МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

БОЙКО Ярослав Володимирович

Математичне та програмне забезпечення кластеризації ухвалення рішень/Mathematical tools and software for clustering of decision making

спеціальність: 8.05010301 - Програмне забезпечення систем
магістерська програма - Програмне забезпечення систем

Магістерська робота

Виконав студент групи
ПЗСм-21/14
Я. В. Бойко

Науковий керівник:
Ю. Р. Піговський

Магістерську роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2017

ЗМІСТ

ВСТУП

Актуальність теми. В рамках кожної управлінської функції ухвалюються певні рішення, типові для різних функцій управління. Виконання загальних і конкретних функцій управління вимагає ухвалення рішення. Наприклад, при плануванні ухвалюються планові рішення, при організації - організаційні, при мотивації - стимулюючі рішення.

Ухвалення рішення, як і обмін інформацією - складова частина будь-якої управлінської функції. Виконання різних функцій управління можна представити як послідовність відповідних рішень.

Можна говорити, що функція ухвалення рішень виконує в процесі управління особливу роль — вона необхідна для здійснення всіх інших функцій.

Прийняття рішень відбувається в часі, тому вводиться поняття "процес ухвалення рішення", який складається з послідовності етапів і процедур і направлений на розв'язання проблемної ситуації [1].

З процесом ухвалення рішення суспільство стикається щоденно у повсякденному побуті. Проте одним з найважливіших етапів у житті свідомого громадянина є вибір гідного кандидата на посаду депутата. Виборці звертають увагу на біографічні особливості депутата та його законодавчої діяльності.

Основним завданням є поєднання процесу ухвалення рішення та одного з методів кластеризації, що дозволить зробити правильний вибір серед великої кількості кандидатів у депутати.

Зв’язок роботи з науковими програмами, планами, темами

Напрямок виконаних досліджень безпосередньо пов’язаний з науково-дослідним напрямком кафедри “комп’ютерних наук” Тернопільського національного економічного університету.

Мета і задачі дослідження. Метою роботи є розробка алгоритму прийняття рішення на основі ієрархічної кластеризації з використанням двох основних методів – роздільного та об'єднуючого, який дозволить громадянам

України зробити правильний вибір серед великої кількості кандидатів у депутати.

Основними задачами дослідження є:

- проведення аналізу алгоритмів кластеризації;
- розробка алгоритму прийняття рішення на основі методу ієрархічної кластеризації;
- розробка інформаційної системи для підтримки процесів прийняття рішення на основі методу ієрархічної кластеризації.

Об'єкт та предмет дослідження. Об'єктом дослідження є процес прийняття рішення за допомогою алгоритму кластеризації. Предметом дослідження є вирішення задачі процесу прийняття рішення за допомогою ієрархічного алгоритму.

Методи дослідження. Теоретичні дослідження по розробці моделі користувача ґрунтуються на застосуванні системного аналізу, методології функціонального моделювання, інженерії знань, семантичних мереж, теорії множин, теорії графів, теорії нечіткого виведення, теорії алгоритмів та об'єктно-орієнтованого проектування.

Наукова новизна одержаних результатів. Запропоновано алгоритм прийняття рішення на основі ієрархічної кластеризації з використанням двох основних методів – роздільного та об'єднуючого, який дозволить громадянам України вирішити проблему вибору з-поміж великої кількості кандидатів обрати гідного відповідальної посади народного депутата України.

Практичне значення одержаних результатів полягає у наступному: спроектовано online-помічник із запропонованим алгоритмом прийняття рішення на основі ієрархічної кластеризації з використанням двох основних методів – роздільного та об'єднуючого.

Особистий внесок магістранта. Всі результати отримані автором самостійно.

Апробація результатів дипломної роботи. Основні положення дипломної роботи апробовані на п'ятій Всеукраїнській школі-семінарі молодих

вчених і студентів «Сучасні комп’ютерній інформаційні технології», що проходила 22-23 травня 2015 року у м. Тернополі на базі Тернопільського національного економічного університету.

Публікації. Бойко Я.В., Струбицька І.П. Ієрархічний метод кластеризації для задачі ухвалення рішень / Бойко Я.В., Струбицька І.П. // “Сучасні комп’ютерні інформаційні технології”: Матеріали V Всеукраїнської школи-семінару молодих вчених і студентів ACIT'2015 — Тернопіль : ТНЕУ, 2015. – 111-112 с.

РОЗДІЛ 1
АНАЛІЗ АЛГОРИТМІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСТЕРИЗАЦІЇ УХВАЛЕННЯ РІШЕНЬ

1.1. Особливості формування думки громадянина щодо здійснення вибору кандидата.

На сьогоднішній день важливим є визначення власної позиції громадян, чітке уявлення якої здатне сформувати бачення щодо підбору найефективнішого та працездатнішого складу законодавчої гілки влади, з метою недопуску недобросовісних депутатів, що в подальшому може згубно вплинути на розвиток всієї держави. Яскравим прикладом виступають події зими 2013-2014 років, а саме так звана «Революція Гідності», яка зуміла об’єднати український народ по всіх містах країни.

У моменти, коли народному обранцю потрібно довести свою здатність принести користь суспільству, важливим є розвіювання сумнівів громадян щодо довіри окремому депутату або партії задля здійснення правильного вибору.

Зараз, у зв’язку із переходом від архаїчних поглядів щодо ведення політики до прогресивніших методів та підходів, проявляється тенденція щодо збільшення кількості молодих та перспективних кандидатів у депутати, які здатні привити політичному бомонду нові бачення у законотворчості та інших галузях політичної діяльності. Але, попри це випливає основна проблема, а саме питання довіри громадян обранцям та дилема молодості проти досвіду.

Попри всі події в людей залишилися сумніви, якому депутату та блоку чи партії довіряти більше, адже їх дуже багато. Народний депутат – це особа, обрана до складу Верховної Ради України [2]. Простіше кажучи – це людина, яка буде приймати важливі рішення. У своєму виборі у громадян зазвичай прийнято відштовхуватись від загально прийнятих особистісних якостей кандидата:

- молодий чи досвідчений;

- аскетичний чи сноб;
- комунікабельний чи замкнутий;
- креативний чи безталанний;
- професійний або просто хороша людина;
- з нових або з колишніх.

Недобросовісні депутати, які проходять до складу Верховної Ради України можуть спричинити ряд проблем:

- погано вплинути на розвиток країни;
- використання повноважень для власного збагачення;
- зловживання повноваженнями;
- здійснення злочинних дій прикриваючись депутатською недоторканністю.

Найчастіше питання довіри постає під час виборів. Виборці звертають увагу на такі критерії діяльності депутата:

- законопроекти, які депутат висунув;
- законопроекти, які депутат підтримав;
- законопроекти, які депутат не підтримав;
- законопроекти, за які депутат не голосував;
- біографічні особливості.

Наведені вище фактори дозволяють людині більш детально побачити напрямок діяльності обраного депутата.

Отже, для забезпечення ефективного функціонування гілки законодавчої влади актуальною є проблема прийняття рішення у виборі кандидата на місце законодавця, що в подальшому здатне забезпечити становлення України, як самодостатньої розвинутої держави. Наведені фактори дозволяють людині більш детально аналізувати напрям діяльності та активність обраного депутата.

1.2. Аналіз відомих методів кластеризації.

У багатьох прикладних задачах вимірювати ступінь подібності об'єктів істотно простіше, ніж формувати ознаковий опис. Наприклад, набагато простіше порівняти дві фотографії і сказати, що вони належать одній людині, ніж зрозуміти, на підставі яких ознак вони схожі. Головна відмінність кластеризації від класифікації полягає в тому, що перелік груп чітко не заданий і визначається в процесі роботи певного алгоритму [3].

Іншими словами кластеризація - це поділ множини вхідних векторів на групи (кластери) за ступенем «схожості» один на одного [4]. Усередині кожної групи повинні опинитися «схожі» об'єкти, а об'єкти різних груп повинні відрізнятися один від одного [5].

Завдання кластеризації полягає в наступному. Задана вибірка $X^\ell = \{x_1, \ldots, X_\ell\} \subset X$ і функція відстані між об'єктами $\rho(x, x')$. Завдання полягає у розбитті вибірки на непересічні підмножини, названі кластерами, так, щоб кожен кластер складався з об'єктів, близьких по метриці ρ, а об'єкти різних кластерів істотно відрізнялися. При цьому кожному об'єкту $x_i \in X^\ell$ приписується мітка (номер) кластера y_i.посилання

Алгоритм кластеризації - це функція $a: X \to Y$, яка будь-якому об'єкту $x \in X$ ставить у відповідність мітку кластера $y \in Y$. Безліч міток Y в деяких випадках відомо наперед. Однак часто виникає задача визначення оптимального числа кластерів, з точки зору того чи іншого критерію якості кластеризації.

Вирішення задачі кластеризації принципово неоднозначне. По-перше, не існує однозначно найкращого критерію якості кластеризації. Відомий цілий ряд корисних критеріїв, а також ряд алгоритмів, які не мають чітко вираженого параметру, але здійснюють досить хорошу кластеризацію «з побудови». Всі вони можуть давати різні результати. По-друге, число кластерів, як правило, невідоме наперед і встановлюється відповідно до деякого суб'єктивного

критерію. По-третє, результат кластеризації істотно залежить від метрики ρ, вибір якої, як правило, також суб'єктивний і визначається експертом.

Цілі кластеризації можуть бути різними і залежать від особливостей конкретної прикладної задачі:

- зрозуміти структуру множини об'єктів X^ℓ, розбивши їх на групи схожих об'єктів. Спростити подальшу обробку даних і прийняття рішень, працюючи з кожним кластером окремо (стратегія «розділяй і володарюй»);
- скоротити обсяг збережених даних у разі надвеликої вибірки X^ℓ, залишивши по одному найбільш типовому представникові від кожного кластера;
- виділити нетипові об'єкти, які не підходять до жодного з кластерів. Цю задачу називають однокласовою класифікацією, виявленням нетиповості або новизни (novelty detection).

У першому випадку число кластерів намагаються зробити якомога меншим. У другому випадку важливо забезпечити високу ступінь подібності об'єктів усередині кожного кластера, а кластерів може бути безліч. У третьому випадку найбільший інтерес представляють окремі об'єкти, які не вписуються в жоден із кластерів.

У всіх цих випадках можна застосовувати ієрархічну кластеризацію, коли великі кластери дробляться на більш дрібні, ті в свою чергу дробляться ще на дрібніші, і т. д. Такі задачі називаються алгоритмами таксономії (taxonomy). Результатом таксономії є не просте розбиття множини об'єктів на кластери, а деревоподібна ієрархічна структура. Замість номера кластера об'єкт характеризується перерахуванням всіх кластерів, до яких він належить (від великого до дрібного).

Таксономія будується в багатьох областях знань, щоб упорядкувати інформацію про великі кількості об'єктів [6].

Прості алгоритми кластеризації, такі як алгоритм k-середніх, як правило, вузько спеціалізовані і дають адекватні результати тільки в одній-двох ситуаціях. Складніші алгоритми, такі як FOREL або алгоритм Ланса-Вільямса,

справляються з декількома типами ситуацій. Проте створення алгоритму, працюючого успішно у всіх ситуаціях без винятку, представляється важкою і в більшості випадків не вирішуваною задачею [7].

Для того, щоб визначити «схожість» об'єктів для початку потрібно скласти вектор характеристик для кожного об'єкта. Зазвичай це набір числових значень, наприклад, ріст-вага людини. Однак існують також алгоритми, що працюють з якісними (так званими категорійними) характеристиками.

Після того, як визначено вектор характеристик, можна провести нормалізацію, щоб всі компоненти давали однаковий результат при розрахунку «відстані». У процесі нормалізації всі значення зводяться до деякого діапазону, наприклад, [-1, -1] або [0, 1].

На останньому етапі, для кожної пари об'єктів вимірюється «відстань» між ними - ступінь схожості. Існує безліч метрик для визначення ступеня схожості:

1. Евклідова відстань - найпоширеніша функція відстані. Це геометрична відстань в багатовимірному просторі:

$$\rho(x, x') = \sqrt{\sum_i^n (x_i - x_i)^2}, \quad (1.1)$$

де ρ – відстань між об’єктами x; x_i – значення; i – властивості об′єкта x.

2. Квадрат евклідової відстані. Використовується для додання більшого значення віддаленим один від одного об'єктам. Ця відстань обчислюється за формулою:

$$\rho(x, x') = \sum_i^n (x_i - x_i)^2, \quad (1.2)$$

3. Відстань Геммінга (манхеттенська відстань, сіті-блок відстань, відстань міських кварталів). Це різниця по координатах. У більшості випадків

ця міра відстані приводить до таких же результатів, як і для звичайної відстані Евкліда. Проте відзначу, що для цієї міри вплив окремих великих різниць (викидів) зменшується (так як вони не зводяться в квадрат). Відстань Геммінга обчислюється за формулою:

$$\rho(x, x') = \sum_i^n |x_i - x_i|, \tag{1.3}$$

4. Відстань Чебишева. Це відстань може виявитися корисною, коли потрібно визначити два об'єкти як «різні», якщо вони відрізняються по якій-небудь одній координаті. Іншими словами приймає значення найбільшого модуля різниці між значеннями відповідних властивостей (ознак) об'єктів. Відстань Чебишева обчислюється за формулою:

$$\rho(x, x') = \max(|x_i - x_i|), \tag{1.4}$$

5. Степенева відстань. Використовується у випадку, коли необхідно збільшити або зменшити значення, що відноситься до розмірності, для якої відповідні об'єкти сильно відрізняються. Степенева відстань обчислюється за наступною формулою:

$$\rho(x, x') = \sqrt[r]{\sum_i^n (x_i - x_i)^p} \tag{1.5}$$

де r і p - параметри, які визначаються користувачем.

Параметр p відповідає за поступове зважження різниць по окремих координатах, параметр r відповідає за прогресивне зважження великих відстаней між об'єктами. Якщо обидва параметри, r і p, дорівнюють двом, то ця відстань збігається з відстанню Евкліда. Рекомендується вибирати значення обох параметрів у межах від 1 до 4.

Вибір метрики повністю залежить від типу задачі та дослідника, оскільки результати кластеризації можуть істотно відрізнятися при використанні різних метрик.

Отже, класифікують наступні методи по обробці даних:

- неієрархічні методи;
- методи за способом аналізу даних;
- методи за кількістю застосувань алгоритмів кластеризації;
- методи по можливості розширення обсягу оброблюваних даних;
- методи за часом виконання кластеризації.

Неієрархічні методи, засновані на поділі, які представляють собою ітераційні методи дроблення вихідної сукупності. У процесі поділу нові кластери формуються доти, поки не буде виконано правило зупинки.

Така неієрархічна кластеризація полягає в розділенні набору даних на певну кількість окремих кластерів. Існує два підходи. Перший полягає у визначенні меж кластерів як найбільш щільних ділянок в багатовимірному просторі вихідних даних, тобто визначення кластера там, де є велике "згущення точок". Другий підхід полягає в мінімізації міри розбіжності об'єктів.

Найбільш поширеним серед неієрархічних методів є алгоритм k-середніх, також відомий, як швидкий кластерний аналіз. Приклад результату кластеризації із згаданим вище алгоритмом зображений на рисунку 1.1.

Для можливості використання цього методу необхідно мати гіпотезу про найбільш ймовірний кількості кластерів. Тому методи неієрархічної кластеризації та зокрема алгоритм k-середніх не підходить для вирішення задачі ухвалення рішення.

Наступними є методи за способом аналізу даних. Розрізняють чіткі і нечіткі алгоритми. Чіткі (або непересічні) алгоритми кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одному кластеру.

