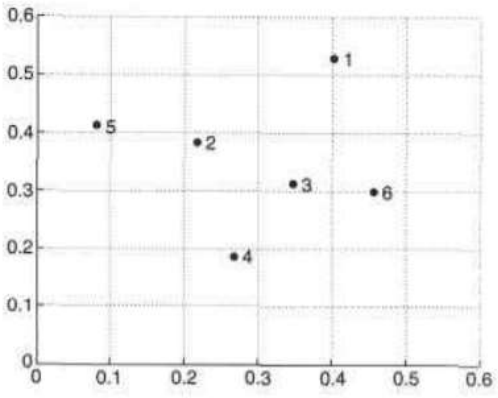

Рисунок 1.10 – Набір даних що представлений 6 двовимірними точками

Координати r та g точок та евклідова відстань між ними наведені на рисунку 1.11.

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Рисунок 1.11 – Евклідова відстань між вузлами

В одноканальному або MIN версії ієрархічної кластеризації близькість двох кластерів визначається як мінімум відстані (максимум подібності) між будь-якими двома точками в двох різних кластерах. Метод єдиної ланки добре справляється з нееліптичними фігурами, але чутлива до шуму та викидів.

Рисунок 1.12 показує результат застосування техніки єдиного посилання до для прикладу набору даних із шести точок. Рисунок 1.12 (a) показує вкладені кластери як послідовність вкладених еліпсів, де числа, пов’язані з еліпсами, вказують порядок кластеризації. Рисунок 1.12 (b) показує ту саму інформацію, але у вигляді дендрограми. Висота, на якій два кластери об’єднані в дендрограмі, відображає відстань двох кластерів. Наприклад, з рисунку 1.13 ми бачимо, що

відстань між точками 3 і 6 дорівнює 0,11, і це висота (крок), на якій вони об’єднані в одне скупчення у дендрограмі.

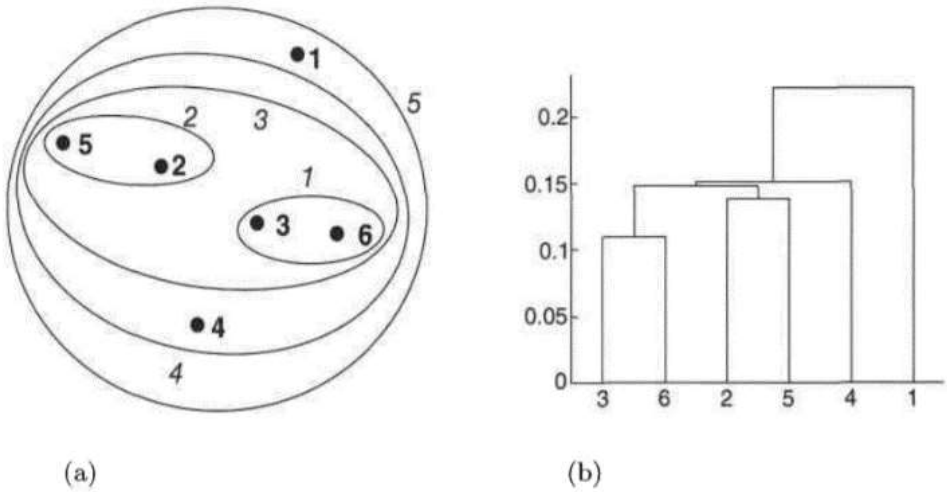

Рисунок 1.12 – Результат виконання підходу з одинарним зв’язком

Для повного зв’язку або версії MAX ієрархічного кластерингу близькість двох кластерів визначається як максимальна відстань (мінімум подібності) між будь-якими двома точками в двох різних кластерах. Повне посилання менш сприйнятливе до шуму та викидів, але воно може розбивати великі скупчення і сприяє утворенню кулястих форм. Рисунок 1.13 показує результати застосування методу MAX до набору даних із шести одинарних кластерів.

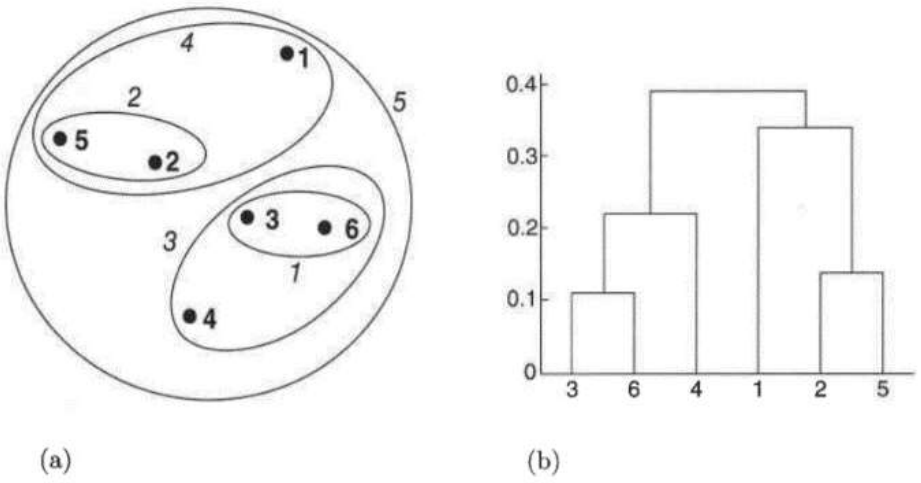

Рисунок 1.13 – Результат виконання підходу з повним зв’язком

Для середньо-групової кластеризації близькість двох кластерів визначається як середня парна близькість серед усіх пар точок у різних кластерах. Це проміжний підхід між підходами з єдиним та повним зв’язком. Рисунок 1.14 показує результати застосування середньо-групового підходу до вибіркової сукупності даних із шести балів. Для ілюстрації того, як працює усереднення по групі, ми обчислюємо відстань між деякими кластерами.

Для методу Уорда близькість між двома кластерами визначається як збільшення похибки в квадраті, що виникає при об'єднанні двох кластерів. Таким чином, цей метод використовує ту саму цільову функцію, що і кластеризація K-засобів. Хоча може здатися, що ця особливість робить метод Уорда дещо відмінним від інших ієрархічних прийомів, можна математично показати, що метод Уорда дуже схожий на метод середнього по групі, коли близькість між двома точками приймається як квадрат відстані між ними [6].

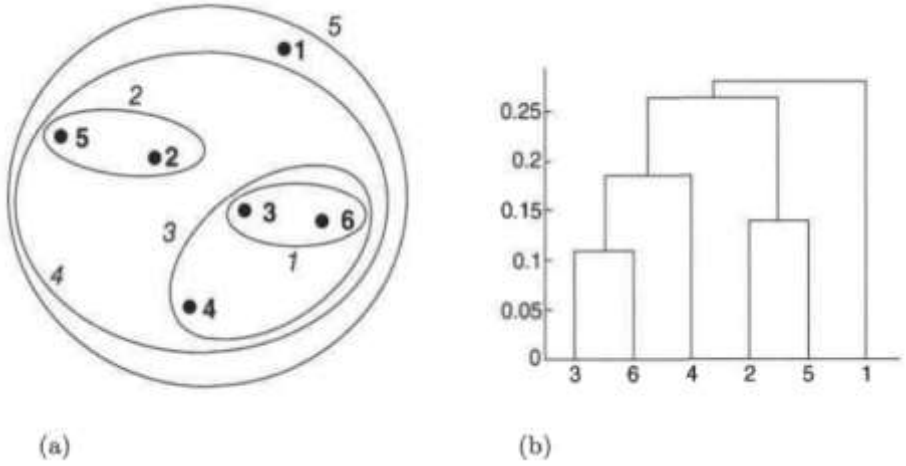

Рисунок 1.14 – Результат виконання проміжного підходу

На рисунку 1.15 показуються результати застосування методу Уорда до вибіркового набору даних із шести балів. Кластеризація, яка створюється, відрізняється від кластеризації, створеної за одним посиланням, повним посиланням та середньо-груповим.

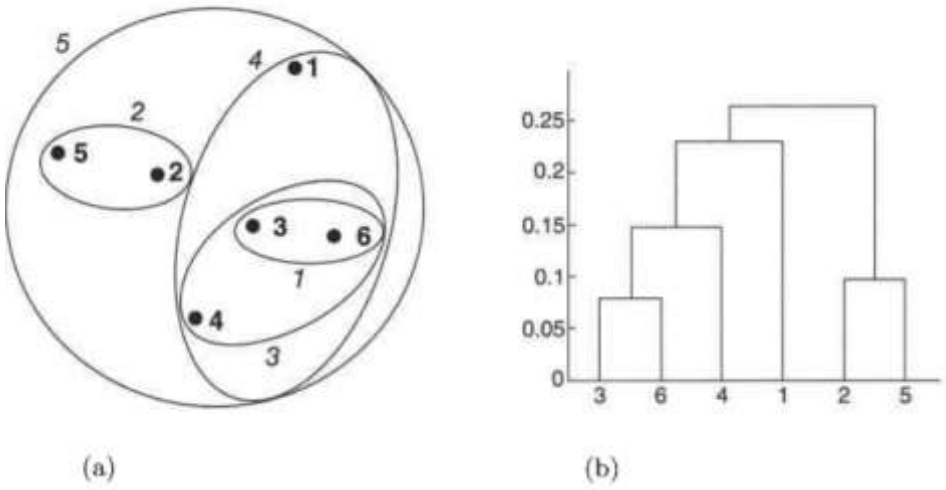

Рисунок 1.15 – Результат виконання проміжного підходу

Центроїдний метод обчислює близькість між двома скупченнями, обчислюючи відстань між центроїдами скупчень. Окрім цього, даний метод має характеристику, яку часто вважають негативною, даною характеристикою не володіють інші ієрархічні методи кластеризації: можливість інверсій. Зокрема, два кластери, які об'єднані, можуть бути більш подібними (менш віддаленими), ніж пара кластерів, які були об'єднані на попередньому кроці [7]. Для інших методів відстань між об'єднаними кластерами монотонно збільшується (або, в гіршому випадку, не збільшується), оскільки відбувається перехід від одиночних кластерів до одного всеохоплюючого кластера.

1.10 Формула Ленса-Вільямса для близькості кластерів

Будь-яку близькість кластера, можна розглядати як вибір різних параметрів для близькості між кластерами Q і R, де R утворюється шляхом злиття кластери A і B. У цьому рівнянні p (.,.) є функцією близькості, тоді як m_a, m_b і m_Q- кількість точок у кластерах A, B та Q відповідно. Іншими словами, після об'єднання кластерів A і B, щоб сформувати кластер R, близькість нового

кластеру R до існуючого кластера Q є лінійною функцією близькості Q відносно вихідних кластерів А та Б.

$$p(R,Q) = \alpha_A p(A,Q) + \alpha_B p(B,Q) + \beta p(A,B) + \gamma|p(A,Q) - p(B,Q) \quad (1.1)$$

Будь-яка ієрархічна техніка кластеризації, яка може бути виражена за допомогою формули Ланса-Війліама, не потребує збереження вихідних точок даних. Натомість близькості оновлюється у міру того, як відбувається кластеризація [8].

1.11 Ключові проблеми ієрархічної кластеризації

Раніше згадувалось, що агломератна ієрархічна кластеризація не може розглядатися як глобальна оптимізація цільової функції. Натомість агломератні ієрархічні методи кластеризації використовують різні критерії для місцевого вирішення на кожному кроці, які кластери слід об’єднати (або розділити для підходів, що розділяють). Таким чином алгоритми кластеризації дозволяють уникнути труднощів спроби вирішити складну комбінаторну задачу оптимізації. Крім того, такі підходи не мають проблем з локальними мінімумами або труднощів у виборі початкових точок. Звичайно, часова складність O($m^2 \log m$) і просторова складність O(m^2) у багатьох випадках непосильні [9].

Однією із проблем агломератної ієрархічної кластеризації, є спосіб обробки відносних розмірів пар кластерів. Дана проблема стосується лише схем близькості кластера, які включають суми, такі як центроїд, Уорда та середнє значення групи. Існує два підходи: зважений, який обробляє всі кластери однаково, і незважений, що враховує кількість точок у кожному кластері рахунок. Термінологія зважених або незважених відноситься до точок даних, а не до кластерів. Іншими словами, обробка кластерів неоднакового розміру

однаково дає різну вагу точкам у різних кластерах, тоді як врахування розміру кластерів дає точкам у різних кластерах однакову вагу.

Агломератні ієрархічні алгоритми кластеризації, як правило, приймають правильні локальні рішення щодо поєднання двох кластерів, оскільки вони можуть використовувати інформацію про попарну подібність усіх точок. Однак, як тільки буде прийнято рішення про об'єднання двох кластерів, це не можна скасувати пізніше. Цей підхід заважає локальному критерію оптимізації стати загальним критерієм оптимізації.

Хоча критерій "мінімізувати квадратну помилку" з K-засобів використовується при вирішенні того, які кластери об'єднати в методі Уорда, кластери на кожному рівні не представляють локальних мінімумів щодо загальної SSE. Кластери не є стабільні, в тому сенсі, що точка в одному кластері може бути ближче до центроїда якогось іншого кластера, ніж до центроїда його поточного кластера. Тим не менше, метод Уорда часто використовується як надійний метод ініціалізації кластеризації K-засобів, вказуючи, що локальна цільова функція "мінімізувати квадратну помилку" дійсно має зв'язок із глобальною цільовою функцією "мінімізувати квадратну помилку".

Є кілька методів, які намагаються подолати обмеження, остаточного рішення злиття. Один із підходів намагається виправити ієрархічну кластеризацію, рухаючи гілки дерева навколо, щоб поліпшити глобальну цільову функцію. Інший підхід використовує метод кластеризації розділів, такий як K-means, для створення багатьох малих кластерів, а потім виконує ієрархічну кластеризацію, використовуючи ці малі кластери як вихідну точку.