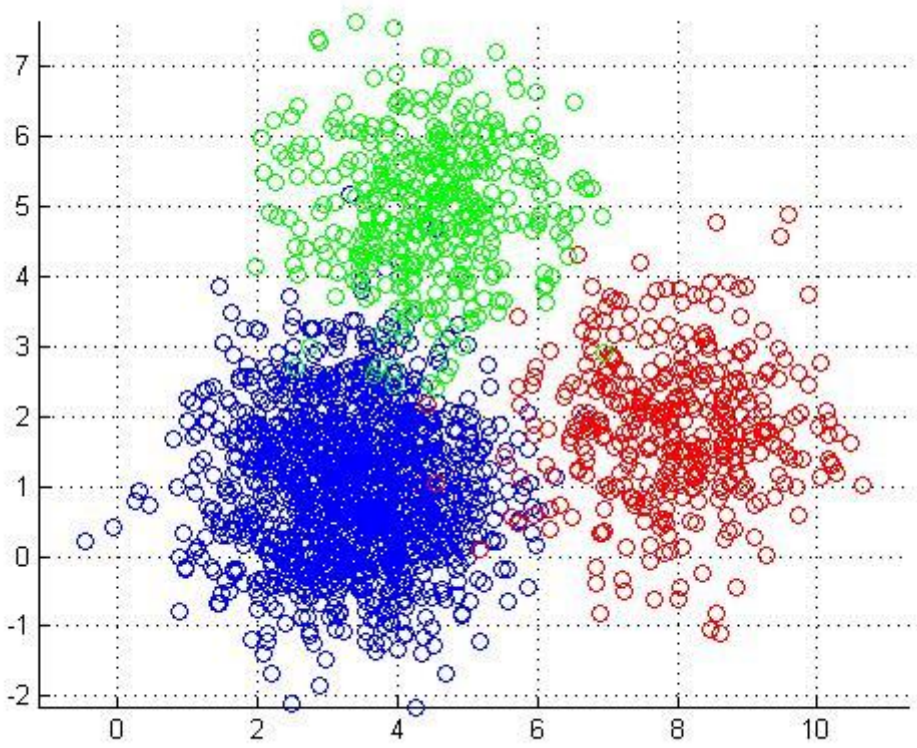

Рис. 1.1. Результат виконання алгоритму k-середніх

Приклад нечіткого алгоритму кластеризації зображений на рисунку 1.2.

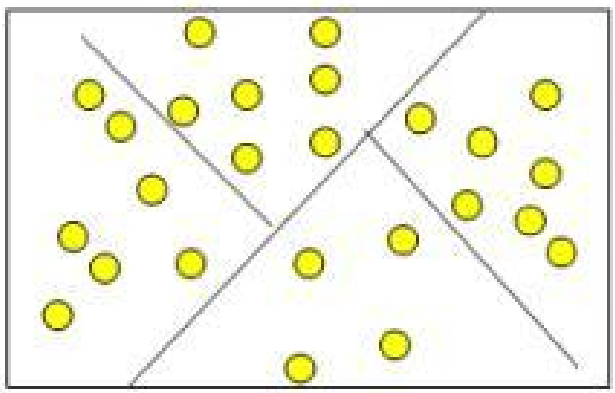

Рис. 1.2. Алгоритму чіткої кластеризації

Нечіткі (або пересічні) алгоритми кожному об'єкту ставлять у відповідність набір речових значень, що показують ступінь відносини об'єкта

до кластерів. Приклад нечіткого алгоритму кластеризації зображений на рисунку 1.3.

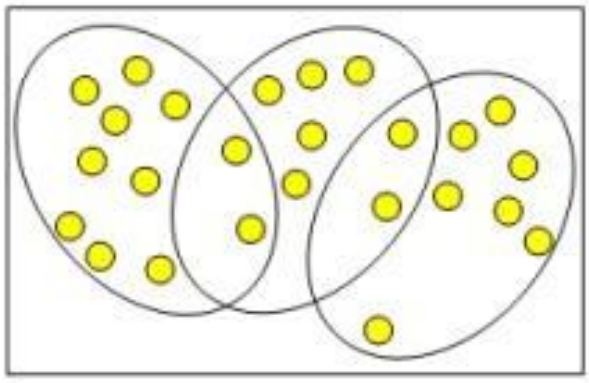

Рис. 1.3. Алгоритму нечіткої кластеризації

Тобто кожен об'єкт відноситься до кожного кластеру з деякою ймовірністю. Цей алгоритм може не підійти, якщо заздалегідь невідомо число кластерів, або необхідно однозначно віднести кожен об'єкт до одного кластеру.

Популярним алгоритмом нечіткої кластеризації є алгоритм c-середніх (c-means). Він являє собою модифікацію методу k-середніх. Цей алгоритм можна представити такими кроками:

1. Вибрати початкове нечітке розбиття n об'єктів на k кластерів шляхом вибору матриці приналежності U розміру n x k;

2. Використовуючи матрицю U, знайти значення критерію нечіткої помилки:

$$E^2(X,U) = \sum_{i=1}^{N}\sum_{k=1}^{K} U_{ik} \left\| x_i^{(k)} - c_k \right\|^2, \tag{1.6}$$

де c_k - «центр мас» нечіткого кластера k:

$$c_k = \sum_{i=1}^{N}\sum_{i=1}^{N} U_k x_i, \tag{1.7}$$

3. Перегрупувати об'єкти з метою зменшення цього значення критерію нечіткої помилки;

4. Повертатися в п. 2 до тих пір, поки зміни матриці U не стануть незначними.

Цей алгоритм може не підійти, якщо заздалегідь невідомо число кластерів, або необхідно однозначно віднести кожен об'єкт до одного кластеру.

За кількістю застосувань алгоритмів кластеризації методи діляться на методи з одноетапною кластеризацією та методи з багатоетапною кластеризацією. Також розрізняють методи по можливості розширення обсягу оброблюваних даних, які діляться на масштабовані та не масштабовані. Крім цього існують методи за часом виконання кластеризації, які включають в себе потокові та не потокові алгоритми.

У алгоритмах квадратичної помилки задачу кластеризації розглядають як побудову оптимального розбиття об'єктів на групи. При цьому оптимальність може бути визначена як вимога мінімізації середньоквадратичної помилки розбиття:

$$e^2(X, L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2, \qquad (1.8)$$

де c_j - «центр мас» кластера j (точка з середніми значеннями характеристик для даного кластера).

Алгоритми квадратичної помилки відносяться до типу плоских алгоритмів. Найпоширенішим алгоритмом цієї категорії є метод k-середніх. Цей алгоритм будує задане число кластерів, розташованих якнайдалі один від одного. Робота алгоритму ділиться на кілька етапів:

1. Випадково вибрати k точок, які є початковими «центрами мас» кластерів;

2. Віднести кожен об'єкт до кластеру з найближчим «центром мас»;

3. Перерахувати «центри мас» кластерів згідно з їх поточним складом;

4. Якщо критерій зупинки алгоритму не задоволений, повернутися до п.2.

В якості критерію зупинки роботи алгоритму зазвичай вибирають мінімальну зміну середньоквадратичного відхилення. Так само можна зупинити роботу алгоритму, якщо на кроці 2 не було об'єктів, що перемістилися з кластера в кластер.

До недоліків даного алгоритму можна віднести необхідність задавати кількість кластерів для розбиття.

Суть алгоритмів, які засновані на теорії графів, полягає в тому, що вибірку об'єктів представлено у вигляді графа G = (V, E), вершинам якого відповідають об'єкти, а ребра мають вагу, рівну «відстані» між об'єктами. Перевагою графових алгоритмів кластеризації є наочність, відносна простота реалізації і можливість внесення різних удосконалень, заснованих на геометричних міркуваннях.

В алгоритмі виділення зв'язних компонентів задається вхідний параметр R і в графі видаляються всі ребра, для яких «відстані» більше R. Сполученими залишаються тільки найближчі пари об'єктів. Сенс алгоритму полягає в тому, щоб підібрати таке значення R, що лежить в діапазоні всіх «відстаней», при якому граф «розвалиться» на кілька зв'язних компонентів. Отримані компоненти і є кластери.

Для підбору параметра R зазвичай будується гістограма розподілів попарних відстаней. У завданнях з добре вираженою кластерною структурою даних на гістограмі буде два піки - один відповідає внутрішньо кластерним відстаням, другий-міжкластерним відстаням. Параметр R підбирається із зони мінімуму між цими піками. При цьому управляти кількістю кластерів за допомогою порога відстані досить важко. Приклад гістограми можна побачити на рисунку 1.4.

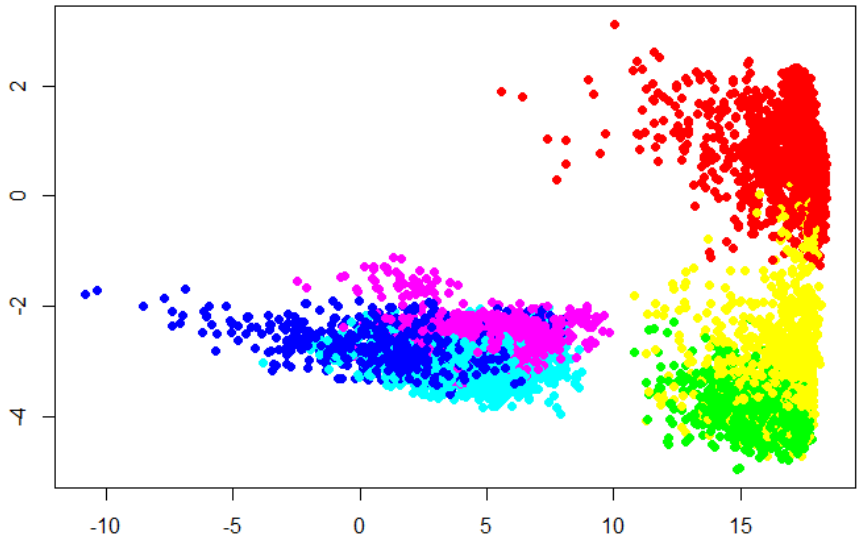

Рис. 1.4. Гістограми кластеризації

Варто відзначити два недоліки цього алгоритму:

- Обмежена застосовність. Алгоритм виділення зв'язних компонентів найбільше підходить для виділення кластерів типу згущень або стрічок. Наявність розрідженого фону або «вузьких перемичок» між кластерами призводить до неадекватної кластеризації.

- Погана керованість числом кластерів. Для багатьох додатків зручніше задавати не параметр R, а число кластерів або деякий поріг «чіткості кластерізації». Управляти числом кластерів за допомогою параметра R досить важко. Доводиться кілька разів вирішувати задачу при різних R, що негативно позначається на тимчасових витратах.

Алгоритм найкоротшого незамкнутого шляху будує граф з ℓ-1 ребер так, щоб вони з'єднували всі ℓ точки і володіли мінімальною сумарною довжиною. Такий граф називається найкоротшим незамкнутим шляхом (КНП), мінімальним покриваючим деревом або каркасом. На рисунку 1.5 зображено мінімальне покриваюче дерево, отримане для дев'яти об'єктів.

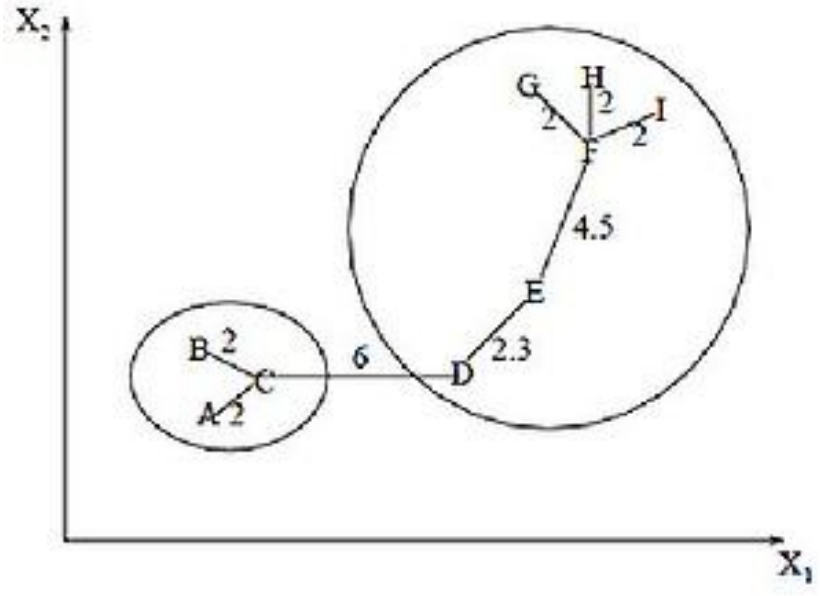

Рис. 1.5. Мінімальне покриваюче дерево

Шляхом видалення зв'язку, поміченої CD, з довжиною рівною 6 одиницям (ребро з максимальною відстанню), отримуємо два кластери: {A, B, C} і {D, E, F, G, H, I}. Другий кластер в подальшому може бути розділений ще на два кластери шляхом видалення ребра EF, яке має довжину, рівну 4, 5 одиницям.

Доведено, що цей граф будується за допомогою нескладної процедури, відповідної крокам 1-4:

1) Знайти пару точок (i, j) з найменшим ρ_{ij} і з'єднати їх ребром;
2) поки у вибірці залишаються ізольовані точки
3) знайти ізольовану точку, найближчу до деякої неізольованої;
4) з'єднати ці дві точки ребром;
5) видалити K - 1 найдовших ребер;

На кроці 5 видаляються K -1 найдовших ребер, і зв'язний граф розпадається на K кластерів.

На відміну від попереднього алгоритму, число кластерів K задається як вхідний параметр. Його можна також визначати графічно, якщо впорядкувати всі відстані, що утворюють каркас, в порядку зменшення і відкласти їх на

графіку. Різкий стрибок вниз десь на початковій ділянці графіка покаже кількість найчіткіше виділених кластерів.

Цей алгоритм, як і попередній, дуже простий і також має обмежену кількість варіантів для застосування. Наявність розрідженого фону або «вузьких перемичок» між кластерами призводить до неадекватної кластеризації. Іншим недоліком КНП є висока трудомісткість - для побудови найкоротшого незамкнутого шляху потрібно O (ℓ^3) операцій.

Алгоритм пошарової кластеризації заснований на виділенні зв'язкових компонент графа на деякому рівні відстаней між об'єктами (вершинами). Рівень відстані задається порогом відстані c. Наприклад, якщо відстань між об'єктами $0 \leq \rho(x, x^1) \leq 1$, то $0 \leq c \leq 1$.

Алгоритм пошарової кластеризації формує послідовність підграфів графа G, які відображають ієрархічні зв'язки між кластерами:

$$G^0 \subseteq G^1 \subseteq \cdots \subseteq G^m, \tag{1.9}$$

де $G^t = (V, E^t)$ — граф на рівні c^t,

$$E^t = \{e_{ij} \in E : \rho_{ij} \leq c_t\}, \tag{1.10}$$

де c^t – t-ий поріг відстані; m – кількість рівней ієрархії; $G^0 = (V, o)$, o – порожня множина ребер графа, получена при $t^0 = 1$; $G^m = G$, тобто граф об’єктів без обмежень на відстань (довжину ребер графа), оскільки $t^m = 1$.

За допомогою зміни порогів відстані $\{c^0, \ldots, c^m\}$, де $0 = c^0 < c^1 < \ldots < c^m = 1$, можливо контролювати глибину ієрархії одержуваних кластерів. Таким чином, алгоритм пошарової кластеризації здатний створювати як плоске розбиття даних, так і ієрархічне.

Алгоритм FOREL (формальні елементи) має численні варіації, в основі яких лежить наступна базова процедура [8, 9].

Нехай задана деяка точка $x_0 \in X$ і параметр R. Виділяються всі точки вибірки $x_i \in X^\ell$, що потрапляють всередину сфери $\rho(x_i, X_0) \leq R$, і точка x_0 переноситься в центр виділених точок. Ця процедура повторюється до тих пір, поки склад виділених точок, а значить і положення центру не перестане змінюватися. Доведено, що згадана процедура сходиться за кінцеве число кроків. При цьому сфера переміщається в місце локального згущення точок. Центр сфери x_0 в загальному випадку не є об'єктом вибірки, тому й називається формальним елементом.