1.12 Висновки

Отже, агломератні ієрархічних алгоритмів кластеризації мають власні сильні та слабкі сторони. Дані алгоритми використовуються в вузьких сферах наприклад, створення таксономії, оскільки вона вимагає створення ієрархії

кластерів. Також було проведено кілька досліджень, які свідчать про те, що ці алгоритми можуть створювати якісніші кластери. Однак агломератні ієрархічні алгоритми кластеризації є дорогими для обчислень та зберігання. Той факт, що всі злиття остаточні, також може спричинити проблеми для зашумлених великих даних. Основним способом вирішення даних проблем є часткова кластеризація даних за допомогою методу k-середніх.

2 ОГЛЯД ВІДОМИХ СИСТЕМ КЛАСТЕРИЗАЦІЇ КОРИСТУВАЧІВ

Кластеризація широко використовується при обробці даних в:

- системах таргетингу;
- стрімінгових сервісах;
- соціальних мережах знайомств;
- чат-платформах.

Дані системи використовують кластеризацію для того аби результати їх пропозицій були цікавими для користувача.

2.1 Кластеризація користувачів в системах таргетингу

Таргетована реклама є одним з способів онлайн-реклами, в якому використовуються методи і налаштування пошуку цільової аудиторії відповідно до заданих характеристиками і інтересами людей, які можуть цікавитися рекламованим товаром або послугою. Така реклама показується лише для обраної (цільової) аудиторії, що дозволяє більш ефективно використовувати рекламний бюджет компанії.

2.1.1 Google ads

Google Ads - це онлайн-рекламна платформа, розроблена Google, де рекламодавці пропонують відображати короткі рекламні оголошення, пропозиції послуг, списки продуктів або відео для веб-користувачів. Він може розміщувати рекламу як у результатах пошукових систем, таких як Пошук Google, мобільних додатках та YoutTube [10].

Система Google Ads частково базується на файлах cookie, а частково на ключових словах, визначених рекламодавцями. Google використовує ці характеристики для розміщення рекламної копії на сторінках, де вони вважають, що це може бути доречним. Рекламодавці платять, коли користувачі

спрямовують свій веб-перегляд, натискаючи на рекламне оголошення. Реклама може бути реалізована на місцевому, національному та міжнародному рівнях.

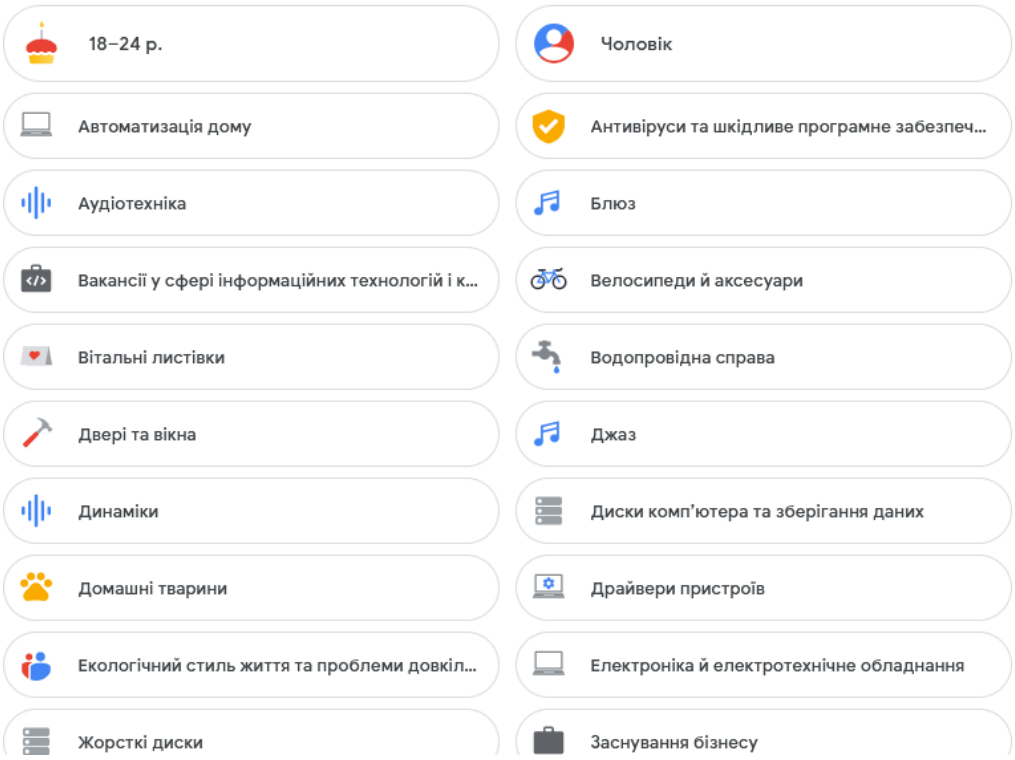

Рисунок 2.1 – Ключові чинники, що були визначенні для профілю користувача

Опираючись на список інтересів користувача, google ads має змогу показувати персоналізовану рекламу (рис 2.1) та зводити споживача та надавача послуг разом використовуючи вбудовані рекламні блоки.

Основною проблемою даного підходу є погана обробка сирих даних, в результаті котрих в список інтересів можуть потрапити діаметрально протилежні чинники що в дійсності мають накладене табу.

Плюсом даної системи для рекламодавців та користувачів є можливість ручного налаштування списку інтересів, що дозволяє ліпше використовувати рекламний бюджет зі сторони постачальника послуг, та отримувати лише корисні рекламні оголошення зі сторони користувача.

2.1.2 Facebook Business Manager

Business Manager дозволяє рекламодавцям керувати своїми маркетинговими ресурсами в одному місці і надавати доступ до них своїй команді, партнерам і постачальникам [11] (рис 2.2).

Даний інструмент дозволяє виконувати такі речі як:

- Створювати та керувати об'єктами, такими як Сторінка Facebook, аккаунт Instagram, список аудиторій або каталог продуктів, в одному місці.
- Керувати доступом і дозволами для всіх, хто працює з рекламними акаунтами, Сторінками і додатками.
- Відстежувати результати реклами на Facebook і в Instagram за допомогою докладних даних по витратах і показам.

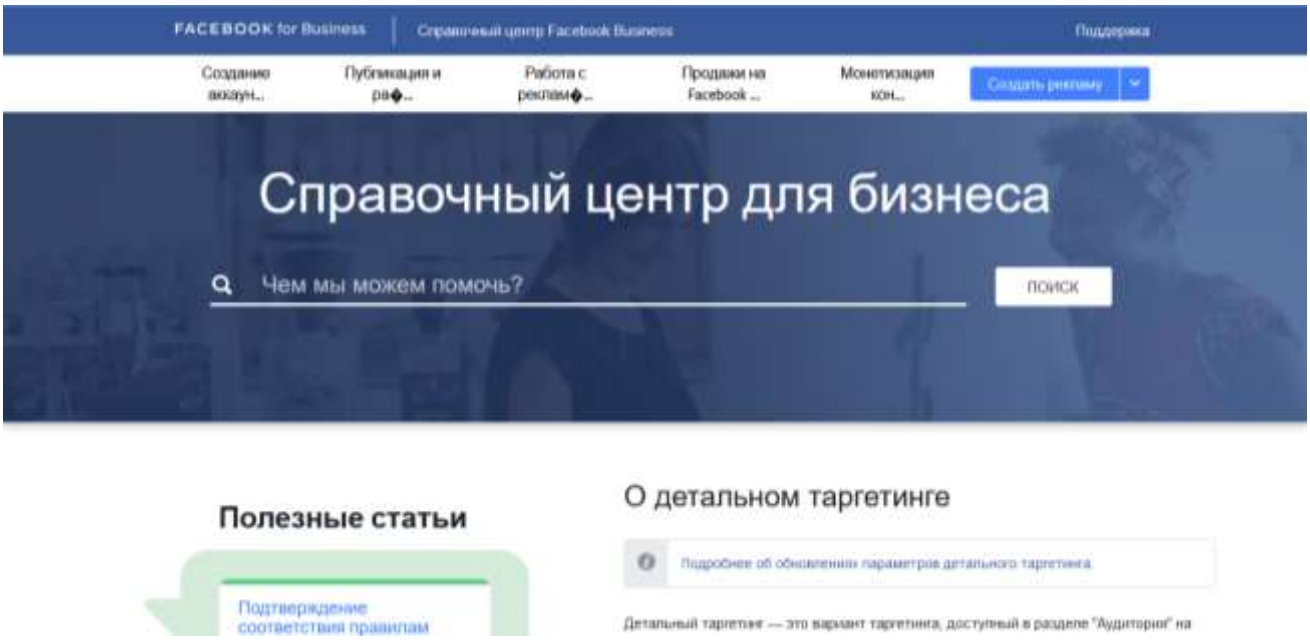

Рисунок 2.2 – Головна сторінка Business Manager

Основною відмінністю від google ads є ліпша аналітика і значно краща робота в команді, проте таргетування в екосистемі Facebook є дещо грішим, оскільки користувач не може на пряму впливати на список інтересів.

2.2 Кластеризація користувачів в стрімінгових сервісах

Рекомендаційні системи в стрімінгових сервісах не працюють на пряму з кластеризацією, проте вона є одним з кроків для побудови колаборативної фільтрації. В даних системах на основі оцінках групи відомих користувачів для прогнозування невідомих оцінок іншого користувача

2.2.1 YouTube

YouTube - відеохостинг, що надає користувачам послуги зберігання, доставки та показу відео. YouTube став одним із найпопулярніших місць для розміщення відеофайлів і другим сайтом у світі за кількістю відвідувачів.

Користувачі можуть завантажувати, переглядати, оцінювати, коментувати, відправляти повідомлення і ділитися тими чи іншими відеозаписами. У січні 2012 року щоденна кількість переглядів відео на сайті досягло 4 млрд. На сайті представлені фільми, музичні кліпи, трейлери, новини, освітні передачі, а також любительські відеозаписи, включаючи відеоблог, слайд-шоу, гумористичні відеоролики та інше [12].

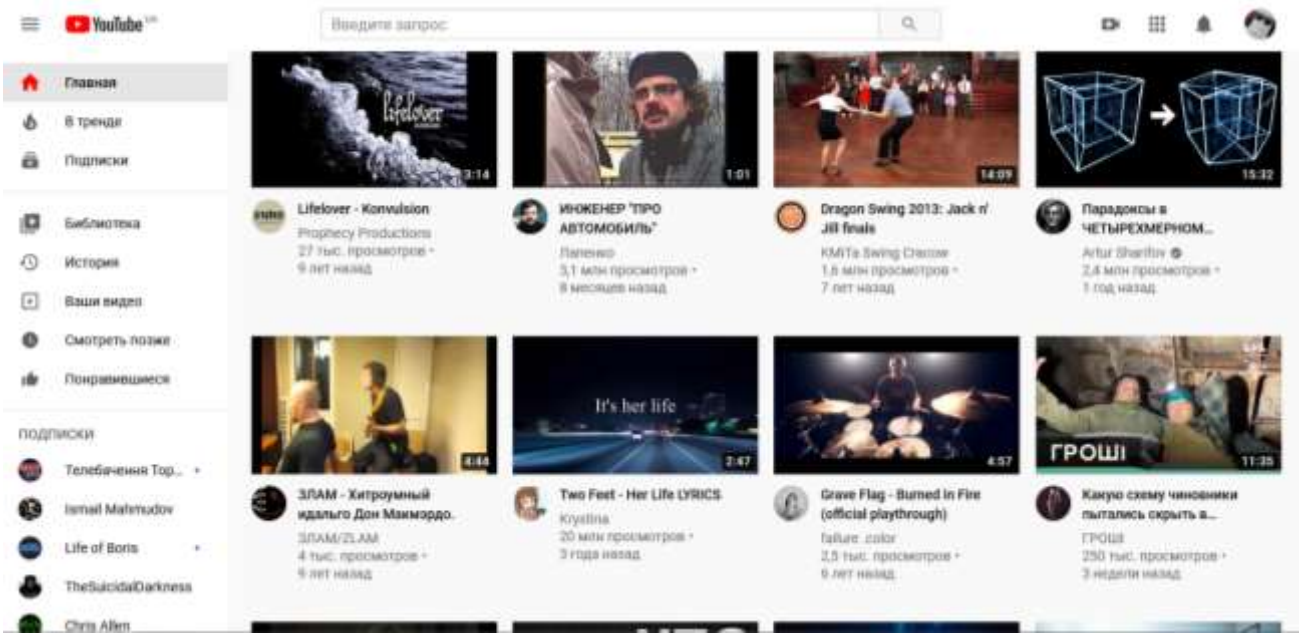

Рисунок 2.3 – Рекомендовані ролики

Розробники з Google опублікували опис end-to-end рекомендаційної системи в YouTube. Система використовує кілька цільових функцій для ранжирування і враховує особисті переваги користувача. Щоб оптимізувати модель на кілька цільових функцій розробники використовували Multi-gate Mixture-of-Experts. За результатами експериментів на реальних користувачів, оновлена модель працює краще попередніх версій.

У своїй роботі дослідники представили великомасштабну систему для ранжирування відео. Щоб оптимізувати відразу кілька цільових функцій, використовується архітектура Multi-gate Mixture-of-Experts. Щоб позбутися від зсуву позицій при ранжируванні, дослідники застосовують Wide & Deep архітектуру моделі.

При дизайні і реалізації рекомендаційної системи є ряд проблем:

- цільові метрики, які потрібно оптимізувати, можуть не збігатися і конфліктувати один з одним;
- у призначених для користувача переглядах закладено зміщення. Наприклад, користувач дивиться перший рекомендував би відео не тому що це те, що він найбільше хоче подивитися, а тому що воно є першим в списку. Цей зсув необхідно обходити, щоб не створювати ефект feedback loop.

Отже, система рекомендацій youtube працює швидше ніж колобаративна фільтрація на основі кластеризації користувачів, проте дана система є не точною, та містить в собі аномалії. Дана система чудово задовольняє потреби користувачів, а аномалії (рис 2.3) дозволяють відкривати для себе нові області людської буденності.

2.2.2 Deezer

Deezer - французький інтернет-сервіс потокової передачі музики. Дозволяє прослуховувати музичні композиції різних лейблів звукозапису, включаючи Universal Music Group, Sony Music та Warner Music Group (належить материнській компанії Deezer Access Industries), на різних пристроях в режимі

онлайн або офлайн. В даний час в базі сервісу знаходяться більш 56 мільйонів музичних композицій і 34 000 радіостанцій [13].

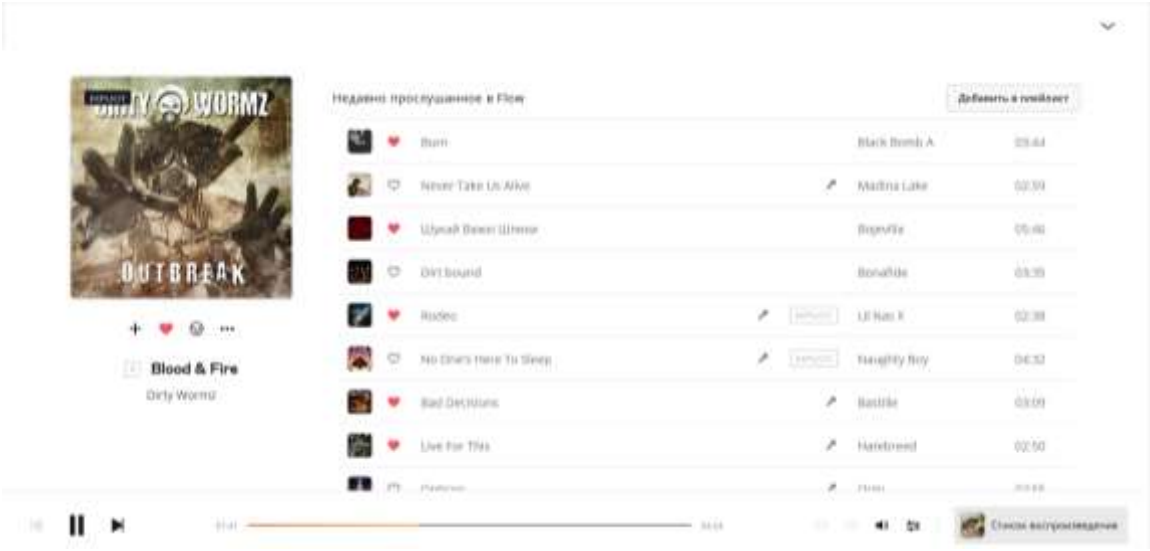

Рисунок 2.4 – Deezer flow

Центральне місце в deezer займає компонент з назвою flow (рис 2.4) що на основі попередніх вподобань створює унікальний плейлист. Сервіс розцінює flow як важливий елемент що дозволяє розширювати кругозір в межах улюблених та суміжних жанрах.

Компанія не розголошує алгоритм роботи власної рекомендаційної системи, проте опираючись на glassdoor можна припустити що використовується модифікований метод кластеризації для колобаративної фільтрації.

Як правило flow надзвичайно точно розуміє смаки користувачів та позбавлений від видачі аномальних рекомендації.

2.3 Кластеризація користувачів в соціальних мережах знайомств

Соціальні мережі формуються за інтересами, потребами, ресурсfv і сфер впливу, соціальним статусам і позиціях.

Розрізняють такі їх види, як політичні, економічні, комерційні, фінансові, культурні, дозвільні, мережі спілкування.

Формування соціальних мереж громадянської дії починається з невеликих спільнот. Особиста довіра між добре знайомими один одному людьми може служити природним початком формування таких мереж. Взаємини з іншими спільнотами та агентами зав'язуються шляхом перекидання "містків" до державних структур, політичними організаціям, фінансовими інститутам, промисловими асоціаціями, профспілками, пресою, релігійними організаціями та іншими групами громадян, що створюють умови для регулярних контактів, встановлення довіри, взаємовигідній дискусії і взаємовпливу.

2.3.1 Tinder

Tinder - це американська програма геосоціальних мереж і знайомств, яка дозволяє користувачам шукати нові знайомства, базуючись на фотографіях профілів, невеликій біографії та спільних інтересах. Як тільки двоє користувачів "збігаються", вони можуть обмінюватися повідомленнями.

Tinder був запущений в 2012 році в стартовому інкубаторі Hatch Labs як спільне підприємство між IAC та фірмою з розробки мобільних додатків Xtreme Labs. До 2014 року Tinder реєстрував близько одного мільярда «користувачів» на день [14].

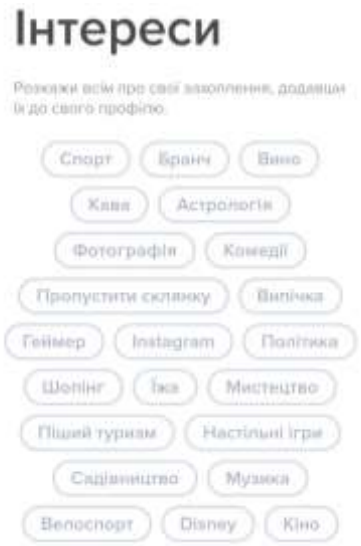

Рисунок 2.5 – Вибір спектру інтересів в соціальній мережі

Для уподібнення користувачів використовується елементи кластеризації та класифікації користувачів. Основним недоліком є відносно невелика кількість (рисунок 2.5) визначених маркерів що не покриває межі інтересів середньостатистичної людини та відсутність можливості ручного вводу.

2.3.2 Badoo

Badoo - це соціальна мережа, орієнтована на знайомства, заснована російським підприємцем Андрієм Андрєєвим у 2006 році. Штаб-квартира знаходиться в Лімассолі, Кіпр та Лондоні, Великобританія , а офіси розташовані на Мальті, в Росії та США. Він працює в 190 країнах і доступний 47 різними мовами, що робить його найбільш широко використовуваною мережею знайомств. Додаток доступний на iOS, Android та в Інтернеті. Badoo працює за моделлю freemium, завдяки якій основні послуги можна використовувати без оплати [15].

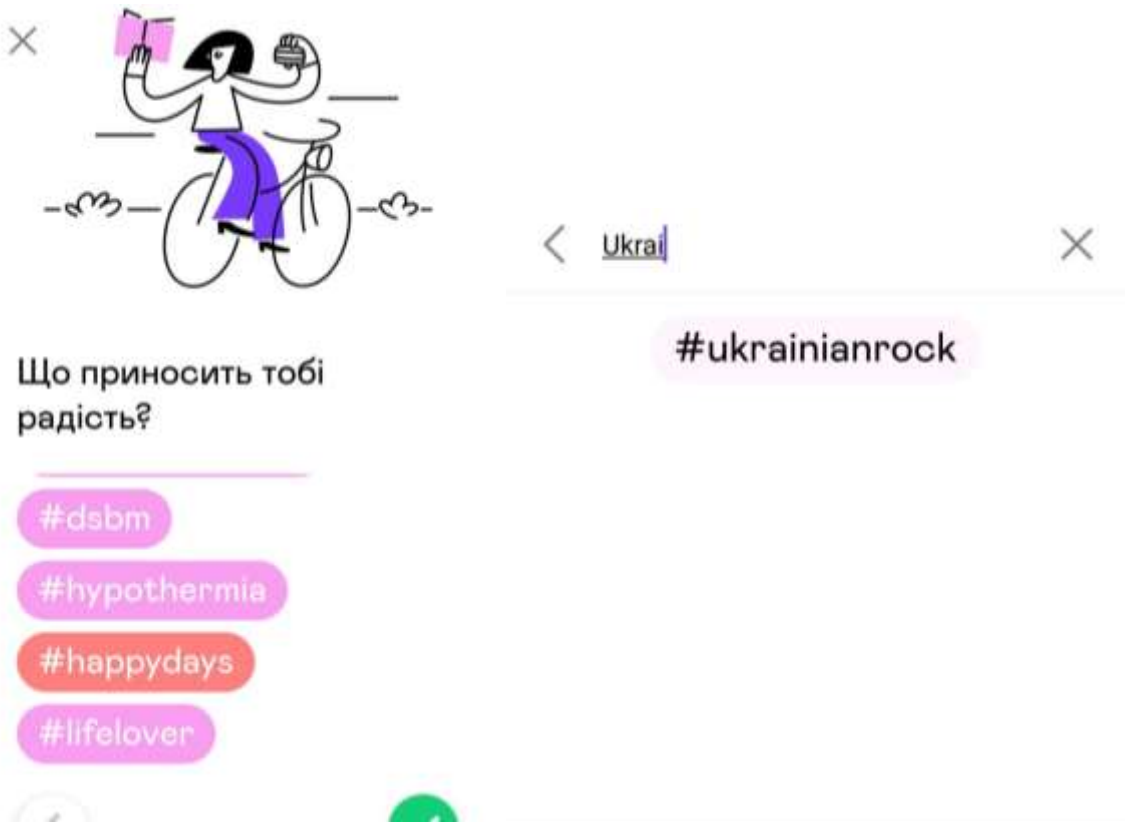

Рисунок 2.6 - Екран вибору спектру інтересів

Функція кластеризації на відмінну від tinder може приймати довільну кількість параметрів (рисунок 2.6) та надає значно більшу кількість варіантів вибору для формованого списку інтересів.

2.4 Кластеризація в системах групових чатів

Кімната чату, в основному може являти собою як синхронні і в окремих випадках асинхронні конференцій. Таким чином, цей термін може означати будь-яку технологію, починаючи від онлайн-чату в режимі реального часу та взаємодії в режимі онлайн з незнайомцями до повністю захоплюючого графічного соціального середовища.

Основне використання кімнати чату - обмін інформацією за допомогою тексту з групою інших користувачів. Можливість спілкуватися з кількома людьми в одній розмові відрізняє чати від програм обміну миттєвими повідомленнями, які, як правило, призначені для спілкування один на один.

2.4.1 ЧатПростоТак

ЧатПростоТак – являє собою поліпшений груповий чат де пошук співрозмовника здійснюються за допомогою створюваних тегів-маркерів, що дає зрозуміти кругозір та напям діалогу чи бесіди [16].

Даний сервіс дозволяє:

- створювати тематичні чати;
- знаходити співрозмовників по власним інтересам;
- додавати та приймати запрошення гостям та користувачам сату.

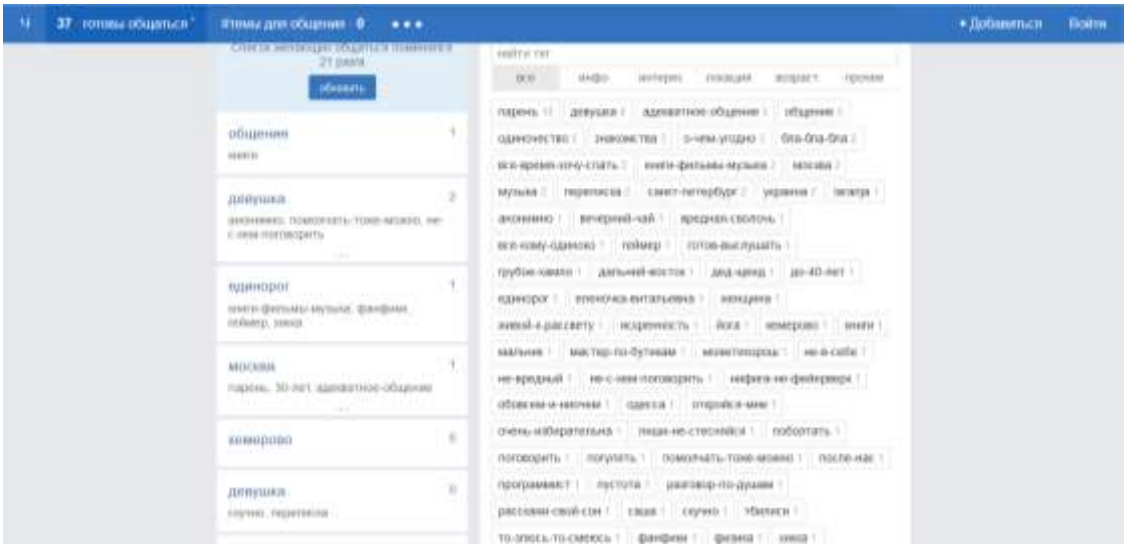

Рисунок 2.7 – Меню додавання та вибору тегів кругозору активних користувачів

Даний сервіс не має обмежень на створювані теги, характер та їх кількість на одну бесіду. Проте має один суттєвий недолік, він не проводить кластеризацію користувачів. Насправді даний сервіс надає можливість вибору серед тегів (рисунок 2.7) та пошук по існуючому пулу онлайн-користувачів, тобто є синхронним.

2.4.2 Amino

Amino являє собою мобільний додаток для конвенційних вузьких груп. Це дозволяє легше формувати групи близьких по інтересам людей.