Для обчислення центру необхідно, щоб безліч об'єктів X було не тільки метричним, але й лінійним векторним простором. Ця вимога виконується, коли об'єкти описуються числовими ознаками. Однак існують завдання, в яких спочатку задана тільки метрика, а додавання і множення на число не визначені на X. Тоді в якості центру сфери можна взяти той об'єкт навчальної вибірки, для якого середня відстань до інших об'єктів кластера мінімальна. Відповідно, в алгоритмі на 6 кроці існуюча формула замінюється на:

$$x_0 := \arg\min_{x \in K_0} \sum_{x' \in K_0} \rho(x, x'). \qquad (1.11)$$

При цьому помітно збільшується трудомісткість алгоритму. Якщо в лінійному просторі для обчислення центру потрібно O(K) операцій, то в метричному - $O(K^2)$, де к - чисто точок у кластері. Алгоритм можна дещо прискорити, якщо зауважити, що перерахунок центру при додаванні або видаленні окремої точки кластера вимагає лише O(K) операцій, а в лінійному просторі - O(1).

Різні варіанти алгоритму FOREL відрізняються способами об'єднання сфер в кластери, способами варіювання параметра R, способами вибору початкового наближення для точок x_0. В алгоритмі представлений один з варіантів, в якому сфери будуються послідовно:

1) Ініціалізувати множину некластеризованних точок: $U := X^\ell$;

2) поки у вибірці є некластеризованні точки, $U \neq \emptyset$;

3) взяти довільну точку $x_0 \in U$ випадковим чином;

4) повторювати;

5) утворити кластер - сферу з центром у x_0 і радіусом R;

$$K_0 := \{x_i \in U \mid \rho(x_i, X_0) \leq R\}. \quad (1.12)$$

6) помістити центр сфери в центр мас кластера:

$$x_0 := \frac{1}{K_0} \sum_{x_i \in K_0} x_i; \quad (1.13)$$

7) поки центр x_0 не стабілізується;

8) позначити всі точки K_0 як кластерезовані:

$$U := U \setminus K_0; \quad (1.14)$$

9) застосувати алгоритм КНП до безлічі центрів всіх знайдених кластерів;

10) кожен об'єкт $x_i \in X^\ell$ приписати кластер з найближчим центром.

На кроці 9 до центрів цих сфер застосовується алгоритм найкоротшого незамкнутого шляху. З одного боку, це вирішує проблему низької ефективності згаданого алгоритму, оскільки сфер набагато менше, ніж вихідних об'єктів. З іншого боку, ми отримуємо тоншу, дворівневу, структуру кластерів: кожен кластер верхнього рівня розпадається на більш дрібні підкластери нижнього рівня. Інша перевага цього алгоритму - можливість описувати кластери довільної геометричної форми. Варіюючи параметр R, можна отримувати кластеризацію різного ступеня детальності. Якщо кластери близькі за формою до кулі, можна зробити R великим. Для опису кластерів складнішої форми слід зменшувати R.

Алгоритм FOREL досить чутливий до вибору початкового положення точки x_0 для кожного нового кластера. Для усунення цього недоліку пропонується генерувати декілька (близько 10..20) кластеризацій. Оскільки початкове положення центрів вибирається випадковим чином, отримані кластеризації будуть досить сильно відрізнятися. Остаточно вибирається та кластеризація, яка доставляє найкраще значення заданому функціоналу [10].

Також доречно навести таблицю з перевагами та недоліками найвідоміших алгоритмів кластеризації. Переваги та недоліки алгоритмів до вище перечислених методів наведені в таблиці 1.1.

Таблиця 1.1

Переваги та недоліки алгоритмів

Алгоритм	Перевага	Недолік
CURE	виконує кластеризацію на високому рівні навіть при наявності викидів, виділяє кластери складної форми і різних розмірів, володіє лінійно залежними вимогами до місця зберігання даних і тимчасову складність для даних високої розмірності	є необхідність у завданні порогових значень та кількості кластерів
BIRCH	двоступенева кластеризація, кластеризація великих обсягів даних, працює в обмеженому обсязі пам'яті, є локальним алгоритмом, може працювати при одному скануванні вхідного набору даних, використовує той факт, що дані неоднаково розподілені по простору, і обробляє області з великою щільністю як єдиний кластер	робота з тільки числовими даними, добре виділяє тільки кластери опуклої або сферичної форми, є необхідність у завданні порогових значень
k-середніх	простота використання; швидкість використання; зрозумілість і прозорість алгоритму	алгоритм занадто чутливий до викидів, які можуть спотворювати середнє; повільна робота з великими базами даних; необхідно задавати кількість кластерів; неможливість застосування алгоритму на даних, де наявні пересічні кластери

Продовження таблиці 1.1

PAM	простота використання; швидкість використання; зрозумілість і прозорість алгоритму, алгоритм менш чутливий до викидів у порівнянні з k-means	необхідно задавати кількість кластерів; повільна робота на великих базах даних
CLOPE	висока масштабованість і швидкість роботи, а також якість кластеризації, що досягається використанням глобального критерію оптимізації на основі максимізації градієнта висоти гістограми кластера. Він легко розраховується і інтерпретується. Під час роботи алгоритм зберігає в RAM невелику кількість інформації по кожному кластеру і вимагає мінімальне число сканувань набору даних. CLOPE автоматично підбирає кількість кластерів, причому це регулюється одним єдиним параметром - коефіцієнтом відштовхування	-
Само-організаційна карта Кохонена	використовується універсальний аппроксіматор – нейронна мережа, навчання мережі без вчителя, самоорганізація мережі, простота реалізації, гарантоване отримання відповіді після проходження даних по шарах	робота тільки з числовими даними, мінімізація розмірів мережі, необхідно задавати кількість кластерів
Алгоритм НСМ	легкість реалізації, обчислювальна простота	завдання кількості кластерів, відсутність гарантії в знаходженні оптимального рішення
Fuzzy C-means	нечіткість при визначенні об'єкта в кластер дозволяє визначати об'єкти, які знаходяться на кордоні, в кластери	обчислювальна складність, встановлення кількості кластерів, виникає невизначеність з об'єктами, які віддалені від центрів всіх кластерів

1.3. Постановка задачі дослідження

З першого пункту зрозуміло, що в силу нестабільної політичної ситуації в державі та низьким рівнем довіри громадян до народних обранців, суспільство має потребу в сервісі, який допоможе їм із вибором кандидата та подальшою підтримкою його на виборах. Основною проблемою є те, що кількість кандидатів на посаду законотворця з кожним разом зростає у геометричній

прогресії. Появляються молоді, розумні та прогресивні кандидати, інформації про яких дуже мало. Крім цього на думку громадян впливає нестабільна політична ситуації. Тому для розвіювання сумнівів виборці найчастіше звертають увагу на наступні фактори, а саме:

- належність кандидата до певної партії;
- минуле кандидата;
- його політична діяльність;
- діяльність на благо громадян.

Звісно перелічені вище фактори є важливими для формування думки громади. Проте мало хто звертає увагу на законотворчі здібності кандидата, які не аби як впливають на розвиток країни.

Крім цього майбутній обранець приймає участь у прийняті інших законів, що є не менш важливим фактором у його політичній біографії. На превеликий жаль, як уже було згадано, більшість людей не звертає уваги на згадані фактори.

Для того щоб вирішити дану проблему потрібно розробити web-орієнтовану систему, яка буде використовувати один із методів кластеризації для здійснення пошуку, який допоможе зробити правильний вибір громадянину України.

У системі буде передбачений функціонал, який оптимізує роботу користувачеві:

- поле для пошуку депутата;
- поле для пошуку закону;
- перегляд списку законів до яких автор має відношення;
- перегляд статистики голосування за шуканий законопроект;
- перегляд детальної інформації про закон.

Перерахований вище функціонал буде представлений в зручному інтерфейсі користувача, що значно спростить роботу громадян.

Висновки до розділу 1

1. Проведено аналіз проблем сучасного суспільства, а саме проблема довіри громадян обранцям.
2. Запропоновано однин із варіантів вирішення згаданого вище питання - це використання кластеризації.
3. Здійснено ознайомлення та проаналізовано основні методи кластеризації.
4. Здійснено постановку ряду задач для кластеризації вибірки об’єктів. При цьому необхідно вирішити ключову проблему забезпечення можливості громадянину у прийнятті рішення щодо вибору гідного народного депутата. Для вирішення поставленої проблеми потрібно застосувати якісний метод кластеризації, який дозволить скоротити часові, фізичні та матеріальні затрати у формуванні бачення громадянином оптимального складу Верховної Ради України.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ КЛАСТЕРИЗАЦІЇ УХВАЛЕННЯ РІШЕНЬ

2.1. Обґрунтування вибору ієрархічного підходу кластеризації ухвалення рішень

Як можна побачити з пункту 1.2 існує велика кількість різноманітних методів та алгоритмів кластеризації даних. Кожен з них підходить для вирішення певного типу задачі. Проте майже всі алгоритми мають певний ряд недоліків, які не підходять для вирішення поставленої задачі. Найвагомішими недоліками розглянутих вище методів та алгоритмів є:

- вказання кількості кластерів для початку роботи алгоритму;
- повільна робота з великими базами даних;
- неможливість застосування алгоритму на даних, де наявні пересічні кластери;
- робота тільки з числовими даними;
- відсутність гарантії в знаходженні оптимального рішення;
- виникнення невизначеності з об'єктами, які віддалені від центрів всіх кластерів.

Враховуючи недоліки розглянутих вище методів найбільш підходящим способом вирішення задачі кластеризації ухвалення рішень буде застосування методу ієрархічної кластеризації.

Ієрархічна кластеризація - процес організації даних в деревовидну структуру, в основі яких стоїть подібність цих даних. Цей метод дуже потужний і корисний для аналізу великих наборів даних [11]. Основним принципом роботи даного алгоритму є створення набору елементів у дереві. Дерево має безліч гілок, якщо елементи відрізняються один від одного, до них приєднуються довгі гілки, і навпаки, якщо їх подібність збільшується, тоді гілки зменшуються.

Серед алгоритмів ієрархічної кластеризації виділяють два основних типи:

- висхідні або об’єднуючі алгоритми;
- низхідні або роздільні алгоритми.

Низхідні або роздільні алгоритми працюють за принципом «зверху-вниз»: на початку всі об'єкти поміщаються в один кластер, який потім розбивається на дрібніші кластери. Приклад роздільного алгоритму зображений на рисунку 2.1.

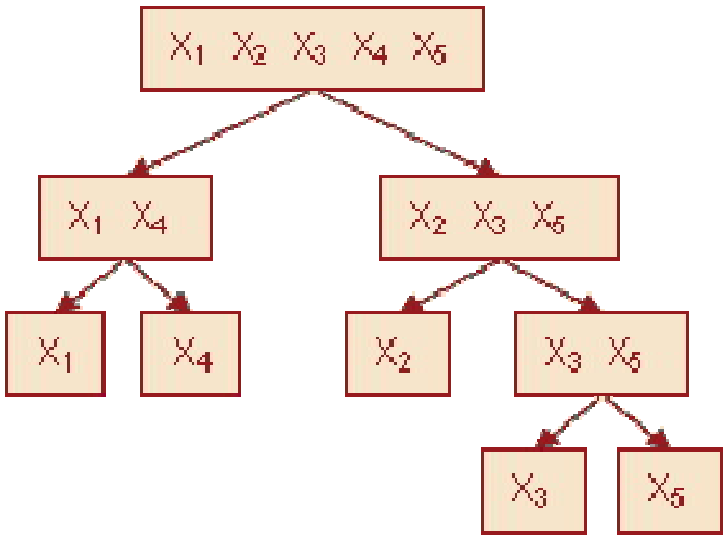

Рис. 2.1. Роздільний алгоритм кластеризації

Поширеніші висхідні або об’єднуючі алгоритми, які на початку роботи поміщають кожен об'єкт в окремий кластер, а потім об'єднують кластери у більші, поки всі об'єкти вибірки не будуть знаходитись в одному кластері. Приклад об’єднуючого алгоритму зображений на рисунку 2.2.

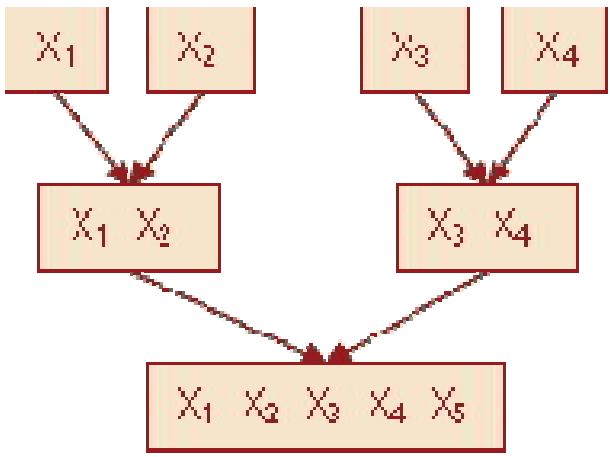

Рис. 2.2. Об’єднуючий алгоритм кластеризації

Ієрархічні алгоритми кластеризації, також відомі як алгоритми таксономії, будують не одне розбиття вибірки на непересічні класи, а систему вкладеного розбиття, що дає змогу значно спростити обробку даних та ухвалення рішень. Таким чином будується система вкладеного розбиття. Результатом таксономії є деревоподібна ієрархічна структура, яка представляється у вигляді таксономічного дерева - дендрограми, яка зображена на рисунку 2.3. При цьому об’єкт характеризується перерахунком всіх кластерів, яким він належить.

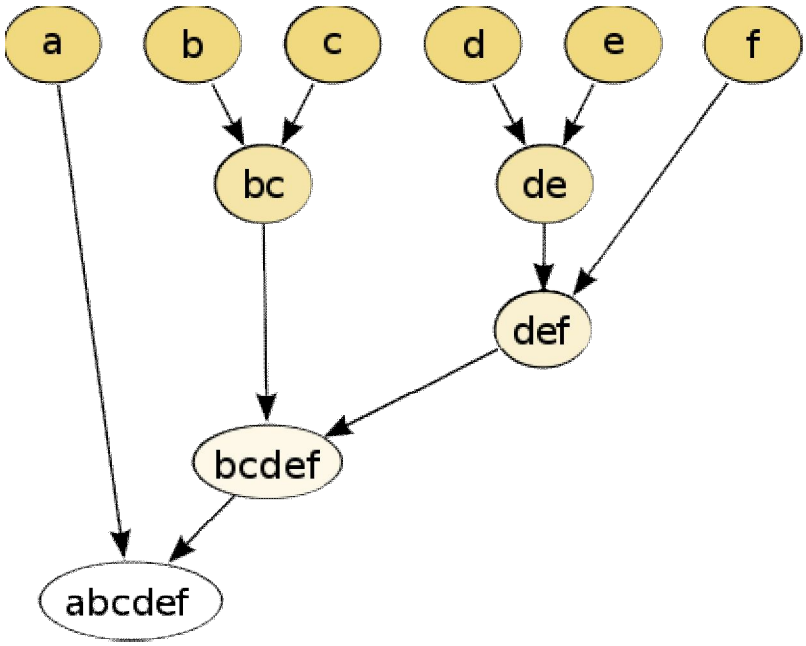

Рис. 2.3. Дендрограма

Під дендрограмою розуміють дерево, тобто граф без циклів побудований по матриці близькості. Дендрограма дозволяє представити взаємозв'язки між об'єктами з заданого переліку [12]. Для створення дендрограми потрібна матриця подібності (відмінності), яка визначає рівень подібності між парами об'єктів. Частіше використовуються агломеративні методи. Класичний приклад такого дерева – це класифікація тварин і рослин [13].

Для обчислення відстаней між кластерами все частіше користуються двома відстанями: одиночним зв'язком або повним зв'язком.