Головна особливість Amino - спільноти, присвячені певній темі, до якої можуть приєднатися користувачі. Користувачі також можуть спілкуватися з іншими членами спільноти трьома способами: в текстовому, голосовому чаті, або-ж в кінозалі - в чаті, який дозволяє користувачам разом переглядати відео під час голосового чату [17]. Інші функції включають опитування, блоги, записи із зображеннями, записи вики, історії та вікторини. У деяких випадках пости, які зроблені дуже добре і були помічені адміністрацією спільноти, в кінцевому

підсумку отримають функцію, завдяки якій вона з'являється на головній сторінці спільноти.

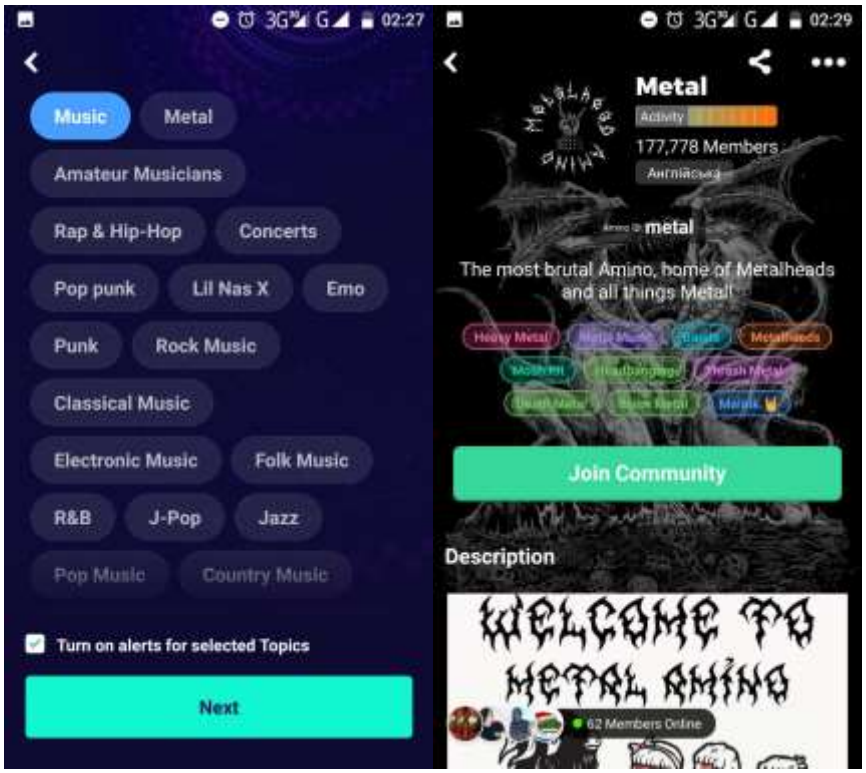

Рисунок 2.8 – Екран вибору спектру інтересів в додатку Amino

На рисунку 2.8 представлено список вибору інтересів та одну з конвенційних спільнот на основі складених інтересів.

В даному випадку основне завдання є співвіднесення користувача до наявних груп а не створення нових, тобто головне завдання є не кластеризація а класифікація користувачів.

2.5 Висновки

В даному розділі було розглянуто відоме програмне забезпечення, що реалізовує кластеризацію, класифікацію чи простий розподіл користувачів на окремі підгрупи чи кластери. Проведено аналіз основного завдання що вирішує

існуюче програмне забезпечення та класифіковано вирішувані проблем. Варто виділити наступні слабкі сторони в існуючих системах, а саме:

- при ієрархічній кластеризації, велика кількість використовуваної пам’яті і велика алгоритмічна складність;
- при методу k-середніх є проблема з точністю розподілення даних на краях кластеру, центр кластеру задається програмно;
- в певних продуктах кластеризація користувачів не реалізована в повній мірі, а саме відсутність групування користувачів за інтересами, вхідні дані слугують лише для тегування облікових записів;
- вхідні дані для системи кластеризації обмежені програмно, що не дозволяє користувачу представити свій кругозір в повній мірі.

Окрім цього, в певних програмних продуктах збір даних йде автоматично та користувач не може вплинути на результати видачі рекомендаційної системи. Результатом цього є зниження цікавості користувача до рекомендаційної системи.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ КЛАСТЕРИЗАЦІЇ КОРИСТУВАЧІВ

3.1 Основні вимоги до програмного забезпечення

Важливим кроком в створенні програмного забезпечення є чітке встановлення вимог що дозволить зрозуміти основні компоненти проектованої системи. Одним з таких принципів є “YAGNI” що в буквальному перекладі означає – “вам це не знадобиться”. Даний принцип застосовується на етапі поставлення завдання і тестуванні прототипів програмних продуктів. Мета даного принципу декларується як відмова від доданої функціональності, в котрій немає безпосередньої потреби.

Окрім цього важливим елементом при побудові програмного забезпечення є використання об’єктно-орієнтованого підходу до програмування. Даний підхід є доволі складним на початку проектування системи відносно функціонального підходу, проте при розширені програмного пакету логіки та вартості супроводу що вимірюється в затраченому часі розробника. Отже, об’єктно-орієнтований підхід виграє при розробці великих масштабованих проектів та є легшим в подальшому супроводі.

Отже, до створюваного програмного забезпечення висунуті такі вимоги.

- виконання агломератної ієрархічної кластеризації використовуючи методи паралельного програмування;
- попередня обробка вхідних даних;
- побудова матриці відстаней за допомогою мультипотокового програмування;
- попарне розбиття універсальної множини на ієрархічні кластери ;
- об’єднання вхідних даних в спорідненні кластери;
- збереження результуючого файлу в документо-орієнтовану базу даних;
- оцінка кластерної еквівалентності.

Окрім функціональних вимог слід висунути певні не функціональні вимоги а саме:

- даний додаток повинен бути кросплатформний;
- обмеження на платформу накладається лише в наслідок;
- покриття функціональними та інтеграційними тестами;
- легке додавання нових методів обробки графу та кластеризації;
- використання патернів багатопотокового програмування;
- дотримання solid та основних oop принципів.

3.2 Опис обраних засобів для розробки програмного забезпечення

Програмне забезпечення для розподіленої ієрархічної кластеризації було розроблено на мові програмування java використовуючи пакет (бібліотеку) concurrency. Основними використовуваними елементами для побудови програмного забезпечення є такі речі як:

- мова програмування – java;
- засіб автоматизації роботи - Maven;
- бібліотека concurrent;
- JDBC драйвер для підключення до бази даних;
- документо-орієнтована база даних MongoDB;
- система контролю версій git;
- середовище розробки intelij idea.

3.2.1 Мова програмування Java

Java – являє собою об’єктно орієнтовану мову програмування, що була вперше презентована в 1995 році компанією “Sun Microsystems”. Основною задачею створення даної мови, це потреба в кросплатформності, аби її можна було використовувати в вбудованих системах, таких як мобільні телефони, аплети чи пристрої керування [18].

Розвитком популярності даної мови є розвиток інтернету та веб-переглядачів що дозволяли завантажувати та запускати програмну логіку не на стороні сервера а на стороні клієнта. Також аплети були одним з перших інструментів що дозволяли робити веб-сторінки інтерактивними.

На даний час мова програмування Java здебільшого використовується при розробці мобільних додатків та побудові клієнт-серверних рішень. Особливої популярності вона набула в банківській сфері.

Основними плюсами мови є:

- кросплатформеність;
- об’єктно-орієнтований підхід до розробки;
- філософія “написане одного разу буде працювати завжди”;
- велика кількість користувачів;
- потужність та ефективність мультипотокових застосунків;
- велика кількість вільних бібліотек .

3.2.2 Використовуванні бібліотеки

Для розробки додатку слід використати прикладний програмний інтерфейс та JDBC з допомогою котрого програмне забезпечення на java здійснює доступ до бази даних. Даний платформо-незалежний інтерфейс є стандартним способом взаємодії java-додатків і різноманітних системи управління базами даних (рисунок 3.1). JDBC-конектор реалізований в вигляді пакету java.sql та входить в стандартну бібліотеку java.

Для тестування використовується бібліотека модульного тестування JUnit та Mockito. Для позначення тестів, а саме їх запуск та особливостей виконання тестування використовують анотації що являють собою метаданими. Даний елемент є доволі потужним рушієм що полегшує читання коду людиною та застерігає від помилок маркуючи класи. Другим важливим елементом в тестуванні є Mockito, при правильній реалізації інверсії контролю, даний елемент дозволяє замінити елементи системи такими елементами як mocks та stabs.

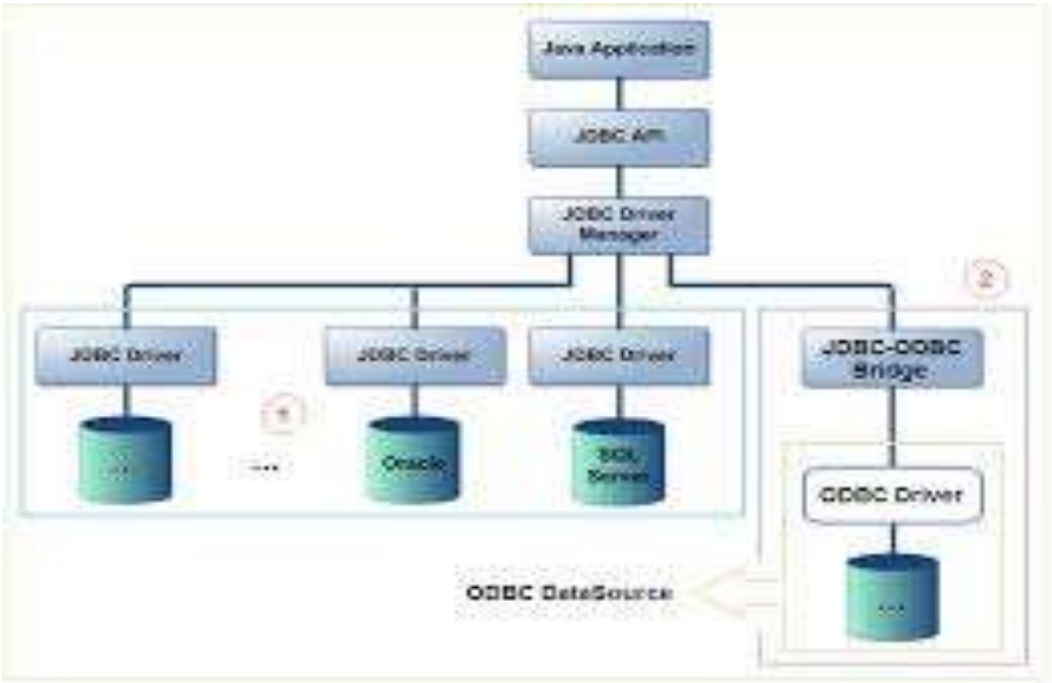

Рисунок 3.1 – Взаємодія java-додатку з базою даних з допомогою jdbc

В сучасних сервісах широко використовується логування, що дозволяє зрозуміти на якому етапі виникла помилка виконання головного потоку. Окрім цього логування використовується не тільки для пошуку помилок в програмі, здебільшого воно носить інформативний характер, тому в log4j виділяють 6 рівнів логування при виконанні програми [19].

Основним елементом даної магістерської роботи є спроба пришвидшити виконання ієрархічної кластеризації використовуючи такий елемент як багатопотоковість. В мові Java як і в інших мовах програмування використання потоків є доволі складним процесом, проте виграш від розпаралелення додатку при великій кількості складних обчислень є доволі високий.

3.2.3 Інтегроване середовище розробки Intelij idea

IntelliJ IDEA - інтегроване середовище розробки мови Java, призначена для створення кроссплатформенних додатків, здатна так само використовувати інші мови програмування (для цього потрібно додати необхідний комплекс програмного забезпечення).

Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи їм сконцентруватися на розробці функціональності додатків, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

Можливості IntelliJ IDEA:

- розумне автодоповнення, інструменти для аналізу якості коду, зручна навігація, розширені рефакторингом і форматування для Java, Groovy, Scala, Clojure і Erlang;
- професійний набір інструментів для розробки Android-додатків;
- підтримка JavaFX 2.0 і інтеграція з SceneBuilder;
- дизайнер інтерфейсу для Swing;
- інтеграція з автоматизованими інструментами побудови та управління проектом, включаючи Maven, Gradle, Ant та іншими;
- інструменти для тестування додатків з підтримкою JUnit, TestNG, Spock, ScalaTest і spec2;
- інтеграція з системами управління версіями, включаючи Git, Subversion, Mercurial і CSV.

3.2.4 Система контролю версій git

Git - розподілена система керування версіями, метою створення якої було управління процесом розробки ядра Linux. Git - це гнучка, розподілена система контролю версій, яка дає багато можливостей розробникам програмних продуктів, для яких важливо управління історією змін [20].

Git дозволяє обмінюватися даними резервування з іншими користувачами в репозиторії. У Git є центральний репозиторій, з яким синхронізуються кілька розробників, кожному з яких надається локальна копія всієї історії розробки.

Система керування версіями забезпечує швидке поділ і злиття версій, робота над версією може йти паралельно в декількох гілках, які потім досить легко можуть частково або повністю об'єднатися, знищитися, відкотитися або розростися в нові гілки проекту.

3.2.5 Система автоматичної збірки maven

Maven-це інструмент для збірки проекту: компіляції, створення jar і створення дистрибутива програми. Інструмент збірки проекту -система збірки, для збору інших програм. На вхід система збирання отримує вихідний код, а на вихід видає програму, яку вже можна запустити [21].