Ієрархічні методи розрізняються правилами побудови кластерів. В якості правил виступають критерії, які використовуються при вирішенні питання про "схожість" об'єктів при їх об'єднанні в групу (об’єднуючі методи) або поділу на групи (роздільні методи). Також однією з переваг ієрархічних методів кластеризації є їх наочність.

У дендрограмі об'єкти можуть розташовуватися вертикально або горизонтально. Приклад вертикальної дендрограмми наведено на рис. 2.4.

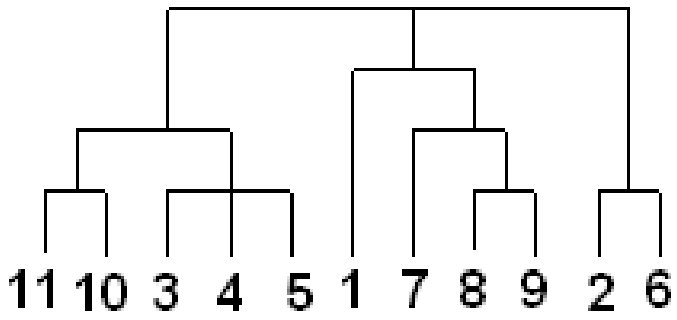

Рис. 2.4. Вертикальна дендрограма

Для побудови дендрограми, використовується наступний вираз:

$$K_\eta([i,j],k) = \left[\frac{n_i K(i,k)^\eta + (n_j K(j,k)^\eta)}{n_i + n_j}\right]^{\frac{1}{\eta}}, -\infty \le \eta \le +\infty, \tag{2.1}$$

де [i,j]— група з двох об'єктів (кластерів), (j, k)— об'єкт (кластер), з яким шукається схожість зазначеної групи; ni— кількість елементів у кластері; nj— кількість елементів у кластері j.

Як уже згадувалося, серед алгоритмів ієрархічної кластеризації розрізняють два основних методи, а саме роздільний та об’єднуючий. Перший метод розбиває вибірку на менші кластери. Другий метод, який є поширенішим, навпаки об’єднує об’єкти в більші кластери. Приклад роздільного та об’єднуючого методу зображений на рисунку 2.5.

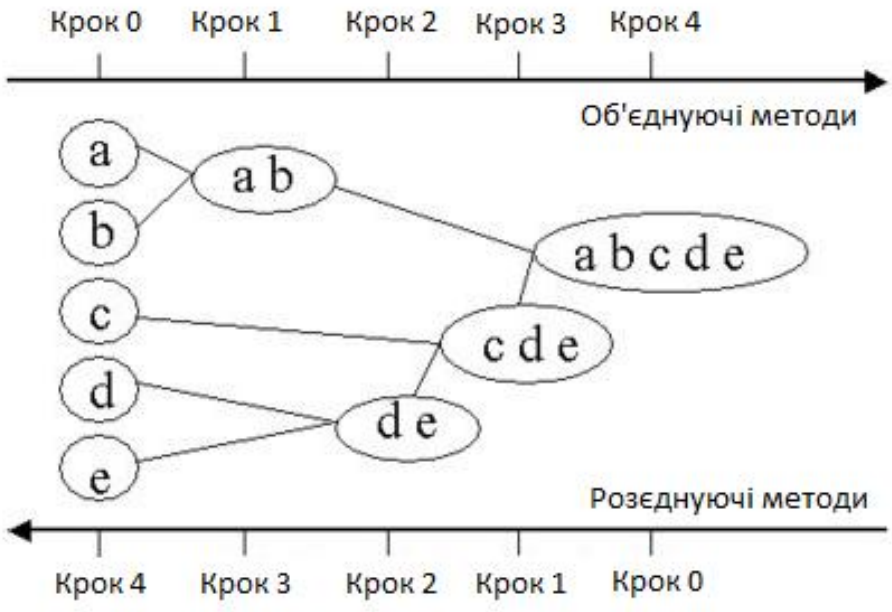

Рис. 2.5. Роздільний та об’єднуючий методи ієрархічної кластеризації

Основною задачею є поєднання вище згаданих методів в один алгоритм. Таке об’єднання дозволяє вирішити проблему ухвалення рішення, спростити та пришвидшити алгоритм пошуку обранця.

За допомогою роздільного методу буде здійснюватися пошук депутатів у вибірці. Після чого до знайденого депутата за допомогою об’єднуючого методу буде застосовано пошук за категоріями законопроектів. В результаті отримано дерево кластерів із якого в подальшому можна вибрати певний закон із потрібними категоріями.

У разі використання ієрархічних алгоритмів постає питання, як з'єднувати між собою кластери, як вираховувати «відстані» між ними. Існує кілька метрик:

1. Одиночний зв'язок (відстані найближчого сусіда). При використанні цього методу відстань між двома кластерами визначається відстанню між двома найближчими об'єктами (найближчими сусідами) у різних кластерах. Це правило у певному сенсі має нанизувати об'єкти разом для формування великого (кінцевого) кластеру, і результуючі кластери мають тенденцію бути представленими довгими "ланцюжками".

2. Повний зв'язок (відстань найвіддаленіших сусідів). У цьому методі відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами в різних кластерах (тобто найвіддаленішими сусідами). Цей метод, зазвичай, працює дуже добре, коли об'єкти походять з окремих груп. Якщо ж кластери мають видовжену форму або їх природний тип є «ланцюговим», то цей метод не підходить.

3. Незважене попарне середнє. У цьому методі відстань між двома різними кластерами обчислюється як середня відстань між усіма парами об'єктів в них. Метод ефективний, коли об'єкти формують різні групи, проте він працює однаково добре і у випадках «ланцюгового» типу кластерів.

4. Зважене попарне середнє. Метод ідентичний попередньому методу, за винятком того, що при обчисленнях розмір відповідних кластерів (тобто число об'єктів, що містяться в них) використовується як ваговий коефіцієнт. Тому цей метод можна використовувати, коли передбачаються нерівні розміри кластерів.

5. Незважений центроїдний метод. У цьому методі відстань між двома кластерами визначається як відстань між їхніми центрами ваги. Це метод незваженого попарного центроїдного усереднення.

6. Зважений центроїдний метод (медіана). Метод, який є ідентичним попередньому, за винятком того, що при обчисленнях використовується ваги для обліку різниці між розмірами кластерів (тобто кількістю об'єктів у них). Тому, якщо є (або підозрюються) значні відмінності в розмірах кластерів, цей метод виявляється прийнятнішим, ніж попередній.

7. Метод Варда. Цей метод відрізняється від всіх інших методів, оскільки він використовує методи дисперсійного аналізу для оцінки відстаней між кластерами. Метод мінімізує суму квадратів для будь-яких двох (гіпотетичних) кластерів, які можуть бути сформовані на кожному етапі кластеризації. В цілому даний метод є дуже ефективним, однак він підходящий для створення кластерів невеликого розміру [14].

2.2. Алгоритм ієрархічного підходу кластеризації ухвалення рішень

На сьогодні існує велика кількість законів та законопроектів, які зазвичай мають абсолютно різну тематику та відносяться до різних категорій. Нажаль пересічний громадянин навіть не здогадується про існування більшості з законів та їх авторів, що в подальшому може вплинути на її вибір. Запропонований алгоритм дозволить вирішити дану проблему. Він автоматично кластеризує закони та депутатів у категорії, які відповідають певній тематиці. У алгоритмі на першому етапі закони проходять попередню обробку — визначення законів для знайденого автора - депутата. Після цього здійснюється кластеризація законів по категоріям, що значно спростить їх пошук та читання користувачеві.

Припустимо, що існує певна кількість законопроектів. Наша задача згрупувати ці закони відповідно до певної категорії. Згадане групування може бути як однорівневим («плоскою», з виділенням таких кластерів, що кожен об'єкт в них є одним з текстів набору кластеризації), так і ієрархічним, коли кластери, отримані в результаті об'єднання найбільш схожих закрнів самі можуть об'єднуватися в кластери, а кластери кластерів — в інші кластери і так далі. Відношення закону до деякого кластера на певному рівні ієрархічної кластеризації може бути однозначною (кожен даний закон належить лише одному кластеру), або неоднозначною (кожен даний закон може належати декільком кластерам). Кластеризація законів була використана, аби автоматично генерувати ієрархічні кластери законопроектів.

Кластеризація законів автоматично виявляє групи семантично схожих законопроектів серед заданої великої фіксованої кількості законів.

Також варто зазначити, що групи формуються лише на основі попарної подібності описів законопроектів, і жодні характеристики цих груп не задаються заздалегідь, на відміну від класифікації [15].

Спочатку кожен об'єкт вважається окремим кластером. Для одноелементних кластерів природним чином визначається функція відстані:

$$R(\{x\},\{x'\}) = \rho(x,x'). \quad (2.2)$$

Потім запускається процес злиття. На кожній ітерації замість пари самих близьких кластерів U і V утворюється новий кластер W = U ∪ V. Відстань від нового кластера W до будь-якого іншого кластера S обчислюється по відстанях R (U, V), R (U, S) і R (V, S), які до цього моменту вже повинні бути відомі:

$$R(U \cup V,S) = \alpha_u R(U,S) + \alpha_v R(V,S) + \beta R(U,V) + \gamma|R(U,S) - R(V,S)|, \quad (2.3)$$

де α_u, α_v, β, γ - числові параметри.

Ця універсальна формула узагальнює практично всі розумні способи визначити відстань між кластерами. Вона була запропонована Лансом і Вільямсом в 1967 році. Тому алгоритм який використовується для кластеризації законів має назву «об'єднуюча кластеризація Ланса-Вільямса».

На практиці найчастіше використовуються два способи обчислення відстаней R (W, S) між кластерами W і S. Для кожного з них доведено відповідність формулою Ланса-Вільямса за певних параметрах:

- відстань найближчого сусіда:

$$R^b(W,S) = \min_{w \in W, s \in S} \rho(w,s); \qquad \alpha_u = \alpha_v = \frac{1}{2}, \beta = 0, \gamma = -\frac{1}{2}; \quad (2.4)$$

- відстань далекого сусіда:

$$R^d(W,S) = \max_{w \in W, s \in S} \rho(w,s); \qquad \alpha_u = \alpha_v = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}. \quad (2.5)$$

В нашому випадку ми скористаємося першим способом для обчислення відстаней R (W, S) між кластерами W і S.

Реалізація алгоритму об'єднання кластерів за формулою Ланса-Вільямса здійснюється наступною послідовністю кроків:

1. Ініціалізація множини кластерів C_1:

$$t := 1;\ Ct = \{x1\}, \ldots, \{x\ell\}. \tag{2.6}$$

2. Для всіх t = 2,…,l (t – номер ітерації).

3. Найти в C_{t-1} два ближніх кластери:

$$\begin{gathered}(U, V) := \arg\min_{U \neq V} R(U, V);\\ R_t := R(U, V).\end{gathered} \tag{2.7}$$

4. Видобути кластери U та V, добавити злитий кластер $W = U \cup V$:

$$C_t := C_{t-1} \cup \{W\} \setminus \{U, V\}. \tag{2.8}$$

5. Для всіх $S \in C_t$.

6. Обрахувати відстань R(W, S) по формулі Ланса-Вільямса.

Як уже не одноразово згадувалося, на початку роботи алгоритму всі об'єкти є окремими кластерами. На першому кроці найбільш схожі об'єкти об'єднуються в кластер. На наступних кроках об'єднання продовжується до тих пір, поки всі об'єкти не будуть складати один кластер.

Блок-схему з послідовністю виконання перечислених вище кроків можна побачити на рисунку 2.6.

Крім цього за допомогою роздільного алгоритму MST (Algorithm based on Minimum Spanning Trees) буде здійснюватися кластеризація депутатів, що спростить пошук депутатів по категоріях серед їх великої кількості.

Однією з переваг алгоритму MST в тому, що він виділяє кластери довільної форми, окремо кластери випуклої та опуклої форми. Після цього обирає найоптимальніше рішення.

Після проходження даних кроків ми отримаємо вибірку законів по певній категорії.

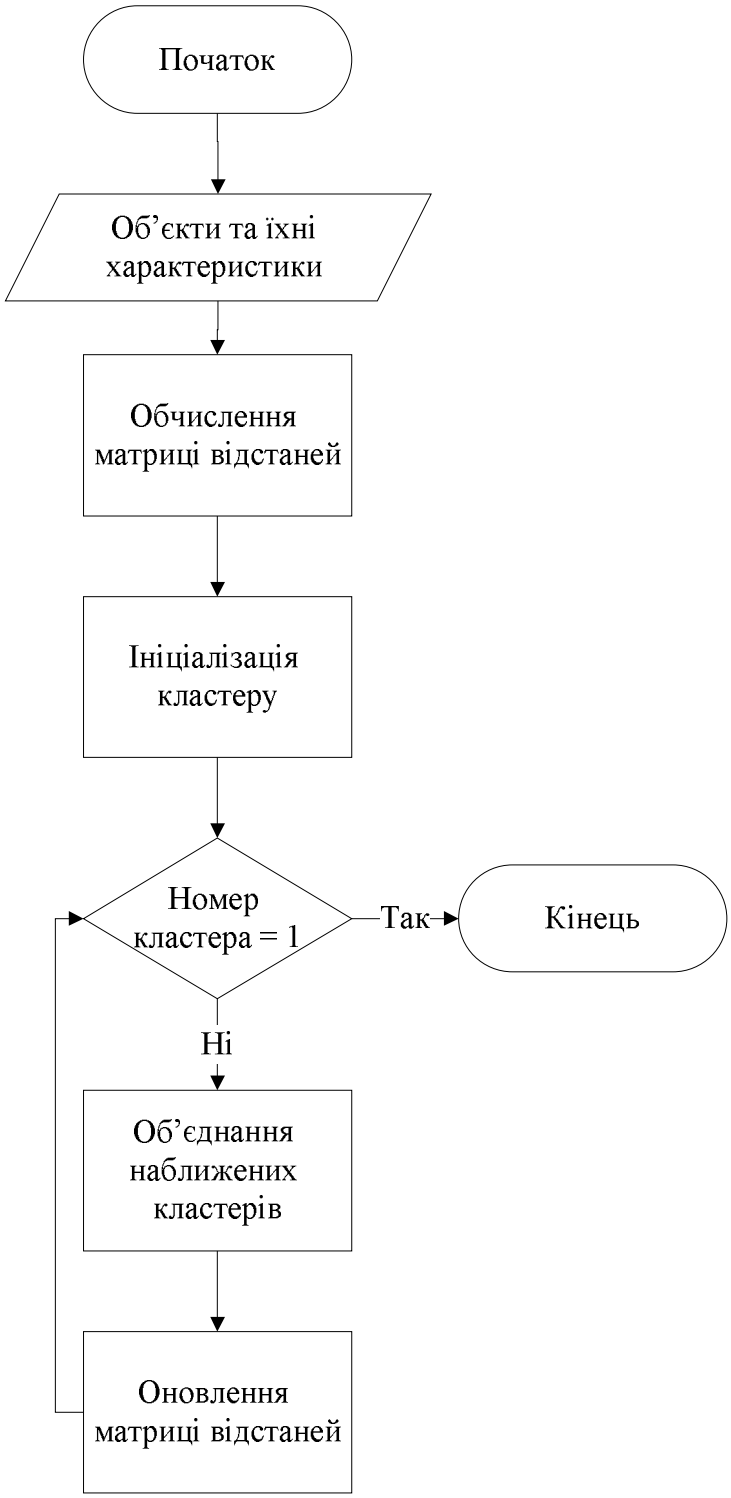

Рис. 2.6. Виконання ієрархічного алгоритму

Першим кроком даного алгоритму є побудова «мінімального кісткового дерева» (у зв'язаному, зваженому, неорієнтованому графі — це кістяк цього графа, що має мінімальну можливу вагу, де під вагою дерева розуміється сума

ваг його ребер): зв'язний, неорієнтовані граф з вагами на ребрах G(V, E), в якому V – безліч вершин (контактів), а E – безліч їх можливих попарних з'єднань (ребер), для кожного ребра (u, v) однозначно визначено деякий дійсне число w (u, v) — його вага (довжина або вартість з'єднання).

Дальше за алгоритмом Борувки проходимо наступні кроки:

- для кожної вершини графа находимо ребро з мінімальною вагою;
- додаємо знайдені ребра до кістяка, за умови їх безпеки;
- знаходимо і додаємо безпечні ребра для незв'язаних вершин до кісткового дерева;

Загальний час роботи алгоритму: O (ELogV). Останнім кроком кластеризації є поділ на кластери. Дуги з найбільшими вагами поділяють кластери.

2.3. Формалізація задачі кластеризації багатопараметричних об’єктів

Нехай в сукупності об'єктів неоднорідної природи можна виділити *n* об’єктів

$$O = \{O_1, O_2, O_n\}, \tag{2.9}$$

де кожен об’єкт розглядається в *p* просторах ознак.

Для кожного простору визначені його ознаки - характеристики об'єктів *O*. Модель багатовимірної структури представлена наступним чином:

$$\begin{gathered} O_i = \left\{ \overline{x}^i \right\}, \\ \overline{x}^i = \left(\overline{x}_1^i, \overline{x}_2^i, , \overline{x}_p^i \right), \\ \overline{x}_j^i = \left(\overline{x}_1^i, , \overline{x}_{q_j}^i \right)^T, j = \left(\overline{1, p} \right), q_j \in \mathrm{Z} \end{gathered} \tag{2.10}$$

де $\overline{x}_j^i$ - вектор величин, що визначають положення об’єкта в просторі ознак Π_j; q_j - кількість ознак в просторі Π_j.

У кожному просторі $\Pi_j, j = \overline{1, p}$ визначена метрика $\rho_j \left(O_l, O_m \right), j = \overline{1, p}$ $l, m = \overline{1, n}$, що дозволяє оцінити близькість об’єктів O_l і O_m в Π_j.

Основними труднощами кластеризації об'єктів у вигляді (2.10) є:

- приведення метрик $\rho_j(O_l, O_m), j = \overline{1, p}\ l, m = \overline{1, n}$, до стандартному виду для застосування класичних методів кластеризації;

- побудова гіпотез щодо кількості шуканих класів *k*;

- якісна оцінка розбиття.

Запропоновано результат кластеризації - розбиття S^0 оцінювати за критеріями

$$\Phi_h = \varphi\left(k, f\left(S^0\right)\right) \to \min,\ h = \overline{1, H}, \qquad (2.11)$$

де *H* - розмірність простору критеріїв; *k* - кількість кластерів у S^0; $f\left(S^0\right)$ - в певному сенсі гладкі функції.

Для вирішення задачі виділено два етапи:

- 1 етап. Багатопараметрична кластеризація.

- 2 етап. Пошук оптимального рішення в просторі критеріїв.

Для вирішення задачі багатопараметричної кластеризації запропоновано використовувати одну метрику в якості основної, а решта відносити в область експертних оцінок.

За основу був узятий алгоритм «constrained clustering», запропонований K. Wagstaff і C. Cardie для кластеризації методом k-means. У цьому алгоритмі для прийняття рішення про приєднання об’єкта до класу використовується не тільки міра подібності (метрика), але і певні обмеження, що відображають експертну оцінку можливості об’єднання об’єктів в один кластер.

Обмеження можуть бути представлені у вигляді:

- обмеження на об’єднання об’єктів за значеннями метрик;

- обмеження на характеристики утворених кластерів, що розглядаються в різних просторах ознак.

Запропоновано дворівнева схема попарного вибору об’єктів для об’єднання в кластер в процесі багатопараметричної кластеризації множини об’єктів. Блок-схема алгоритму наведена на рисунку 2.7. Позначення на

рисунку 2.7: $\rho_j, j=\overline{1,p}$ - метрики, які відповідають різним просторам ознак; $\{B_i\}=\{b_{i_{lm}}\}$ - множина обмежень; $b_{i_{lm}}=\begin{cases}1, \textit{якщо } \rho_i(O_l,O_m)\geq z_i,\\ 0, \textit{якщо } \rho_i(O_l,O_m)<z_i,\end{cases}$ $\{z_i\}$ - набір обмежень.

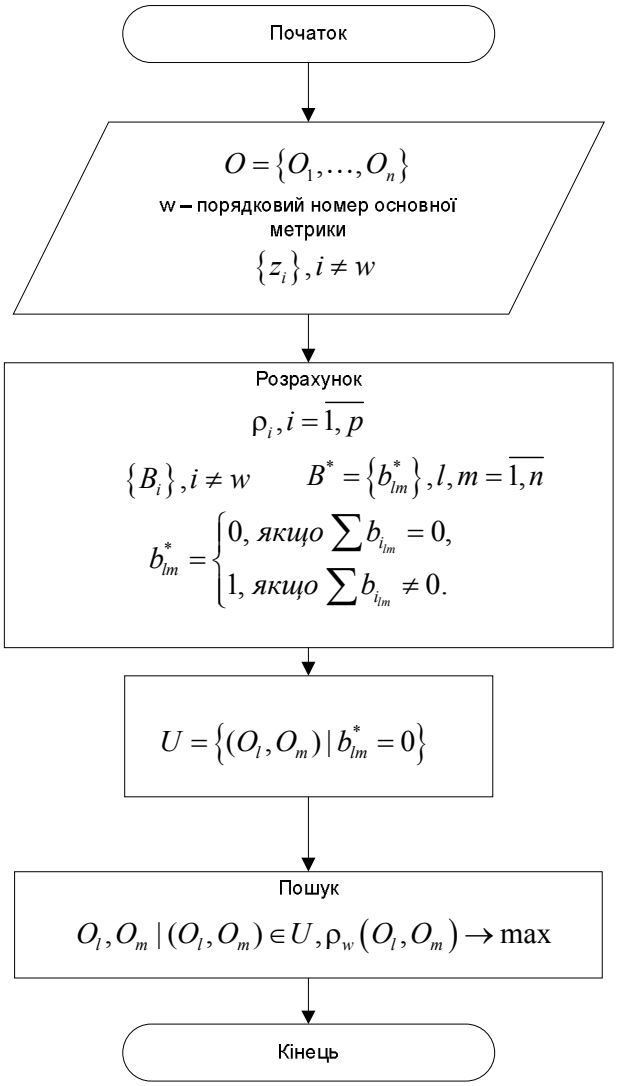

Рис. 2.7. Блок-схема алгоритму, що реалізує дворівневу схему вибору пари об’єктів для об’єднання в кластер

Запропоновано використовувати дану схему в рамках ієрархічного агломеративного алгоритму. Виявлено основні труднощі використання запропонованої схеми:

1. Складність правильного вибору обмежень при значній кількості метрик;

2. Рівнозначність отриманих розбитів при різних обмеженнях у ставленні до внутрішньої мети кластеризації;

3. Великий вплив людського фактора в дослідженні - експертна оцінка може не врахувати всіх можливих альтернатив.

2 етап. Результат S^0 кластеризації безпосередньо залежить від параметрів - $\{z_i\}, i=\overline{1,p}, i \neq w$ обмежень, при цьому результати кластеризації при різних обмеженнях рівнозначні щодо внутрішньої мети кластеризації. Однак щодо зовнішніх критеріїв

$$\Phi_h = \varphi\left(k, f\left(S^0\right)\right), h=\overline{1,H}, \qquad (2.12)$$

серед розбитів, отриманих при різних параметрах кластеризації, можна виділити найбільш бажані. Таким чином, маємо простір обмежень Z_{p-1}, де *p - 1* - кількість обмежень, що задаються експертно, $Z^0 = \left(z_1^0, z_2^0, , z_{p-1}^0\right)$ точку з відповідним розбиттям S^0 і точку $\Phi^0 = \left(\Phi_1, \Phi_2, , \Phi_н\right)$ в просторі критеріїв.

Кожна точка в просторі обмежень знаходить відповідну точку в просторі критеріїв, таким чином, рішення, отримані при різних експертних оцінках, стають порівнянні.

Оскільки критерії $\{\Phi_h\}$, $h=\overline{1,H}$, як правило, суперечливі, запропоновано використовувати діалоговий алгоритм Соболя-Статнікова для пошуку парето-ефективних розбитів.

У застосуванні до задачі кластеризації він буде складатися з наступних кроків:

1. Встановлення інтервалів значень, які можуть приймати обмеження $z_i, i=\overline{1,p-1}$, тобто визначається простір $Z^0 \in Z$ - простір допустимих установок обмежень;

2. В межах Z^0 визначаються рівномірно розподілені випадкові пробні точки;

3. Кластеризація об’єктів $O=\{O_1,O_2,O_n\}$ при різних обмеженнях, відповідних випадковим точкам. Перехід від результатів кластеризації до значень критеріїв через $\varphi\left(k_t,f\left(S_t\right)\right)$, де t - кількість пробних точок;

4. Відображення результатів в критеріальному просторі, побудова парето-ефективної межі.

В магістерському дослідженні розроблено алгоритмічне забезпечення кластеризації на основі обраного алгоритму кластеризації і методу пошуку оптимального рішення.

Нехай є множина об'єктів $\{O_i\}, i=\overline{1,n}$, яка підлягає кластеризації. Причому кожен об’єкт O_i розглядається в p просторах ознак, для кожного з яких визначена метрика ρ_p. Об’єкт представлений у вигляді (2.9).

Нехай визначено основний простір ознак Π_w, відповідно, основна метрика ρ_w, а для інших задані обмеження z_j, $j=\overline{1,p}$, $j\neq w$, і правила побудови висновків $b_{j_{lm}}=g\left(\rho_{j_{lm}},z_j\right), j=\overline{1,p}, j\neq w, l,m=\overline{1,n}$.

Алгоритм багатопараметричної кластеризації приймає наступний вигляд:

1. Ініціалізація множини некластеризованних об’єктів (кожен об’єкт є кластером) $\{C_i\}$, $C_i=O_i$, $i=\overline{1,n}$.

2. Цикл

• Розрахунок $\rho_j\left(C_l,C_m\right)$, $j=\overline{1,p}$, $l,m=\overline{1,n}$. Складання матриць $W=\left\{\rho_w\left(C_l,C_m\right)\mid l,m=\overline{1,n}, l\neq m\right\}$, $B^*=\left\{b^*_{lm}\mid l,m=\overline{1,n}, l\neq m\right\}$, де $b^*_{lm}=f\left(b_{j_{lm}}\right)\in\{1,0\}$, $j=\overline{1,p}$, $b_{j_{lm}}=g\left(\rho_{j_{lm}},z_j\right)$. Складання множини пар кластерів, які відповідають обмеженням $U=\left\{\left(C_l,C_m\right)\mid l,m=\overline{1,n}, l\neq m, b^*_{lm}=0\right\}$; Пошук пари об'єктів C_d, C_e таких що $d,e=\arg\min\left(\rho_w\left(C_l,C_m\right)\mid\left(C_l,C_m\right)\in U\right)$;

• Об'єднання C_d, C_e в один кластер C_{de}, виняток C_d, C_e з $\{C_i\}$, включення C_{de} в $\{C_i\}$. До тих пір, коли $U=\varnothing$ або $n=1$.

На рисунку 2.8 представлена блок-схема алгоритму.

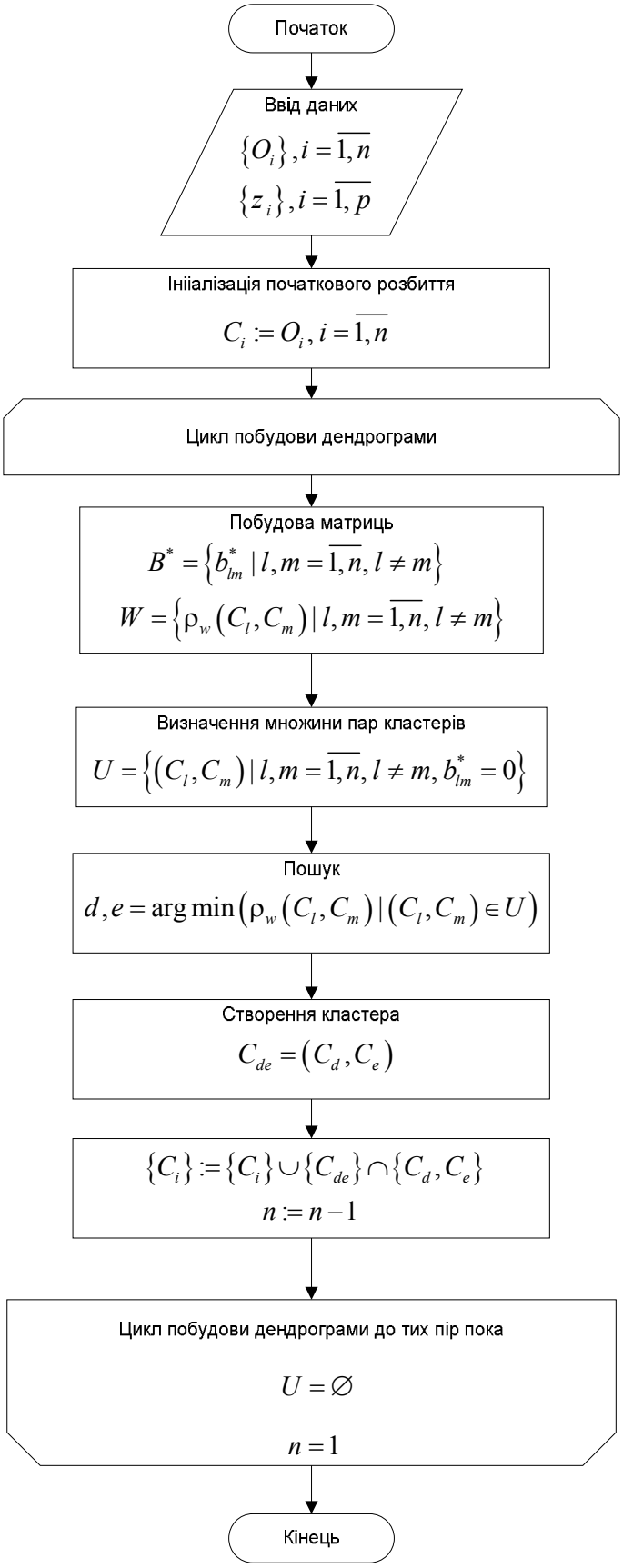

Рис. 2.8. Блок-схема алгоритму багато параметричної кластеризації

Запропоновано алгоритм пошуку парето-ефективних розв'язків задачі кластеризації об'єктів $\{O_i\}, i=\overline{1,n}$. На вході задачі маємо:

• набір інтервалів значень $\left\{\left[z_j^0; z_j^1\right]\right\}, j=\overline{1,p}$, де z_j^0 і z_j^1 - відповідно початок і кінець інтервалу;

• набір критеріїв $\Phi^h = \varphi_h\left(k_t, f\left(S^t\right)\right),, h=\overline{1,H}$, $t=\overline{1,T}$, де H - розмірність простору критеріїв, T - кількість результатів кластеризації, S^t - розбиття t-го результату, k_t - кількість кластерів в S^t.

Блок-схема алгоритму побудови парето-ефективного вирішення задачі багатопараметричної кластеризації зображена на рисунку 2.9. На рисунку 2.10 відображена рекомендована схема етапів проведення кластеризації.