Основні переваги Maven:

- незалежність від ОС. Збірка проекту відбувається в будь-якій операційній системі. Файл проекту один і той же;
- управління залежностями. Рідко які проекти пишуться без використання сторонніх бібліотек (залежностей). Ці сторонні бібліотеки часто теж в свою чергу використовують бібліотеки різних версій. Maven дозволяє керувати такими складними залежностями. Що дозволяє вирішувати конфлікти версій і в разі потреби легко переходити на нові версії бібліотек;
- можлива збірка з командного рядка. Таке часто необхідно для автоматичного складання проекту на сервері (Continuous Integration);
- інтеграція з середовищами розробки. Основні середовища розробки на java легко відкривають проекти які збираються з допомогою maven. При цьому найчастіше проект налаштовувати не потрібно він відразу готовий до подальшої розробки;
- як наслідок - якщо з проектом працюють в різних середовищах розробки, то maven зручний спосіб зберігання налаштувань. Її конфігураційний файл середовища розробки і для збірки один і той же-менше дублювання даних і відповідно помилок.

Maven - є одним з найпопулярніших складальників для Java, він так само допомагає вирішувати складні завдання з управління проектом.

Основним завданням Maven є робота над обробкою проекту ділиться на певний набір етапів - в Maven це називається «фаза». Під кожну фазу (редагування, компіляція, збір) існує вже готова програма - плагін (plugin). Кожен плагін має низку налаштувань, які можна змінювати під свої потреби.

3.3. Реалізація додатку вибраними способами

В об’єктно-орієнтованому підході на відміну від функціонального головне місце посідає спосіб представлення даних, та лише на другому місці методи їх обробки.

3.3.1 Вибір структури для n-вимірного вектору

Першочергово необхідно визначитись з моделлю представлення n-вимірних векторів. Оскільки вектор представляє собою список інтересів що представлений в вигляді стрічок. Де кожна стрічка унікальна та не дублюється в середині вектору, також порядок входження в вектор не грає жодної ролі. Таким чином n-вимірний вектор можна сміливо представити в вигляді множини java [22].

В стандартному пакеті set є насправді інтерфейсом що отримує посилання на створений об’єкт одного з трьох реалізованих в бібліотеці класів HashSet, TreeSet, LinkedHashSet або класів нащадків що реалізував інженер програмного забезпечення, наслідуючи скелетну реалізацію AbstractSet чи одного з трьох вище наведених класів [30].

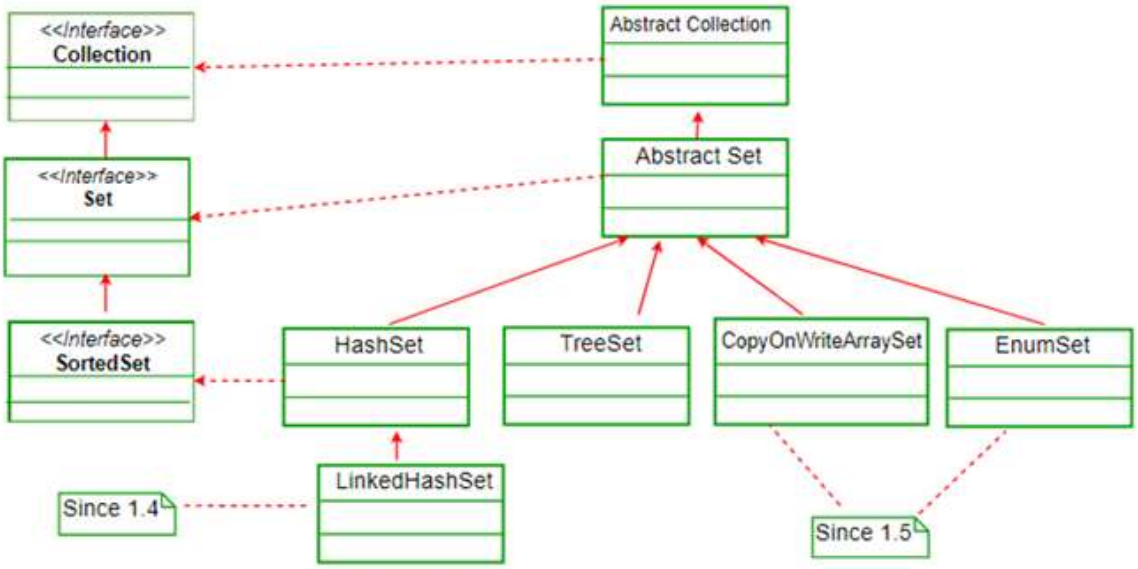

Рисунок 3.2 – Ієрархія та взаємодія класів в пакеті collection framework

Вибираючи між трьома реалізаціями можна замітити що LinkedHashSet та TreeSet мають надлишкову для проектованого модуля логіку роботи, та дещо гіршу продуктивність ніж HashSet. Тому варто зупинити свій вибір на ньому.

3.3.2 Вибір структури ієрархічної кластеризації

Оскільки результат ієрархічної кластеризації є ієрархічне дерево, що було створено попарним об’єднанням вузлів, найменший елемент можна розглядати як одноелементний, однонаправлений вузол [23].

Таким чином даний вузол можна представити в вигляді uml класу.

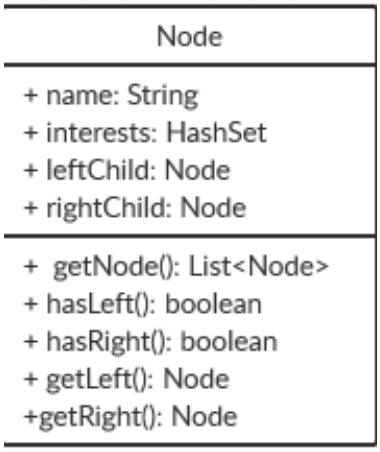

Рисунок 3.3 – Клас Node

На основі даного класу можливо будувати складні ієрархічні дерева використовуючи власну логіку. Також варто замітити що метод getNodes є рекурсивним що дозволяє обійти все дерево та отримати згортку в вигляді узагальненого list’а з типом елементів Node. Даний рекурсивний метод спрощує роботу з структурами даних що мають невідомий розмір використовуючи раніше задані визначення та циклічні посилання.

Лістинг 3.1 – Програмний код пакету

```
package model.node;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Set;

public class Node {
    String name;
    Set<String> intersts;

    public Node leftChild;
    public Node rightChild;

    public Node (String name, Node leftChild,Node rightChild,
Set<String> intersts){
        this.name= name;
        this.leftChild=leftChild;
        this.rightChild= rightChild;
        this.intersts= intersts;
        if (intersts == leftChild|| intersts== rightChild) throw new
IllegalStateException();
    }

    public List getNodes(){
        List<Node> list = new ArrayList<Node>();
        getNodes(list,this);
        return list;
    }

    public boolean hasLeft(){
        return !(leftChild==null);
    }

    public boolean hasRight(){
        return !(rightChild==null);
    }

    private List<Node> getNodes(List<Node> list, Node node){
        if (hasLeft()) {getNodes(list,this.leftChild);
            if (hasRight()) getNodes(list, this.rightChild);
        } else list.add(node);
        return list;
    }

}
```

3.3.3 Пул ієрархічних вузлів

Оскільки на початку та в процесі побудови ієрархічного дерева ми будемо мати слабко зв’язні елементи, варто потурбуватись про місце збереження даних елементів.

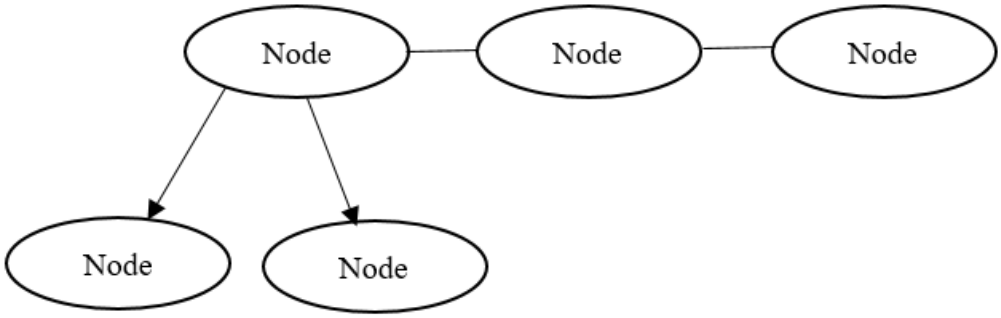

Рисунок 3.4 – Масив кореневих вузлів проміжних дерев

На рисунку 3.4 показано ланцюг (масив) з кореневих вузлів, при об’єднанні двох вузлів, посилання на обидва елементи видаляється з пулу, а натомість створюється новий елемент Node. Даний елемент додається в пул (масив), також він містить видалені на попередньому кроці два посилання в якості піддерев.

```
package model.heap;

import model.node.Node;

import java.util.List;

public class HeapTree {
    List <Node> list ;
    static HeapTree singleObj;

    private HeapTree(){}

    static public HeapTree getHeap(){
        if (singleObj == null) return new HeapTree();
        return singleObj;
    }

…
…
…
…

}
```

3.4 Основні засоби мультипотокового програмування

Більшість низько-кваліфікованих програмістів нехтують потужністю мультипотокового програмування або проектують логіку з критичними помилками що може призвести до неочікуваних наслідків при виконані.

3.4.1 Пул потоків

У Java потоки відображаються на рівні системного рівня, що є ресурсами операційної системи. Якщо неконтрольовано створювати потоки, ці ресурси можуть швидко закінчитися [29].

Перемикання контексту між потоками здійснюється також операційною системою - для того, щоб імітувати паралельність.

Шаблон пулу потоків допомагає економити ресурси в багатопотоковій програмі, а також утримувати паралелізм у певних заздалегідь визначених межах.

При використані пулу потоків, необхідно написати власний паралельний код у вигляді паралельних завдань і подати його на виконання до екземпляра пулу потоків.

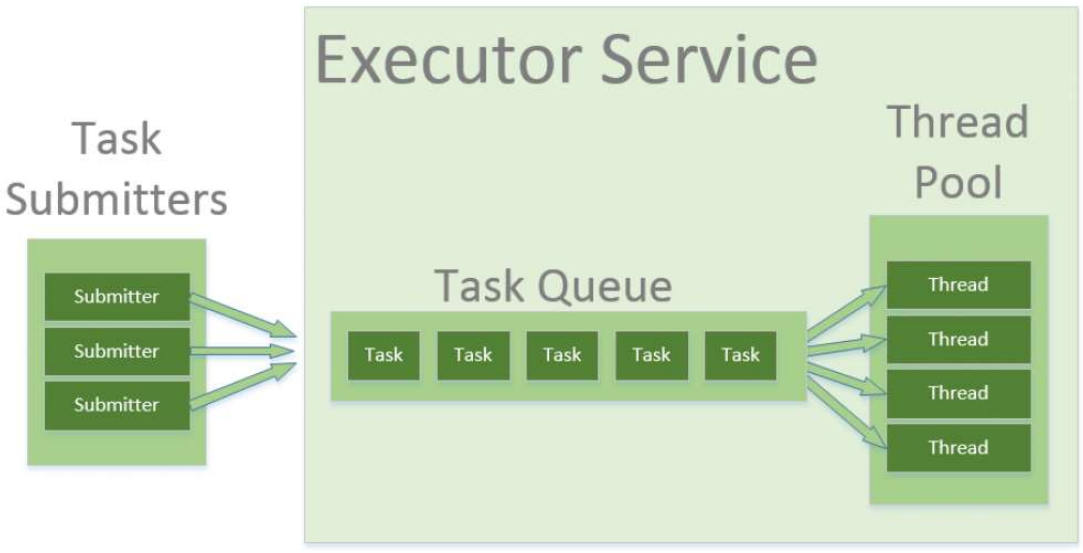

Рисунок 3.5 – Черга на виконання

Даний шаблон дозволяє контролювати кількість потоків, які створює програма, їх життєвий цикл, а також планувати виконання завдань і зберігати вхідні завдання в черзі [30].

Допоміжний клас Executors містить кілька методів для створення попередньо налаштованих для вас екземплярів пулу потоків.

Інтерфейси Executor та ExecutorService використовуються для роботи з різними реалізаціями пулу потоків у Java. Зазвичай необхідно тримати код не відокремленим від фактичної реалізації пулу потоків і використовувати дані інтерфейси у власному додатку [31].

ThreadPoolExecutor - це розширювана реалізація пулу потоків з безліччю параметрів та гачків для тонкої настройки.

Основними параметрами конфігурації, є: corePoolSize, maximumPoolSize і keepAliveTime.

Пул складається з фіксованої кількості основних потоків, які постійно утримуються всередині, і деяких надмірних потоків, які можуть бути створені, а потім припинені, коли вони більше не потрібні. Параметр corePoolSize - це кількість основних потоків, які будуть створені і збережені в пулі. Коли надходить нове завдання, якщо всі основні потоки зайняті, а внутрішня черга заповнена, пулу дозволяється збільшуватися до maxPoolSize.

Оптимальний розмір пулу потоків для підтримки процесорів на бажаному рівні завантаженості є:

$$N_{\text{потоків}} = N_{CPU} * U_{CPU} * (1 + \frac{W}{C}), \quad (3.1)$$

де,

N_{CPU} – число процесорів (ядер);

U_{CPU}- цільова задіяність процесора ($0<U_{CPU}<1$);

$\frac{W}{C}$- відношення часу очікування до часу обчислення .

```
@ThreadSafe
public class BoundedExecutor {
   private final Executor exec;
   private final Semaphore semaphore;

   public BoundedExecutor(Executor exec, int bound) {
      this.exec = exec;
      this.semaphore = new Semaphore(bound);
   }

   public void submitTask(final Runnable command)
      throws InterruptedException {
      semaphore.acquire();
      try {
         exec.execute(new Runnable() {
   public void run() {
      try {
         command.run();
      } finally {
         semaphore.release();
      }
            }
         });
      } catch (RejectedExecutionException e) {
         semaphore.release();
      }
   }
}
```

Вище наведений шматок коду використовує executor для виконання поставлених завдань і семафор для обмеження кількості разів виконання коду.