Рис. 2.9. Блок-схема алгоритму пошуку Паретто-оптимального рішення

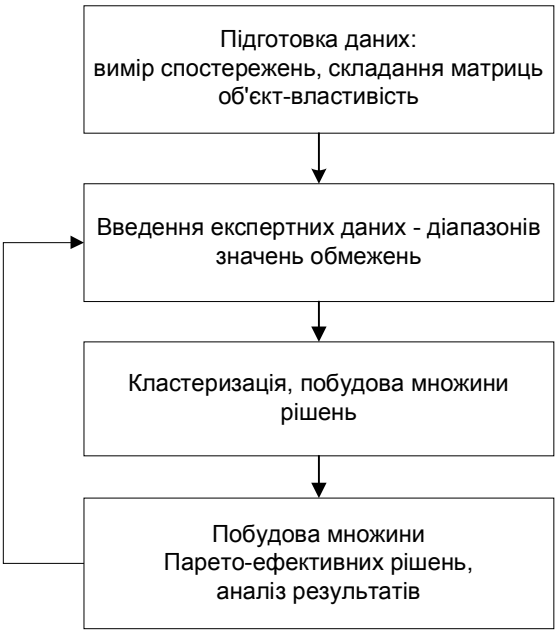

Рис. 2.10. Схема етапів проведення кластеризації

2.4. Модель роботи ієрархічної кластеризації

Першим етапом, як уже згадувалося вище із формули 2.6, є ініціалізація множини кластерів, які в нашому випадку будуть виступати законами. Основною метою є кластеризація законів серед їх великої кількості по певних критеріях. Даний крок дає нам змогу об’єднати законопроекти за категоріями, наприклад, за напрямком закону або ж за його автором.

Наступним етапом є обчислення відстані між кластерами. Існують різні варіанти для знаходження найближчого сусіда. Скористаємося формулою 2.7 для обчислення відстані найближчого сусіда.

Після цього знайденні об’єкти визначаються як кластери та об’єднуються в один кластер за формулою 2.8. Якщо ж таких об’єктів немає, то алгоритм завершує свою роботу.

Нарешті завершальним кроком буде розрахунок нової матриці відстаней за формолою 2.4 Ланса-Вільямса.

На рисунку 2.11 наведені проміжні результати роботи ієрархічного алгоритму кластеризації.

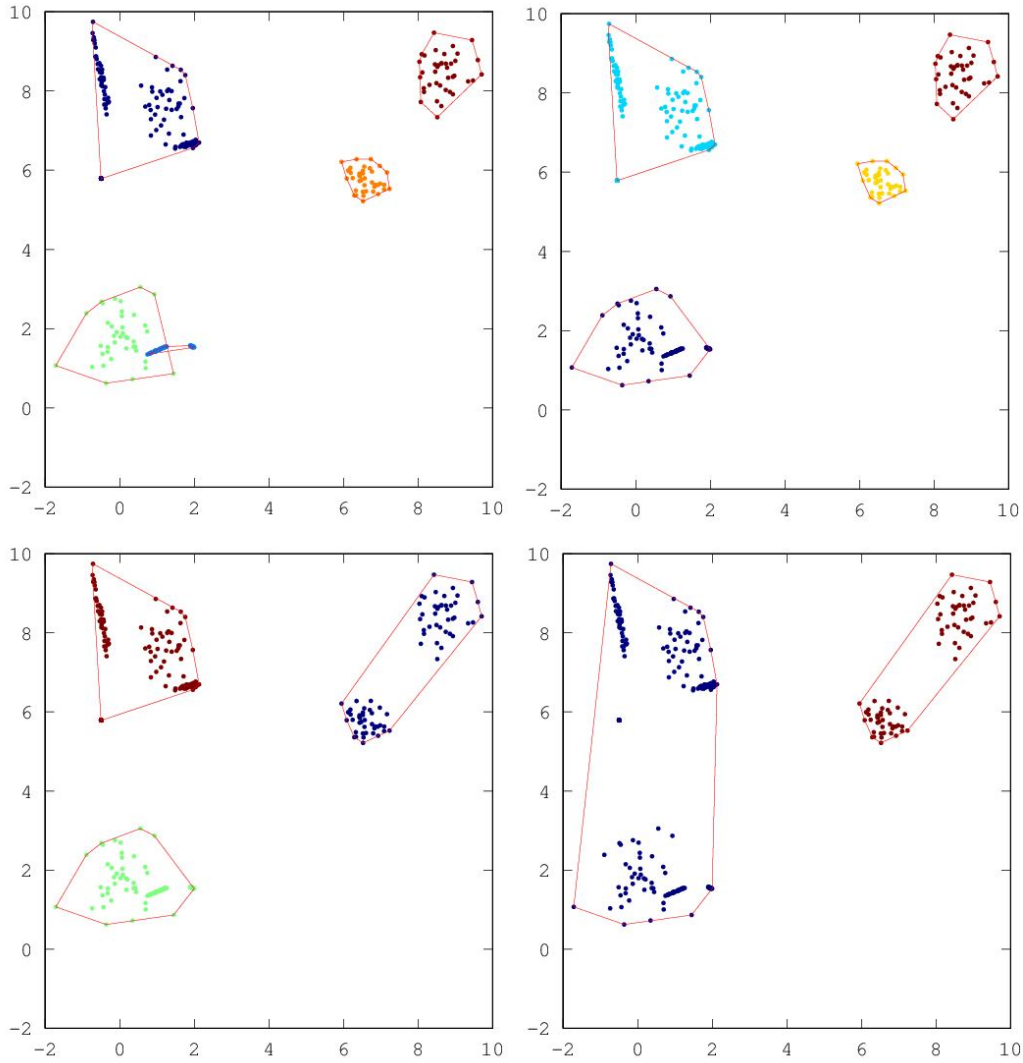

Рис. 2.11. Результат роботи ієрархічної кластеризації

Висновки до розділу 2

1. Запропоновано та сформовано ідею кластеризації процесу ухвалення рішень, шляхом ієрархічного методу кластерного аналізу.

2. Доведено, що ієрархічна кластеризація є найбільш підходящим способом для вирішення поставленої задачі.

- Обрано найоптимальніші алгоритми серед методів ієрархічної кластеризації, а саме: роздільний метод; об’єднуючий метод.

3. Розроблено його математичну модель. Продемонстровано у вигляді блок-схеми процес кластеризації.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ КЛАСТЕРИЗАЦІЇ УХВАЛЕННЯ РІШЕНЬ

3.1. Архітектура системи

Система кластеризації використовує кілька програмних компонентів, що взаємодіють по HTTP-протоколу або шляхом файлообміну по torrent-протоколу. На рисунку 3.1 представлено загальну структуру системи, а на рисунку 3.2 представлено архітектуру взаємодії програмних модулів.

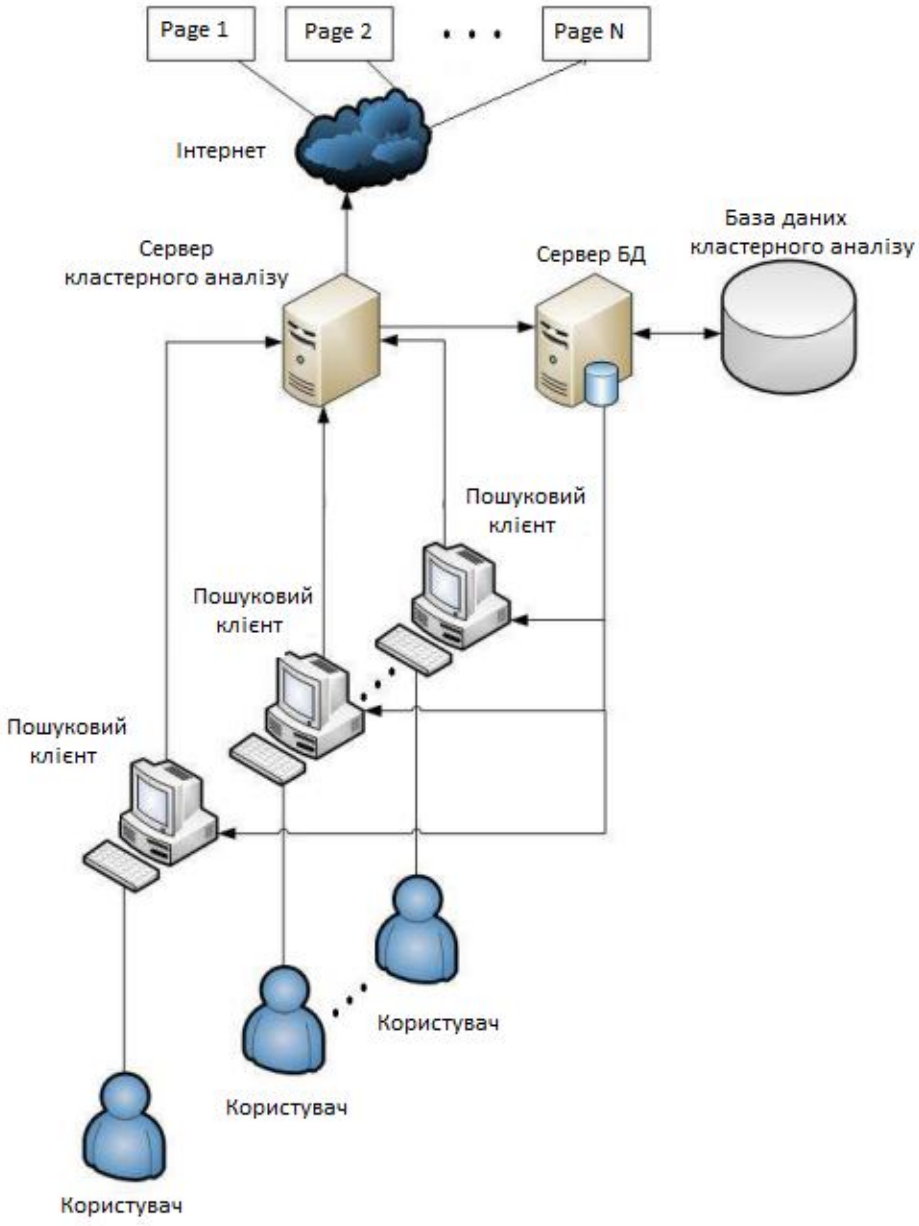

Рис. 3.1. Загальна структура системи

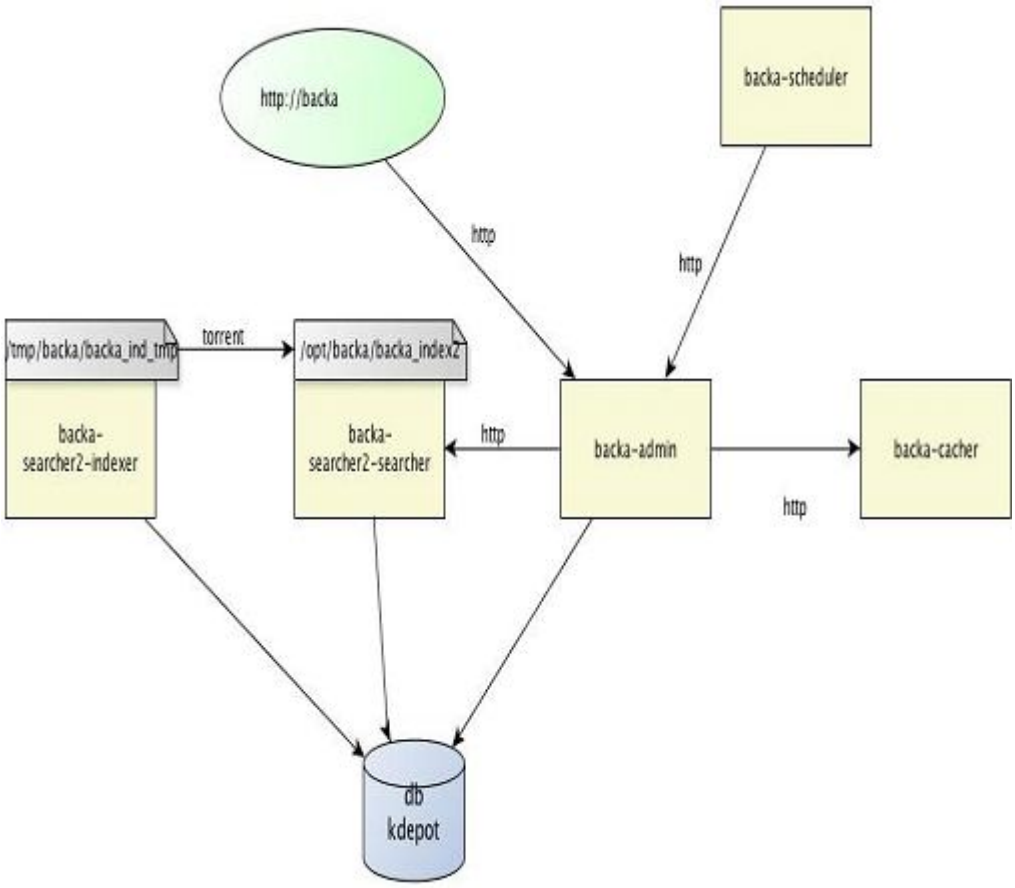

Рис. 3.2. Архітектура взаємодії програмних модулів

Кожен компонент є окремим java-додатком; різні компоненти можуть розташовуватися на різних машинах. Компоненти побудовані з використанням Spring framework. Точкою входу є сервантлет-класи, які опрацьовують HTTP-запити, що здійснюють розбір параметрів і які звертаються до класів, які реалізують логіку пошуку кандидатів і подальшої кластеризації.

Сервлети також перетворять результат в HTTP-відповідь. У таблиці 3.1. представлено опис структури та функціональне призначення модулів.

Таблиця 3.1

Опис модулів

Найменування	Участь в кластеризації	Взаємодія з іншими модулями
backa-searcher2-indexer	• Раз на добу будує повний індекс за даними про кандидатів з бази; • За допомогою torrent'ів формує Інекс в модуль backa-searcher2-searcher • tmp/ backa/ backa_ind_tmp - папка, де зберігається індекс	Взаємодіє з базою по JDBC (http://www.oracle.com/technetwork / Java / JavaSE / JDBC / в dex.html) • Передає індекс в модуль backa-searcher2-пошуковий пристрій по торрент-протоколу • Opt / Бака / backa_index2 - папка, де зберігається індекс
backa-searcher2-searcher	Будує раз в кілька хвилин інкрементальний індекс, відповідає на пошукові запити	Взаємодіє з базою по JDBC (http://www.oracle.com/technetwork / java / javase / jdbc / index.html) • Відповідає на запити backa-admin по http-протоколу
backa-admin	Отримує дані від backa-cacher • Здійснює повний процес кластеризації за запитом від scheduler'a або з інтерфейсу • Будує пошукові запити, для пошуку кандидатів • Відправляє пошукові запити в backa-searcher2-searcher • Порівнює кандидатів • Записує результати кластеризації в бд	Відправляє пошукові запити backa-searcher2-searcher'y по http-протоколу • взаємодіє з базою по JDBC • Відповідає backa-scheduler'y по http-протоколу • З backa-cacher по http-протоколу
backa-cacher	Зберігає часто використовувані дані, наприклад законопроекти, для того щоб не треба було постійно звертатися до бази	За допомогою http-запиту віддає дані в backa-admin • взаємодіє з базою по JDBC
backa-scheduler	Раз на добу запускає процес кластеризації	Задопомогою http-запиту запускає кластеризацию в backa-admin
Backa.ua	Зовнішній інтерфейс довідника депутатів, дозволяє запускати процес кластеризації	Задопомогою http-запиту запускає кластеризацию в backa-admin
Db Kdepot	База даних • зберігає дані про депутатів • зберігає результати кластеризації • зберігає інформацію про запущених і відпрацьованих процесах кластеризації	Модулі взаємодіє з базою по JDBC

3.2. Структури зберігання даних

Структура відомостей про кандидатів в депутати не є жорсткою, склад атрибутів компанії змінюється з часом, тому є необхідність зберігати дані не в звичайній реляційній базі, а в базі з нестрогой структурою.

“Key-value” сховище дозволяє зберігати дані без суворої схеми. Сутність представляє з себе набір атрибутів і їх значень. Існує багато сховищ, що підтримують таку систему (Наприклад: Apache Cassandra, Project Voldemort).

Однак, на цих же даних працює велика кількість аналітичних запитів, тому використовується “Key-value” сховище даних реалізоване на СУБД Oracle і позначене на схемі як "DB kdepot".

Таблиця, що дозволяє зберігати дані таким чином описана в запиті, що представлений на рисунку 3.3

```
CREATE TABLE tr_entity_attr (
    attr_id                          NUMBER(*,0)          PRIMARY KEY
  , entity_id                        NUMBER(*,0)          NOT NULL
  , attr_name                        VARCHAR2(125 CHAR)   NOT NULL
  , attr_npp                         NUMBER(*,0)          NOT NULL
  , attr_as_str                      VARCHAR2(950 CHAR)   NOT NULL
  , CONSTRAINT fk_tr_ea_e_01
      FOREIGN KEY(entity_id)
      REFERENCES tr_entity(entity_id)
);
```

Рис. 3.3. Відношення з даними про кандидатів в депутати

Пошук кандидатів - один з найскладніших процесів в системі. Він повинен бути максимально швидким, підтримувати пошук по логічних умовах на атрибути кандидатів, включаючи комбінацію умов по "І" та "АБО".

Пошук безпосередньо в реляційній базі при існуючих обсягах даних не дає необхідної продуктивності, крім того, він підвищує навантаження на

сховище даних, що заважає іншим процесам, взаємодіє зі сховищем (процеси збереження нових даних).

Для того, щоб забезпечити необхідну продуктивність і знизити навантаження на базу, використовується індекс і пошук кандидатів здійснюється з цього індексу.

Для побудова індексу був використаний lucene. З ним легко працювати з допомогою java (мова розробки всієї системи) і він надає можливість пошуку по складних запитах з комбінацією умов по "І" та "АБО".

Для роботи з індексом призначені два модуля:

• searcher2-indexer;

• searcher2-searcher

Усередині indexer'a відбувається повна побудова індексу і відправка цього індексу searcher'у. Індекс необхідно підтримувати в актуальному стані Для цього будуються два індекси - основний і інкрементальний.

Основний індекс будується раз на добу і відображає стан бази на початок доби. Інкрементальний індекс будується з більшою частотою (раз в декілька хвилин) на основі аналізу змін в базі, що сталися з моменту побудови основного індексу Оскільки кількість змін набагато менше загальної кількості сутностей в базі, час побудови інкрементального індексу набагато менше часу побудови повного індексу.

Компонент пошуку searcher2-searcher при пошуку використовує обидва індекси - основний і інкрементальний і список id кандидатів, які зустрічаються в обох індексах, тим самим забезпечується актуальність інформації. Процес пошуку за індексами описаний на рисунку 3.4.

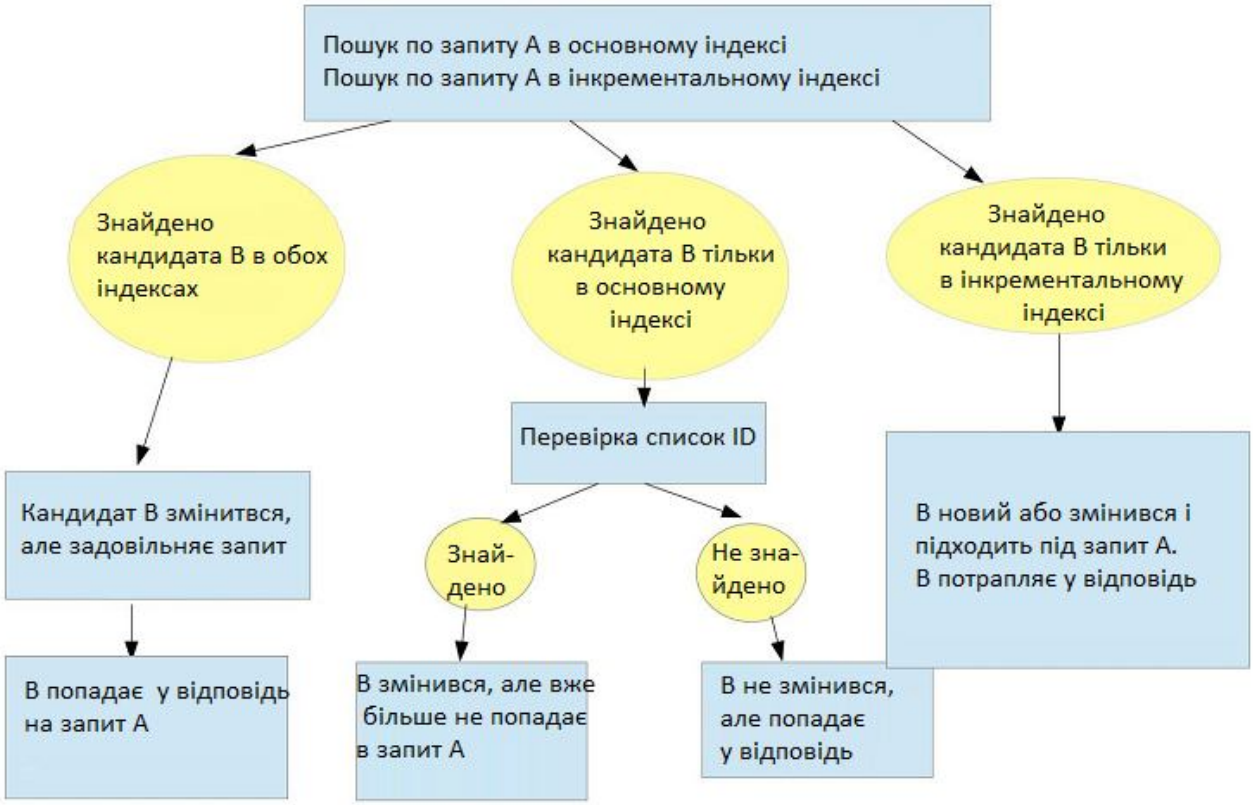

Рис. 3.4. Опис пошуку в індексах

3.3. Синхронізація двох фаз кластеризації

Кластеризація може ініціюватися користувачем системи через Web-інтерфейс модуля backa-admin або за розкладом модулем backa-sheduler, який звертається до модуля backa-admin.

Процес починається з першої фази, для чого створюється клас FirstPhase, який здійснює пошук кандидатів з допомогою індексу і передає результат - список груп кандидатів - в другу фазу.

У другій фазі здійснюється по-парне порівняння кандидатів всередині групи. Сам процес порівняння описаний в класі PairReducer. При порівнянні кандидатів враховуються такі атрибути: особисті дані кандидата, телефон, url, адреса. Варто зазначити, що для різних депутатів структура атрибутів може відрізнятися, але в нього завжди входять особисті дані та дані результативності роботи, тому необходіма гнучка система для роботи з атрибутами.

Порівняння кндидатів має складну структуру, так як його треба проводити по декількох полях. Для кожного атрибуту було реалізовано кілька алгоритмів порівняння:

• за допомогою ваг (порівняння різних полів з присвоєнням різних ваг. Якщо сума всіх ваг перевищує порогове значення, то кандидати вважаються однорідними). У прикладі, який описано на рисунку 3.5 описана функція порівняння з вагами.

```
public double distance(final Aliaser<String, String> aliaser, @NotNull final
QEntity o1, @NotNull final QEntity o2, final HistoryCollector
historyCollector) {

        int result = 0;

        if (NAME_CONTAINS_EXCEP.distance(o1, o2) < TRESHOLD) return
TOTALLY_DIFFERENT;

       if (getNameWithRemovedRubricMetricFunction(rubricProvider,
rubricatorCache).distance(o1, o2) > TRESHOLD) {

            result++;

            if (TR_URL.distance(o1, o2) > TRESHOLD) result++;

        } else {

            if (URL.distance(o1, o2) > TRESHOLD) result++;

        }

        if (PHONE.distance(o1, o2) > TRESHOLD) {

            result++;

        }

        if (RUBRIC.distance(o1, o2) > TRESHOLD) {

            result++;

        }

        if (getAddressDecisionTree().resolve(o1, o2)) result += 2;

        return (result >= 5) ? 0.0 : TOTALLY_DIFFERENT;

    }
```

Рис. 3.5. Функція порівняння з вагами

У персональних даних ваги більше, ніж у інших полів, так як в ході статистичних досліджень на даних стало ясно, що кандидати з подібними особистими даними частіше бувають дублями.

За допомогою дерева "прийняття рішень" (Desicion Tree) будується дерево, в вершинах якого знаходяться операції порівняння. Залежно від відстані між полями, відбувається перехід по правому або лівому ребру. У листі дерева - результати порівняння. У прикладі, що на рисунку 3.6 описана функція порівняння з деревом прийняття рішень.

```
private DecisionTree<QEntity, Boolean> getDecisionTree(final RubricProvider
rubricProvider, final RubricatorCache rubricatorCache,
                                                       final
QEntityFeatureService entityFeatureService, final FeatureValueService
featureValueService) {
        return node(SYSTEM_ATTR, TRESHOLD, NO,
                node(getNameWithRemovedRubricMetricFunction(rubricProvider,
rubricatorCache), TRESHOLD, NO,
                        node(PHONE, TRESHOLD, node(RUBRIC, TRESHOLD, NO,
                                node(LOCALITY_TR, TRESHOLD, NO,
                                        node(SIDESTREET_TR, TRESHOLD, NO,
                                                node(STREET_TR, TRESHOLD, NO,
                                                        node(BUILDING_NUM,
TRESHOLD,
                                                                node(SUPPLIER
_COORDINATE, TRESHOLD, NO, YES),
                                                                YES)
                                                )
                                        )
                                )
                        ),
                                node(RUBRIC, TRESHOLD, NO,
                                        node(LOCALITY_TR, TRESHOLD, NO,
                                                node(SIDESTREET_TR, TRESHOLD,
NO,
                                                        node(STREET_TR,
TRESHOLD, NO,
                                                                node(SUPPLIER
_COORDINATE, TRESHOLD, NO, YES),
                                                                YES)

                                                )
                                        )
                                )
                        )
                )
        );
    }
```

Рис. 3.6. Функція порівняння з деревом прийняття рішень

Для підрахунку відстані між полями використовуються n-грами. Алгоритм порівняння полягає в наступному:

• з рядків викидаються слова, довжина яких становить менше 2;

• рядки разом з пробілами та іншими знаками пунктуації розбиваються на 2-грами;

• підраховується кількість загальних 2-грам і ділиться на суму для двох рядків;

• теж проробляється для 3-грам;

• і 4-ох грам;

• отримані значення множаться на коефіцієнти і сумуються;

• результат порівнюється з граничним значенням.

Для визначення ваги і порогових значень в дереві було використано машинне навчання з учителем на навчальній вибірці. Для деяких кандидатів крім особистих даних було вказано точну url-адерсу. Це дозволило в різних випадках застосовувати різні підходи до порівняння.

• стандартний (обробляється як рядок);

• по url адресі.

При обох цих підходах інші поля порівнюються як зазвичай.

3.4. Збереження результатів кластеризації

Кластери зберегаються у відношені, яке представлено на рисунку 3.7.

```
create table tr_du_cluster (
        session_id                number(22,0)
      , cluster_id                number(22,0)
      , company_id                number(22,0)
      , cluster_size              number(10,0)
      ,IS_BAD                     NUMBER(1,0) DEFAULT 0
      ,IS_MERGE                   NUMBER(1,0) DEFAULT 1
      ,IS_TASK                    NUMBER(1,0) DEFAULT 0
      , CONSTRAINT pk_tr_cluster
       PRIMARY KEY (session_id, company_id)
      , CONSTRAINT fk_tr_du_cluster_session_id
      FOREIGN KEY (session_id)
       REFERENCES tr_du_session(id)
);
```

Рис. 3.7. Відношення з даними про кластери

Якщо дві пари кандидатів перетинаються, то вони об'єднуються в один кластер. Процес кластеризації запускається регулярно і шукає дублі для нових і змінених даних кандидатів. Такий процес називається сесією кластеризації. Інформація про сесії зберігається у відношені, описаному на рисунку 3.8.

```
create table tr_du_session (
      id                        number(22,0)                    primary key
       , start_time               date
      , finish_time              date
      , rule_set_id              number(22,0)
      , bulker_session_id        number(22,0)
      , status                   varchar2(256 char)
      , descr                    varchar2(512 char)
      , region_id                number(22,0)
      , process_id               number
);
```

Рис. 3.8. Відношення з даними про сесії кластеризації

Під час процесу кластеризації відбувається багато роботи з базою. Всі java класи, що відповідають за це повинні легко налаштовуватися під конкретне сховище, використовується технологія Spring. Вона дозволяє налаштовувати java класи в спеціальних конфігураційних файлах. У наступному прикладі описані кілька бінів для роботи з базою:

```
<bean name="auditTrService"
class="ua.rada.common.kdepotng.ext.audit.AuditServiceImpl">
      <property name="kdepot" ref="trKdepot"/>
      <property name="schemaPrefix" value="tr_"/>
   </bean>
   <bean name="auditTrSearchHelper"
class="="ua.rada.common.kdepotng.ext.audit.AuditSearchHelper">
      <property name="kdepot" ref="trKdepot"/>
      <property name="auditService" ref="auditTrService"/>
   </bean>
   <bean id="trKdepot"
class="="ua.rada.common.kdepotng.impl.DefaultService">
      <property name="schemaPrefix" value="tr_"/>
   </bean>
```

Так як результати сесій кластеризації для різних кандидатів не використовуються одночасно, то дані про сесії будемо зберігати в префіксних сховищах. Це дозволить винести абсолютно всю роботу з базою в різні сховища для різних кандидатів і легко налаштовувати всі java-класи для роботи з базою.

3.5. Використання зовнішнього API для отримання даних про результативність роботи депутатів

Для початкового налаштування роботи системи, а саме для наповнення початковими даними про результативність роботи депутатів використовується зовнішня API для доступу до системи «Рада». Це дозволить отримати дані для подальшої кластеризації та надання рекомендацій стосовно вибору того чи іншого кандидата. Розглянемо основні методи та параметри для використання вказаної API.

Перше, що тобі потрібно, це ключ. Ключ використовується тільки для особистих цілей. Ключ отримується автоматично, якщо авторизуватися в системі. Після виконання запиту, отримуємо інформацію в JSON форматі.

Для отримання всіх депутатів, які є в парламенті необхідно виконати наступний запит

```
GET https://rada4you.org/api/v1/people.json?key=[api_key]
```

Цей запит надає базову інформацію про кожного народного депутата, який наразі є членом парламенту. Він містить їхні імена, спосіб обрання, партію.

Щоб отримати більш детальну інформацію про депутата, використовується id і виконується наступний запит:

```
GET https://rada4you.org/api/v1/people/[id].json?key=[api_key]
```

який видасть всю корисну і деталізовану інформацію, яка представлена на рисунку 3.9.

Параметр	Опис
`rebellions`	Кількість разів, коли вони голосували проти лінії фракції
`votes_attended`	Загальну кількість голосувань депутата
`votes_possible`	Кількість можливих голосувань, де вони могли б голосувати
`offices`	не використовується
`policy_comparisons`	Сукупність політик, за який депутат міг голосувати і їх підрахований `agreement` бал в проміжку від 0 до 100. `voted` показує, чи вони колись голосували за законопроект з цієї політики.

Рис. 3.9. Деталі щодо депутатів

Наступний запит видає базову інформацію про політики, включаючи параметри, які представлені на рисунку 3.10.

```
GET https://rada4you.org/api/v1/policies.json?key=[api_key]
```

Параметр	Опис
`id`	Унікальний визначник для політики. Використовуй `id`, щоб отримати більше інформації про цю політику.
`name`	Коротка назва політики
`description`	Більше деталей про цю політику
`provisional`	`true` або `false`. Це проект політики, яка ще не завершена і не відображається за замовчуванням.