3.4.2 Модифікатори доступу

Основною небезпекою мультипотокового програмування є порушення цілісності об’єктів та виконуваної логіки, що відбувається при звертанні до змінних (mutable) об’єктів. Одним з механізмів збереження цілісності є використання синхронізованих методів, моніторів, модифікаторів типу, заміна на незміні об’єкти [33].

Основне місце в збережені цілісності логіки при спільному використанні змінних об’єктів є займає модифікатор доступу volatile що накладає певні додаткові умови на читання/запис змінних та гарантує що кожен потік буде мати

ідентичну зміну в власному стеку. Даний прийом запобігає стану гонки та вирішує левову частку проблем.

3.4.3 Бар’єр

При використанні засобів багато потокового програмування також виникає завдання синхронізації потоків, коли для початку наступного завдання необхідно отримати результат поточного.

Одним з таких синхронізаторів є CyclicBarrier. Він дозволяє пулу потоків очікувати загальної точки виконання[35].

CyclicBarriers використовуються в програмах, в яких у нас є фіксоване число потоків, які повинні зв’язати одного, щоб досягти загальної точки, перш ніж продовжити виконання.

Бар'єр називається циклічним, тому що його можна використовувати повторно після звільнення очікуваних потоків[36].

Конструктор для CyclicBarrier доволі проста. Необхідно вказати одне ціле число, яке визначає кількість потоків, яке необхідно викликати методом await () для екземпляра бар’єра, щоб визначити досягнення загальної точки виконання.

```
public class CellularAutomata {
    private final Board mainBoard;
    private final CyclicBarrier barrier;
    private final Worker[] workers;
    public CellularAutomata(Board board) {
        this.mainBoard = board;
        int count = Runtime.getRuntime().availableProcessors();
        this.barrier = new CyclicBarrier(count,
            new Runnable() {
                public void run() {
                    mainBoard.commitNewValues();
                }
            });
        this.workers = new Worker[count];
        for (int i = 0; i < count; i++)
            workers[i]  =  new  Worker(mainBoard.getSubBoard(count,
i));
    }
    private class Worker implements Runnable {
        private final Board board;
```

```
        public Worker(Board board) {
            this.board = board;
        }
        public void run() {
            while (!board.hasConverged()) {
                for (int x = 0; x < board.getMaxX(); x++)
                    for (int y = 0; y < board.getMaxY(); y++)
                        board.setNewValue(x,   y,   computeValue(x,
y));
                try {
                    barrier.await();
                } catch (InterruptedException ex) {
                    return;
                } catch (BrokenBarrierException ex) {
                    return;
                }
            }
        }
    }
    public void start() {
        for (int i = 0; i < workers.length; i++)
            new Thread(workers[i]).start();
        mainBoard.waitForConvergence();
    }
}
```

3.5 Побудова матриці подібностей

Важливим кроком при побудові ієрархічного кластеру є побудова матриці подібностей. В java використання матриць не є хорошим тоном, рекомендованим є використання складних структур даних з пакету collection frameworks. Проте в деяких випадках використання масивів є обумовлена необхідністю великої кількості звертань [38].

3.5.1 Порівняння наборів даних

Порівняти отримані множини можна використавши бінарну міру подібності, що була запропонована Полем Жаккаром. Використовуючи різницю об’єднання та перетину множин отримаємо коефіцієнт подібності n-вимірних векторів, що в свою чергу відображає відповідність спектру інтересів користувачів один до одного [37].

Так міру подібності можна знайти за формулою:

$$J(A,B) = \frac{A \cap B}{A \cup B}. \quad (3.2)$$

Що в декартовому просторі буде виглядати як (рис. 3.6):

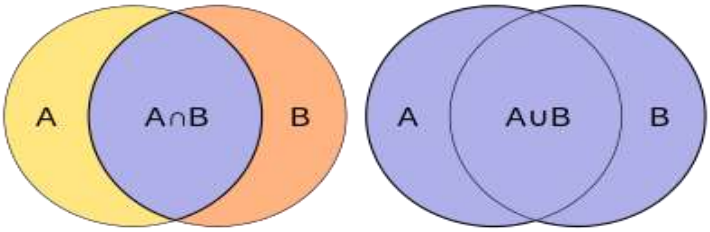

Рисунок 3.6 – Диз’юнкція та кон’юнкція множин

Першим кроком буде отримання пулу кореневих вузлів та створення матриці ідентичної розмірності. Витягування з кореневого вузла інформацію про вузли сформованого кластеру, та пошук мінімальної відстані.

Наступним кроком є використання різниці об’єднання та перетину множин ми отримаємо коефіцієнт подібності n-вимірних векторів що в свою чергу відображає відповідність спектру інтересів користувачів один до одного.

3.5.2. Побудова ієрархічного дерева на основі матриці подібностей

Отримавши матрицю подібностей необхідно вибрати два вузла з пулу кореневих вузлів для подальшого об’єднання. Для цього необхідно пройти матрицю та знайти найбільший результат, що являє собою коефіцієнт подібності [39].

3.6 Тестування на коректність

Для тестування на коректність зроблено функціональне тестування створених компонентів. Даний тип тестування програмного забезпечення,

перевіряє програмну систему на відповідність функціональним вимогам і специфікаціям. Мета функціональних тестів - перевірити кожну функцію програмного додатку, надаючи відповідні вхідні дані, перевіряючи вихідні дані відповідно до функціональних вимог.

Функціональне тестування в основному передбачає тестування чорних ящиків, і його не турбує вихідний код програми. Це тестування перевіряє користувальницький інтерфейс, API, базу даних, безпеку, взаємодію клієнт / сервер та інші функції тестованої програми. Тестування може проводитися як вручну, так і за допомогою автоматики [40].

В результаті тестування було виявлено ряд помилок.

3.7 Продуктивність

Продуктивність створеного додатку є доволі низькою через велику кількість перехресних посилань, великий час виділення та утримування потоків (рис. 3.7).

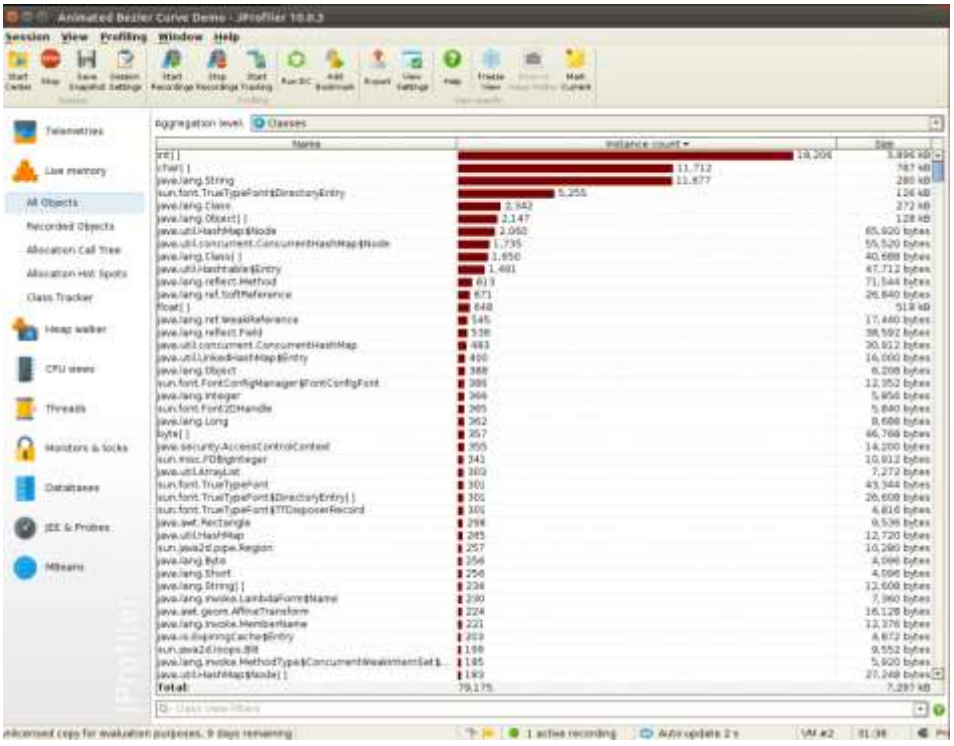

Рисунок 3.7 – Використання пам’яті

Одним з небагатьох позитивних досягнень є низьке використання пам’яті, завдяки використанню перехресних посилань при побудові вектору інтересів та пулу потоків.

3.8 Висновки

В даному розділі розглянуто сучасні технології та підходи проектування та розробки програмного забезпечення, з допомогою яких було створено і протестований даний модуль інформаційної системи.

Проведено аналіз використовуваних структур даних, а саме:

- збалансовані бінарні дерева;
- вузли;
- куча;
- багатовимірні масиви.

Виконано розподіл класів збереження даних і класів логіки. Розглянуто та використано патерни мультипотокового програмування, а саме семафорів, бар’єрів та пулу потоків виконання. Зроблено огляд модифікаторів доступу для забезпечення цілісності даних та логіки при виконанні багатопотокових програм.

На основі структури HashSet реалізовано метод Жаккара для порівняння двох векторів, що дозволяє надати оцінку близькості векторів, що використовується для побудови матриці близькостей.

Окрім цього при розробці було використано перехресні посилання на незміні об’єкти.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Оскільки у даному випадку процес праці являє собою дослідницьку діяльність, яка проводиться за персональним комп'ютером, то аналіз небезпечних або шкідливих факторів в умовах праці буде розглядатись саме з цієї позиції.

4.1 Застереження нещасних випадків та управління ризиками

Сучасний етап функціонування промислових об'єктів характеризується ускладненням контролю за умовами праці та виробничого середовища. Якщо протягом попередніх періодів вибір способів і шляхів комплексного вирішення проблем розвитку підприємства та визначення основних параметрів об'єктів проводилися, в першу чергу, на основі мінімізації економічних витрат, то сьогодні стає актуальним питання оцінювання можливих негативних наслідків їх експлуатації і, в першу чергу – рівня безпеки виробництва. Один з найважливіших напрямків вирішення проблеми – прийняття комплексу технічних і організаційних рішень на основі концепцій теорії ризику [41].

Під ризиком розуміють кількісну міру небезпеки, що враховує ймовірність настання негативних наслідків від здійснення господарської діяльності та можливий розмір втрат від них. Тоді прийнятний ризик – це такий ризик, який не перевищує гранично допустимого рівня. Для кожного об'єкта, що досліджується, та його персоналу можна розрахувати (з використанням відповідних методик) ризик таких негативних подій, як аварія або нещасний випадок. Після порівняння значень розрахункового та прийнятного ризиків можна зробити обґрунтований висновок щодо рівня безпеки об'єкта. Відповідно до концепції «прийнятного ризику», практична діяльність підприємства не може бути виправданою, якщо вигода від цієї діяльності в

цілому не перевищує викликаного нею ймовірного збитку. Під час планування заходів щодо забезпечення безпечних (нешкідливих) умов функціонування треба враховувати весь спектр існуючи небезпек. Обґрунтованим вважається варіант збалансованих витрат на створення систем безпеки за рахунок зниження рівня ризику та підвищення вигоди, яка одержується від господарської діяльності.

Виходячи з формалізованого підходу, ризики можна поділити на три категорії [42]:

- прийнятний ризик (рівень ризику, з яким суспільство в цілому може миритися заради одержання визначних благ чи вигоди у результаті своєї діяльності);
- ризик, що потребує подальших оцінок;
- неприйнятний ризик (рівень ризику, що встановлюється адміністративними чи регулювальними органами як максимальний, вище якого необхідно вживати заходи для його усунення).

Прийнятність ризику в різних ситуаціях може бути визначено, виходячи з аналізу чинного законодавства з промислової безпеки, правил і норм безпеки, додаткових вимог наглядових органів, наявних статистичних даних про негативні події та їх наслідки. Метою аналізу ризику є ідентифікація та оцінка чинників, що впливають на небезпеку об’єкта, оцінка ймовірності негативної дії її наслідків [43].

Слід зауважити, що Закон України від 22.02.2001 «Про страхові тарифи на загальнообов’язкове державне соціальне страхування від нещасного випадку на виробництві, які спричинили втрату працездатності» встановлює страхові тарифи на страхування залежно від професійного ризику виробництва. Отже, в законодавчому порядку закріплено механізм економічного управління діяльністю зі створення безпечних нешкідливих умов праці на виробництві. Оскільки віднесення підприємства до одного з класів здійснюється шляхом аналізу стану охорони праці за попередній період, керівництво підприємства

зацікавлене в створенні таких умов праці, що дозволять знизити професійний ризик на наступний рік, та, як наслідок, сумарні витрати з охорони праці. При цьому існує проблема визначення цього ризику, а також вибору оптимальних рішень щодо покращення умов праці. Для обґрунтування вибору оптимального рішення з множини можливих використовуються спеціальні методи підтримки прийняття рішень та експертні оцінки [44].

Сьогодні існують методики оцінювання ризику аварій для об'єктів різних галузей, насамперед – для об'єктів підвищеної небезпеки. Однак питання наукового обґрунтування кількісної оцінки ризиків виробничого травматизму для конкретних виробництв залишаються невирішеними. Пропонується алгоритм прийняття управлінських рішень на базі оцінки ризиків настання травматизму для конкретних виробничих умов. Процес прийняття рішень з управління ризиком травматизму викладено у вигляді алгоритму, що складається з наступних етапів.

Етап 1. Отримання інформації про ситуацію. Основним при отриманні інформації про ситуацію прийняття рішення є підготовка аналітичного матеріалу, що відображає основні характеристики та тенденції розвитку ситуації. Для сфери охорони праці актуальним є застосування методів порівняльної оцінки з іншими періодами або з іншими робочими місцями, де використовуються аналогічні технологічні процеси та обладнання. Щоб мати можливість встановити динаміку розвитку ситуації під впливом тих або інших чинників, необхідно перейти до кількісних методів, вводячи в розгляд кількісні характеристики чинників у вигляді змінних, значення яких можуть змінюватися в певному діапазоні залежно від можливих впливів.

Для виявлення чинників, що визначають розвиток ситуації, може бути використано спеціально розроблені методи, такі як факторний, регресійний, кореляційний аналіз та інші.

Етап 2. Прогноз розвитку ситуації. Особливу роль при прийнятті рішень відіграють проблеми, пов'язані з оцінкою очікуваного розвитку ситуації, що

аналізується, та очікуваних результатів реалізації запропонованих альтернативних варіантів рішень.

Не прогнозуючи хід розвитку подій, керувати, принаймні, є нерозумним. Оскільки при використанні експертної інформації велике значення мають не лише кількісні, але й якісні оцінки, традиційні методи розрахунків прогнозів далеко не завжди можуть бути застосовані. До того ж, в багатьох складних ситуаціях не завжди особа, яка приймає рішення (далі – ОПР), володіє достовірною статистичною інформацією, необхідною для розробки прогнозу. Зазначені вище причини роблять актуальною проблему застосування методів прогнозування, що орієнтуються на роботу як із кількісними даними, так і з якісними експертними оцінками.

Етап 3. Генерування та оцінка альтернативних варіантів рішень. Генерування альтернативних варіантів рішень, керуючих впливів тощо може здійснюватись або безпосередньо, або за допомогою спеціальних процедур. При генеруванні альтернативних варіантів управлінських рішень має в повній мірі використовуватися інформація щодо ситуації прийняття рішення, результати аналізу та оцінювання ситуації, результати її діагностики та прогнозу розвитку ситуації при різних альтернативних варіантах можливого розвитку подій.

Після того, як розроблено альтернативні варіанти управлінських впливів, представлені у вигляді ймовірної технологічної послідовності дій, можливих способів реалізації варіантів рішень, має бути здійснений їх попередній аналіз з метою відсіювання варіантів, які не можна застосувати, або поступаються іншим, також запропонованим до розгляду. При відборі основних варіантів управлінських впливів необхідно враховувати як їх досить високу порівняльну оцінку, так і відсутність дублювання, щоб спектр альтернативних варіантів рішень, відібраних для більш глибокого оцінювання, був досить повним, але не надто широким. У підсумку, у розпорядженні ОПР залишається k різних способів керуючих впливів

(управлінських рішень) на стан умов праці в ситуації, що склалася: U={Uk}. Аналіз декількох альтернативних варіантів розвитку ситуації, зазвичай, виявляється більш інформативним та сприяє виробленню більш ефективних рішень.

Етап 4. Прийняття рішення ОПР. Результати попередньої оцінки альтернативних варіантів рішень є основою для прийняття остаточного варіанту управлінського рішення. Задача прийняття рішень з управління ризиком травматизму полягає в обґрунтованому визначенні критеріїв, застосування яких до множини наявних альтернатив можливих рішень дозволить вибрати найбільш придатну альтернативу для досягнення поставленої мети. Частота настання нещасного випадку:

$$\lambda_H = \frac{H_{CP}}{Ч_{CP} \cdot 240} \qquad (4.1)$$

де H_{CP} – середньоарифметична кількість нещасних випадків за даною професією у даному виді економічної діяльності за останні 3 роки;

$Ч_{CP}$ – середньорічна кількість працюючих у даному виді економічної діяльності за 3 роки;

240 – кількість робочих днів у році з урахуванням відпустки.

Введення такого коефіцієнту при подальшому обрахуванні загального ризику настання нещасного випадку з різними матеріальними втратами – від нещасного випадку без втрати робочих днів (переведення на легкий труд) до важкого та нещасного випадку зі смертельним наслідком дає можливість нівелювати ці розбіжності.

4.2. Освітлення виробничих приміщень для роботи з ВДТ та локальній комп’ютерній мережі

Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ) [45].

Основними вимогами є:

- приміщення для роботи з ВДТ повинні мати природне та штучне освітлення;
- природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природної освітленості (КПО) не нижче ніж 1,5 %.

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ ЕОМ та ПЕОМ, має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, у разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення, додатково встановлюються світильники місцевого освітлення).

Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300 - 500 лк. Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використовувати місцеве освітлення. При цьому світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати бліків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

Як джерела світла в разі штучного освітлення мають застосовуватись переважно люмінесцентні лампи типу ЛБ. У разі влаштування відбитого освітлення у виробничих та адміністративно-громадських приміщеннях допускається застосування метало-галогенних ламп потужністю 250 Вт. Допускається застосування ламп розжарювання у світильниках місцевого освітлення[46].

Система загального освітлення має становити суцільні або переривчасті лінії світильників, розташовані збоку від робочих місць (переважно ліворуч), паралельно лінії зору працюючих.

Допускається використання світильників таких класів світлорозподілу:

- прямого світла - П;
- переважно прямого світла - Н;
- переважно відбитого світла - В.

Для загального освітлення слід застосовувати світильники серії ЛПО 3б із дзеркальними гратами, укомплектовані високочастотними пускорегулювальними апаратами (ВЧ ПРА). Допускається застосовувати світильники цієї серії без ВЧ ПРА тільки в модифікації "Кососвітло". Застосування світильників без розсіювачів та екрануючих грат заборонено.

Яскравість світильників загального освітлення в зоні кутів випромінювання від 50 до 90 град. з вертикаллю в повздовжній та поперечній площині має становити не більше ніж 200 кд/кв. м, захисний кут світильників - не менше ніж 40 град.

Світильники місцевого освітлення повинні мати просвічуючи відбивач із захисним кутом не меншим ніж 40 град.

Слід передбачити обмеження прямої блискоти від джерел природного та штучного освітлення. При цьому яскравість світлих поверхонь (вікна, джерела штучного освітлення), що розташовані в полі зору повинна бути не більше ніж 200 кд/кв. м.

Необхідно обмежувати блискіт на робочих поверхнях відносно джерел природного і штучного освітлення. При цьому яскравість бліків на екрані ВДТ має не перевищувати 40 кд/кв. м, а яскравість стелі в разі застосування системи відбитого освітлення - 200 кд/кв. м.

Показник осліпленості у разі використання джерел загального штучного освітлення у виробничих приміщеннях має не перевищувати 20, а показник дискомфорту в адміністративно-громадських приміщеннях має бути не більше за 40.

Необхідно обмежувати нерівномірність розподілу яскравості в полі зору працюючих з ВДТ. При цьому співвідношення яскравостей робочих поверхонь має бути не більшим ніж 3:1, а співвідношення яскравостей робочих поверхонь та поверхонь стін, обладнання тощо - 5:1.

Коефіцієнт запасу (К куб.) для освітлювальних установок загального освітлення має дорівнювати 1,4.

Коефіцієнт пульсації має не перевищувати 5 %, що забезпечується застосуванням газорозрядних ламп у світильниках загального та місцевого освітлення з ВЧ ПРА для світильників будь-яких типів. Якщо не має світильників з ВЧ ПРА, то лампи багатолампових світильників або світильники загального освітлення, розташовані поруч, слід вмикати на різні фази трьохфазної мережі.

Для забезпечення нормованих значень освітленості у приміщеннях з ВДТ ЕОМ та ПЕОМ слід чистити шибки і світильники принаймні двічі на рік і вчасно замінювати лампи, що перегоріли.

Отже, для безпечного виробничого процесу необхідно дотримуватись правил прописаних в санітарних нормах. Дотримання даних правил мінімізує загальну шкоду на організм.

ВИСНОВКИ

У кваліфікаційній роботі виконано завдання покращення якості побудови кластерів використовуючи ієрархічні кластеризацію та отримано такі результати:

- в процесі аналізу існуючих технологій кластеризації обґрунтовано актуальність завдання підвищення якості кластеризації;
- встановлено, що для побудови ієрархічних кластерів, враховуючи специфіку роботи методів швидкої кластеризації, існуючі методи є доволі не точними, та обґрунтовано необхідність детальнішого дослідження методів пришвидшення побудови ієрархічних кластерів;
- запропоновано метод пришвидшення ієрархічної кластеризації, а саме, здійснювати обробку матриці подібностей та векторів інтересів паралельно розбивши на дрібніші завдання;
- на основі запропонованих рішень для пришвидшеного опрацювання проміжних кроків кластеризації створено програмне забезпечення у вигляді окремого пакету;
- проведено тестування розробленого ПЗ, в результаті чого визначено, що використання запропонованого методу дозволяє збільшити швидкість побудови ієрархії відносно не розподіленого методу;
- також визначено що при малій кількості даних, засоби що підтримують цілісність логіки і даних забирають велику кількість ресурсів, що в результаті призводить до не задовільних результатів.

ПЕРЕЛІК ДЖЕРЕЛ

1. Apache mahout: overview [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://mahout.apache.org/docs/latest/– Дата доступу: 10.11.2020.

2. Java Competitve Learning Application [Електронний ресурс] - Режим доступу: http://homepages.feis.herts.ac.uk/~nngroup/software.html

3. A. K. Jain. Data clustering: a review / A. K. Jain, M. N. Murty, P. J. Flynn.– ACM Comput. Surv.–1999. – No31. – 60 с.

4. Goetz, Brian; Joshua Bloch; Joseph Bowbeer; Doug Lea; David Holmes; Tim Peierls (2006). Java Concurrency in Practice. Addison Wesley. - 384 p.

5. Witten, Ian H. (2011). Data Mining - Practical Machine Learning Tools and Techniques with JAVA Implementations. Elsevier. – 416 p.

6. Rokach, Lior, and Oded Maimon. «Clustering methods.» Data mining and knowledge discovery handbook. Springer US, 2005. 321—352.

7. «Штейнгауз Р.» Математичний калейдоскоп. — М.: Наука, 1981. — 160 с.

8. G. N. Lance, W. T. Williams; A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems, The Computer Journal, Volume 9, Issue 4, 1 February 1967, Pages 373–380.

9. «Sneath P.H.A., Sokal R.R.» Numerical taxonomy: The principles and practices of numerical classification. — San-Francisco: Freeman, 1973. — 573 p.

10. "How Google AdWords Works". Google AdWords [Електронниий ресурс] - Режим доступу: https://adwords.google.com/home/ – Дата доступу: 10.11.2020.

11. “Business Manager Overview – facebook” [Електронниий ресурс] - Режим доступу: https://business.facebook.com/ – Дата доступу: 10.11.2020.

12. Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys). Boston, MA, 191--198.

13. "Deezer.com Traffic, Demographics and Competitors - Alexa". www.alexa.com. – дата доступу 18.11.2020.

14. “A Guide To Tinder” [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.help.tinder.com/hc/en-us/categories/115000755686-A-Guide-To-Tinder- – дата доступу 19.11.2020.

15. “Badoo FAQ". Badoo. [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://web.archive.org/web/20150107031642/http://corp.badoo.us/en/faq/ – дата доступу 16.11.2020.

16. “ чатпростотак: технические вопросы” [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://chatprostotak.com/uid/mDLvooGxZZHZK1F9ogaJ - дата доступу 20.11.2020

17. “Amino : Community Guidelines” [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://support.aminoapps.com/hc/en-us/articles/360022679554-Community-Guidelines - дата доступу 21.11.2020

18. “Java™ Programming Language” [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/technotes/guides/language/ – Дата доступу: 10.11.2020.

19. “IntelliJ IDEA overview ” [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.jetbrains.com/help/idea/discover-intellij-idea.html – Дата доступу: 10.11.2020.

20. GIT [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Git. – Дата доступу: 10.11.2020.

21. “Welcome to Apache Maven” [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://maven.apache.org/– Дата доступу: 10.11.2020.

22. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009). Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. ISBN 978-0262033848.

23. “ The Java™ Tutorials Generic Types” [Електронний ресурс]. – 2020. – режим доступу https://docs.oracle.com/javase/tutorial/java/generics/types.html - дата доступу - 10.11.2020.

24. “Java 8 Concurrency Tutorial: Threads and Executors” [Електронний ресурс]. – 2020. – режим доступу https://winterbe.com/posts/2015/04/07/java8-concurrency-tutorial-thread-executor-examples/ – Дата доступу: 10.11.2020.

25. “Concurrent Programming in Java: Design Principles and Patterns “— Addison Wesley, 1999. — ISBN 0-201-31009-0.

26. Frank Nielsen (2016). "Chapter 8: Hierarchical Clustering". Introduction to HPC with MPI for Data Science. Springer.

27. Kaufman, L.; Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis (1 ed.). New York: John Wiley. ISBN 0-471-87876-6.

28. CORMEN, THOMAS H. (2009). INTRODUCTION TO ALGORITHMS. United States of America: The MIT Press Cambridge, Massachusetts London, England. pp. 151–152. *ISBN 978-0-262-03384-8*.

29. "The Java® Language Specification, Java SE 7 Edition". Oracle Corporation. [Електронний ресурс]. – 2020. – режим доступу http://docs.oracle.com/javase/specs/jls/se7/html/jls-17.html#jls-17.4.4 – Дата доступу: 10.11.2020.

30. *Black, Paul E. "array". Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology.* [Електронний ресурс]. – 2020. – режим доступу: https://xlinux.nist.gov/dads/HTML/array.html – Дата доступу: 02.11.2020.

31. Dasgupta, Abhiit (2014). Set theory: with an introduction to real point sets. New York: Birkhäuser.

32. Management and Processing of Complex Data Structures: Third Workshop on Information Systems and Artificial Intelligence, Hamburg, Germany, February 28 - March 2, 1994. Proceedings, ed. Kai v. Luck, Heinz Marburger, p. 76

33. Rokach, Lior, and Oded Maimon. "Clustering methods." Data mining and knowledge discovery handbook. Springer US, 2005. 321-352.

34. Zhang, et al. "Agglomerative clustering via maximum incremental path integral." Pattern Recognition (2013).

35. “Примеры Java-семафоров” [Електронний ресурс]. – 2020. – режим доступу https://www.codeflow.site/ru/article/java__java-semaphore-examples/– Дата доступу: 10.11.2020.

36. “Пять секретов... многопоточного Java-программирования” [Електронний ресурс]. – 2020. – режим доступу https://www.ibm.com/developerworks/ru/library/j-5things15/index.html – Дата доступу: 10.11.2020.

37. “The Jaccard Similarity algorithm” [електроний ресурс]. – 2020 –режим доступу: - https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/jaccard/ – Дата доступу: 10.11.2020.

38. “ThreadPoolExecutor Java™ Platform Standard Ed. 7” [електроний ресурс]. – 2020 – режим доступу: - https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html - Дата доступу: 10.11.2020.

39. “Collections in Java” ” [електроний ресурс]. – 2020 –режим доступу: https://www.javatpoint.com/collections-in-java – Дата доступу: 10.11.2020.

40. “JUnit 5 User Guide” [електроний ресурс] – 2020 – режим доступу до ресурсу https://junit.org/junit5/docs/current/user-guide/

41. Класифікація ризиків. Види ризиків та їх показники. // URL: https://pidruchniki.com/72392/ekologiya/klasifikatsiya_rizikiv.

42. ЗАКОН УКРАЇНИ. Про страхові тарифи на загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності // URL: https://zakon.rada.gov.ua/laws/show/2272-140 .

43. Закон України "Про охорону праці" [електроний ресурс]. – 2020 – режим доступу: -https://dnaop.com/html/3428/doc-zakon-ukrajini-pro-ohoronu-praci - Дата доступу: 10.11.2020.

44. Ткачук К. Н. Застосування інформаційних систем в галузі охорони праці: науково-методичний посібник / К. Н. Ткачук, О. Є. Кружилко, Н. А. Праховнік. – К.: Експодата, 2004. – 186 с.

45. “Освітлення виробничих приміщень” [електроний ресурс]. – 2020 – режим доступу: - https://buklib.net/books/35234/ - Дата доступу: 10.11.2020.

46. Смирнов В.А., Дикань С.А. Безпека життєдіяльності: навч. посібник. К. : Кафедра. 2012. 304 с.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

УДК 004.032.24
С.С. Заверуха
(Тернопільський національний технічний університет імені Івана Пулюя)

ВИКОРИСТАННЯ ЗАСОБІВ БАГАТОПОТОКОВОГО ПРОГРАМУВАННЯ ДЛЯ ПРИШВИДШЕННЯ ПОБУДОВИ МАТРИЦІ ПОДІБНОСТЕЙ N-ВИМІРНИХ ВЕКТОРІВ

UDC 004.032.24
S.S. Zaverukha

USING MULTITHREADED PROGRAMMING TO ACCELERATE THE CONSTRUCTION OF A MATRIX OF SIMILARITIES OF N-DIMENSIONAL VECTORS

Починаючи з кінця 90-х років XX століття в зв'язку з необхідністю підвищення ефективності виконання операцій обчислювальною технікою для вирішення завдань в різних сферах набуває популярності парадигма мультипотокового програмування. Під мультипотоком розуміється можливість програмного забезпечення підтримувати виконання кількох потоків одночасно.

При розробці програмного забезпечення часто виникає проблема вибору інструменту розробки. Одним з можливих рішень даної проблеми є використання мови програмування Java та бібліотеки (пакету) Concurrency, що дозволяє будувати власний застосунок використовуючи всю потужність багатоядерних систем. Попри це, мультипотокове програмування є доволі складним процесом що має низку проблем, такі як: велика складність, непередбачуваність швидкості виділення потоку, ресурсний голод, стан гонки чи взаємне блокування потоків.

Можливими варіантами виходу з проблеми є використання атомарних типів, синхронізаця методів, модифікатора volatile, заміна змінних (mutable) об'єктів на не зміні (immutable). Особливу увагу необхідно звернути на використання незмінних об'єктів, даний підхід є ефективним рішенням у випадку створення малих об'єктів, проте при створенні великих об'єктів це надзвичайно сильно завантажує підсистему пам'яті спочатку при створені а потім при видалені.

Основною метою даного дослідження є поліпшення архаїчного методу ієрархічної кластеризації, що складається з таких кроків: 1) пошук матриці відповідності та 2) попарне об'єднання вузлів в агломератному підході.

Завдання пошуку матриці відповідності векторів є доволі простим але затратним по часу при виконанні в одному потоці, однак його можна виконати швидко та безпечно з допомогою бібліотеки concurrency.

Паралельне програмування скорочує час виконання розпаралеленої частини, проте пришвидшення не приймає лінійної залежності від кількості виділених потоків. Виділення та запуск нового процесу забирає певний період машинного часу, окрім цього потокам необхідна синхронізація та програмний бар'єр для зберігання цілісності логіки.

Отже, використання паралельного програмування є ефективним при розробці додатків що обробляють великі масиви даних, проте вони накладають певні обмеження та складності.

Література:

1. Goetz, Brian; Joshua Bloch; Joseph Bowbeer; Doug Lea; David Holmes; Tim Peierls (2006). Java Concurrency in Practice. Addison Wesley. - 384 p.
2. Witten, Ian H. (2011). Data Mining - Practical Machine Learning Tools and Techniques with JAVA Implementations. Elsevier. – 416 p.

Додаток Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

ІХ Міжнародної науково-технічної конференції молодих учених та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

УДК 004.043

С.С. Заверуха
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ВИКОРИСТАННЯ БІНАРНИХ N-ВИМІРНИХ ВЕКТОРІВ ДЛЯ ВСТАНОВЛЕННЯ МІРИ ПОДІБНОСТІ КОРИСТУВАЧІВ ІНФОРМАЦІЙНИХ СИСТЕМ

S.S. Zaverukha

USING BINARY N-DIMENSIONAL VECTORS TO ESTABLISH THE SIMILARITY OF INFORMATION SYSTEM USERS

У еру розвитку сучасних комп'ютерних технологій постає проблема звуження соціальних кіл, що створює труднощі в комунікативному аспекті користувачам інформаційних систем. Ця, проблема обумовлена не тільки внутрішніми чинниками (характер, емоційна зрілість), а й зовнішніми (зміна місця проживання тощо.). Враховуючи, що людина являє собою соціальну істоту, яка має власний спектр інтересів, пошук важливих соціальних контактів на основі тем, що цікавлять є необхідною соціальною потребою.

Через складність пошуку важливих соціальних контактів відбувається процес асиміляції діаспор, малочисельних культур та субкультур. У певних випадках результатом уподібнення є доволі болісний процес інтеграції особистості у соціум, що призводить до стику двох культур. Як результат, в певних випадках процес інтеграції особистості в нове оточення є досить складним, що в подальшому може призвести до стику двох культур. Одним з варіантів вирішення даної проблеми є формування кластеру подібних користувачів на основі множин вподобань конкретних особистостей. Оскільки, кластерний аналіз являє собою статистичну процедуру збору та пошуку груп наближено схожих об'єктів, він дозволяє легше зрозуміти набір даних, та обрати відповідні методи обробки для певної групи об'єктів[1]. Для побудови кластеру можна скористатись графовим методом кластеризації, а саме методом ієрархічної кластеризації (кластерного дерева). Даний метод будує ієрархію кластерів з окремих елементів шляхом поступового злиття кластерів, орієнтуючись на матрицю відстаней між вузлами графа [2]. Порівняти отримані множини можна використавши бінарну міру подібності, що була запропонована Полем Жаккаром. Використовуючи різницю об'єднання та перетину множин ми отримуємо коефіцієнт подібності n-вимірних векторів, що в свою чергу відображає відповідність спектру інтересів користувачів один до одного[3].

Розробками алгоритмів для кластеризації користувачів на основі отриманих даних займається ряд компаній, такі як Rakuten, Facebook, Google та інші. Окрім цього створено програми, які за допомогою алгоритмів кластеризації, здійснюють розподіл користувачів на різні підгрупи. Проте, зважаючи на точність та недосконаліть сучасних методів кластеризації n-вимірних векторів, дослідження даної технології все ще, залишається складною задачею.

Література

1. Chapter 8: Hierarchical Clustering. Introduction to HPC with MPI for Data Science. Springer [електроний ресурс] – режим доступу: https://www.springer.com/gp/book/9783319219028

2. Java Competitve Learning Application [Електронний ресурс] - Режим доступу: http://homepages.feis.herts.ac.uk/~nngroup/software.html

3. SimMetrics a sourceforge implementation of Jaccard index and many other similarity metrics [Електронний ресурс] - Режим доступу: http://sourceforge.net/projects/

Додаток В

```
package model.node;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class Node {
    String name;
    Set<String> intersts;

    public Node leftChild;
    public Node rightChild;

    public Node (String name, Node leftChild, Node rightChild,
HashSet<String> intersts){
        //this.name= name;
        this.leftChild=leftChild;
        this.rightChild= rightChild;
        this.intersts= intersts;
        //if (intersts == null && rightChild==null) throw new
IllegalStateException();
    }

    public List getNodes(){
        List<Node> list = new ArrayList<Node>();
        getNodes(list,this);
        return list;
    }

    public boolean hasLeft(){
        return !(leftChild==null);
    }

    public boolean hasRight(){
        return !(rightChild==null);
    }

    private List<Node> getNodes(List<Node> list, Node node){
        if (hasLeft()) {getNodes(list,this.leftChild);
            if (hasRight()) getNodes(list, this.rightChild);
        } else list.add(node);
        return list;
    }

}
```

```
package com.apporiented.algorithm.clustering;

public class Distance implements Comparable<Distance>, Cloneable {

    private Double distance;
    private Double weight;

    public Distance() {
          this(0.0);
    }

    public Distance(Double distance) {
          this(distance, 1.0);
    }

    public Distance(Double distance, Double weight) {
        this.distance = distance;
        this.weight = weight;
    }

    public Double getDistance() {
        return distance;
    }

    public void setDistance(Double distance) {
        this.distance = distance;
    }

    public Double getWeight() {
        return weight;
    }

    public void setWeight(Double weight) {
        this.weight = weight;
    }

    public boolean isNaN() {
        return distance == null || distance.isNaN();
    }

    @Override
    public int compareTo(Distance distance) {
        return distance == null ? 1 :
getDistance().compareTo(distance.getDistance());
    }

    @Override
    public String toString() {
          return String.format("distance : %.2f, weight : %.2f",
distance, weight);
     }}
```

```
package com.apporiented.algorithm.clustering;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class HierarchyBuilder {

private DistanceMap distances;
private List<Cluster> clusters;
private int globalClusterIndex = 0;

public DistanceMap getDistances() {
return distances;
}

public List<Cluster> getClusters() {
return clusters;
}

public HierarchyBuilder(List<Cluster> clusters, DistanceMap
distances) {
this.clusters = clusters;
this.distances = distances;
}

/**
* Returns Flattened clusters, i.e. clusters that are at least
apart by a given threshold
* @param linkageStrategy
* @param threshold
* @return flat list of clusters
*/
public List<Cluster> flatAgg(LinkageStrategy linkageStrategy,
Double threshold)
{
while((!isTreeComplete()) && (distances.minDist() != null) &&
(distances.minDist() <= threshold))
{

//System.out.println("Cluster Distances: " +
distances.toString());
```

```
//System.out.println("Cluster Size: " + clusters.size());
agglomerate(linkageStrategy);
}
//System.out.println("Final MinDistance: " +
distances.minDist());
//System.out.println("Tree complete: " +
isTreeComplete());
return clusters;
}
public void agglomerate(LinkageStrategy linkageStrategy) {
ClusterPair minDistLink = distances.removeFirst();
if (minDistLink != null) {
clusters.remove(minDistLink.getrCluster());
clusters.remove(minDistLink.getlCluster());
Cluster oldClusterL = minDistLink.getlCluster();
Cluster oldClusterR = minDistLink.getrCluster();
Cluster newCluster =
minDistLink.agglomerate(++globalClusterIndex);
for (Cluster iClust : clusters) {
ClusterPair link1 = findByClusters(iClust,
oldClusterL);
ClusterPair link2 = findByClusters(iClust,
oldClusterR);
ClusterPair newLinkage = new ClusterPair();
newLinkage.setlCluster(iClust);
newLinkage.setrCluster(newCluster);
Collection<Distance> distanceValues = new
ArrayList<Distance>();
if (link1 != null) {
Double distVal = link1.getLinkageDistance();
Double weightVal =
link1.getOtherCluster(iClust).getWeightValue();
distanceValues.add(new Distance(distVal, weightVal));
distances.remove(link1);
}
if (link2 != null) {
```

```
Double distVal = link2.getLinkageDistance();
Double weightVal =
link2.getOtherCluster(iClust).getWeightValue();
distanceValues.add(new Distance(distVal, weightVal));
distances.remove(link2);
}
Distance newDistance =
linkageStrategy.calculateDistance(distanceValues);
newLinkage.setLinkageDistance(newDistance.getDistance());
distances.add(newLinkage);
}
clusters.add(newCluster);
}
}
private ClusterPair findByClusters(Cluster c1, Cluster c2) {
return distances.findByCodePair(c1, c2);
}
public boolean isTreeComplete() {
return clusters.size() == 1;
}
public Cluster getRootCluster() {
if (!isTreeComplete()) {
throw new RuntimeException("No root available");
}
return clusters.get(0);
}
```