Рис. 3.10. Всі політики

Наступний запит видає всі види корисної деталізованої інформації, що включають основні характеристики, які представлені на рисунку 3.11.

```
GET https://rada4you.org/api/v1/policies/[id].json?key=[api_key]
```

Для отриманні інформації про всі голосування необхідно виконати наступний запит. Цей запит повертає базову інформацію про останні 250 голосувань, що включають характеристики, які детально описано на рисунку 3.12.

```
GET https://rada4you.org/api/v1/divisions.json?key=[api_key]
```

Параметр	Опис
`id`	Унікальний визначник для голосування. Використовуй `id`, щоб отримати більше інформації про політику
`name`	Коротка назва політики
`description`	Більше інформації про цю політику, що означає політика
`provisional`	`true` або `false`. Це проект політики, яка ще не завершена і не відображається за замовчуванням.
`policy_divisions`	Сукупність голосувань, які включені до цієї політики. Кожне голосування має відповідне `vote`, яке може бути `strong` і робить голосування дуже важливим
`people_comparisons`	Сукупність політик, за який депутат міг голосувати і їх підрахований `agreement` бал в проміжку від 0 до 100. `voted` показує, чи вони колись голосували за законопроект з цієї політики.

Рис. 3.11. Всі політики

Параметр	Опис
`id`	Унікальний визначник для голосування. Використовуй `id`, щоб отримати більше інформації про це голосування
`house`	Незмінний параметр `rada`
`name`	Скорочена назва
`date`	Дата в форматі `yyyy-mm-dd`
`number`	Перше голосування в визначений день. Кожне наступне голосування пронумеровано відповідно
`clock_time`	Час голосування в форматі `hh:mm AM` або `hh:mm PM`, чи `null`, якщо недоступний
`aye_votes`	Кількість депутатів, які проголосували “ЗА”
`no_votes`	Кількість депутатів, які проголосували “ПРОТИ”
`possible_turnout`	Кількість діючих депутатів на момент голосування
`rebellions`	Загальна кількість голосів, які проти лінії фракції
`edited`	`true`, якщо опис голосування був відредагований

Рис. 3.12. Інформація про всі голосування

Щоб отримати більше результатів та голосувань за певний проміжок часу використовується запит наступного виду

```
GET https://rada4you.org/api/v1/divisions.json?end_date=2014-09-01&house=rada&start_date=2014-08-01&
```

Цей запит видасть не більше 250 результатів голосувань(це обмеження API). Для того щоб переконатися, що отримано всі очікувані дані. На практиці, якщо отримано 250 результатів голосувань, то необхідно звузити часовий проміжок запиту.

Для отримання деталізованої інформації про результати голосувань необхідно виконати наступний запит, який поевертає всю корисну інформацію, що включає параметри, які описано на рисунку 3.13.

```
GET https://rada4you.org/api/v1/divisions/[id].json?key=[api_key]
```

Параметр	Опис
`id`	Унікальний визначник для голосування. Використовуй `id` , щоб отримати більше інформації про це голосування
`house`	не використовується
`name`	Скорочена назва
`date`	Дата в форматі `yyyy-mm-dd`
`number`	Перше голосування в визначений день. Кожне наступне голосування пронумеровано відповідно.
`clock_time`	Час голосування в форматі `hh:mm AM` або `hh:mm PM`, чи `null`, якщо недоступний
`aye_votes`	Кількість депутатів, які проголосували “ЗА”
`no_votes`	Кількість депутатів, які проголосували “ПРОТИ”
`possible_turnout`	Кількість діючих депутатів на момент голосування
`rebellions`	Загальна сума голосів, яка проти ліній фракцій
`edited`	`true`, якщо опис голосування був відредагований
`summary`	Якщо `edited` є `true`, тоді це остання версія опису. Вона відформатована в Markdown
`votes`	Сукупність голосів учасників голосування. Відображаються тільки ті депутати, які брали участь в цьому голосуванні
`policy_divisions`	Сукупність політик, до яких підв'язане голосування з вказанням підтримки чи непідтримки політики ("Так", "Так тверде", "Ні", "Ні тверде")
`bills`	Сукупність законопроектів, які пов'язані з голосуванням

Рис. 3.13. Інформація про деталі щодо голосування

3.6. Проектування та реалізація основни інтерфейсів

Розглянемо основні форми та сторінки взаємодії користувачів із системою. Дана система дозволяє здійснити аналіз роботи того чи іншого депутата протягом його каденції і відповідно сформувати думку про результативність його роботи.

На рисунку 3.14 представлена сторінка з інформацією про народних депутатів відсортованих по їх територіальній приналежності.

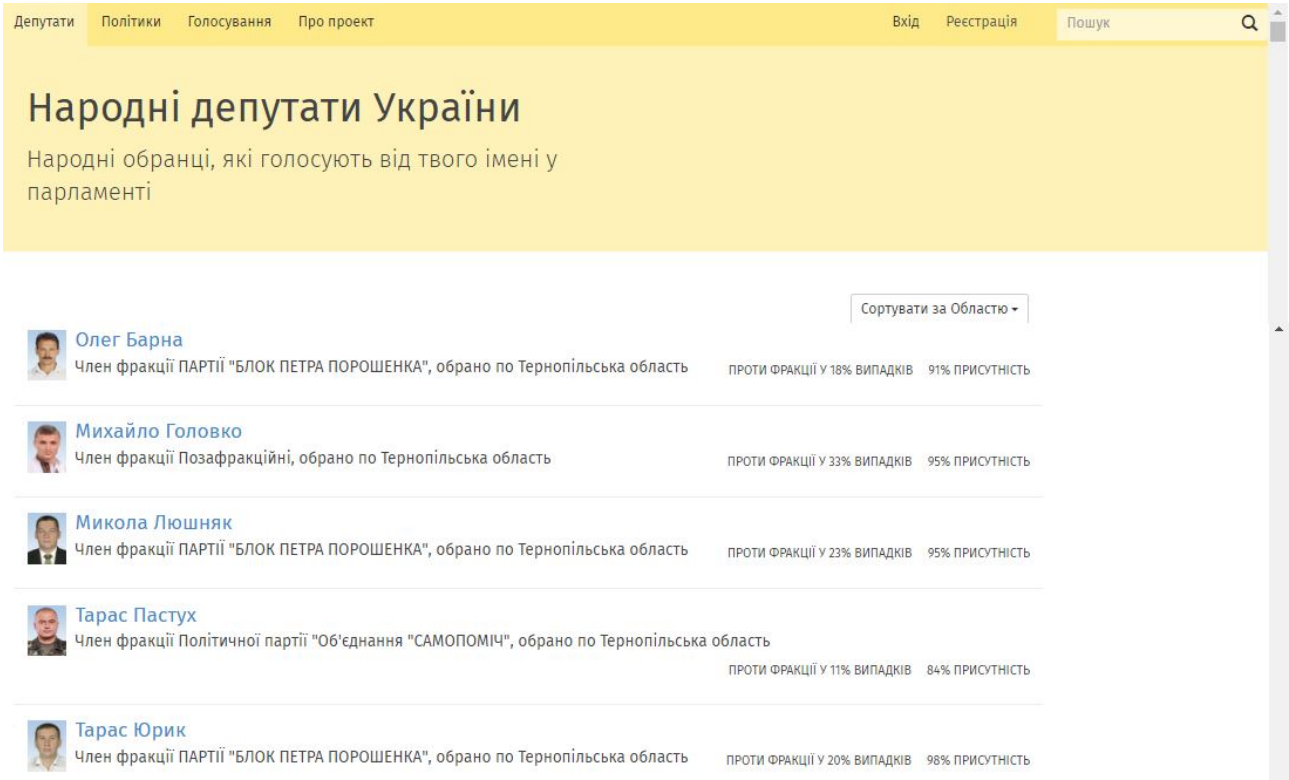

Рис. 3.14. Інформація про народних депутатів України

В системі передбачена можливість отримання детальної інформації про законопроекти в певних сферах, які найбільше цікавлять користувачів. Також можна дізнатися як голосує той чи інший депутат за відповідні законопроекти.
На рисунку 3.15 представлено інформацію про політики. Політики - це перелік голосувань за законопроект/пакет пов’язаних законопроектів, які у сукупності впроваджують у дію конкретну позицію з певного питання.

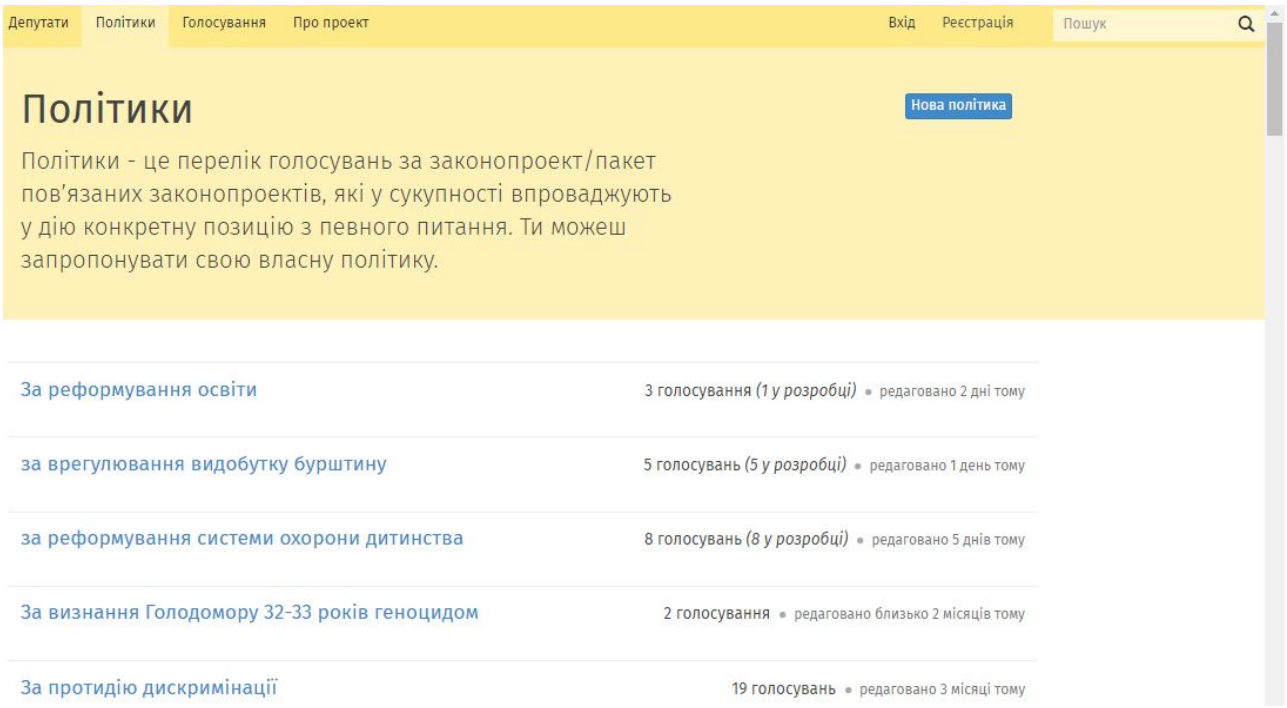

Рис. 3.15. Інформація про політики

Також в системі можна переглянути чи послідовно голосував депутат за законопроекти, які пов'язані між собою і спрямовані на вирішення тих або інших питань. На рисунку 3.16 представлено інформацію про голосування за посилення заходів з кібербезпеки.

Рис. 3.15. Інформація про послідовність голосувань

На рисунку 3.16 представлено інформацію про результативність голосування в розрізі парламентських фракцій.

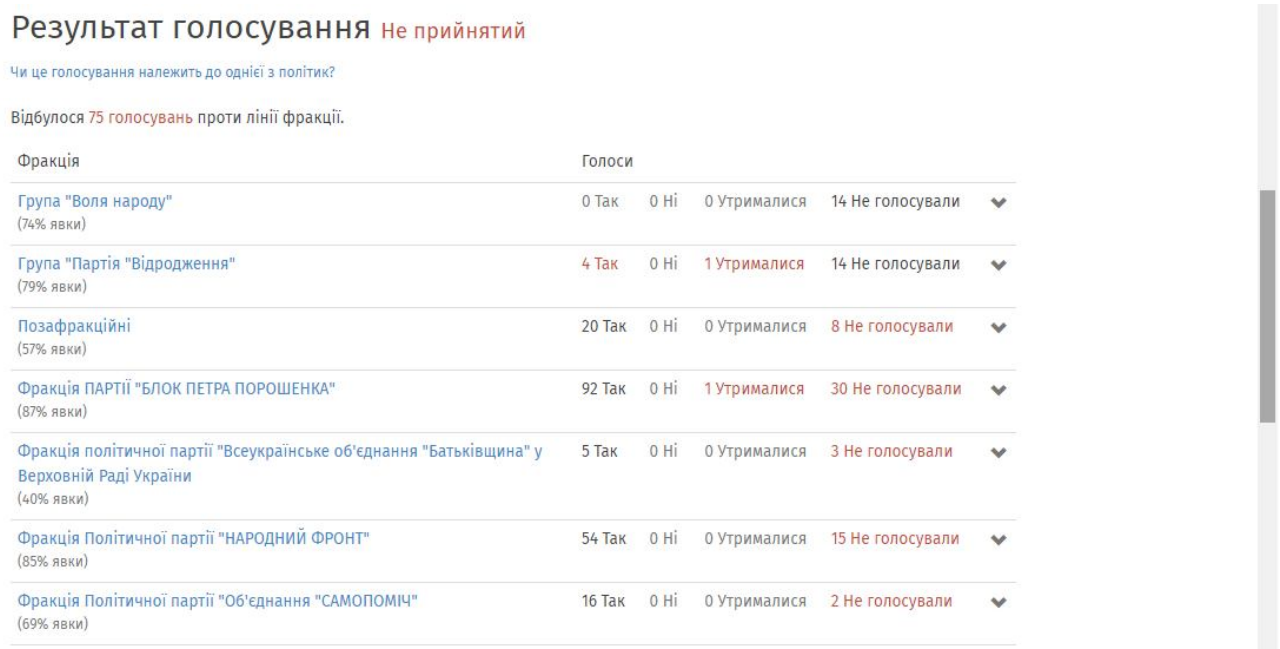

Рис. 3.16. Інформація про голосування в розрізі фракцій

Використовуючи отриману та накопичену інформацію, можна перейти до процедури кластеризації і подальшого відображення результатів у зручній для користувача формі.

3.7. Експериментальні дослідження та оцінка результатів

В ході виконання магістерської роботи розв’язана задача кластеризації кандидатів в депутати залежно від результативності їх роботи з метою ухвалення правильного вибору. Як множина об'єктів розглядаються прийняті законопроекти з тієї чи іншої сфери, яка найбільше цікавить користувача. Для здійснення правильного вибору, при оцінці великої кількості законопроектів з кількістю $n>100$ вкрай незручно враховувати особливості кожного для здійснення результативної діяльності (важливість законопроекту, його впровадження, лінійки законопроектів з урахуванням особливостей

зовнішнього середовища). В даному випадку природно виділити групи об'єктів, подібних за своїми характеристиками, і управляти групами, проектуючи згодом результати на конкретні законопроекти, а згодом і оцінка результативності роботи депутата.

Обрані найбільш значущі характеристики: - співвідношення важливості за категоріями, - динаміка імплементації, - результативність прийняття. В якості основної метрики обрана - метрика, відповідна характеристиці. У таблиці 3.1 представлені вхідні параметри алгоритму кластеризації депутатів та законопроектів.

Таблиця 3.1

Параметри алгоритма кластеризації

Вихідні дані	Формула
Множина об’єктів досліджуваної сукупності	$O = \{O_i\},\ i = \overline{1, N}$, де N – кількість об’єкті.
Модель об’єкта досліджуваної множини	$O_i = \{X_1^i, X_2^i, X_3^i\},\ X^i = \{X_1^i, X_2^i, X_3^i\}$ $X_1^i = \{x_{1_1}^i, \ldots, x_{1_W}^i\},\ X_2^i = \{x_{1_1}^i, \ldots, x_{1_D}^i\},\ X_3^i = \{x_{1_1}^i, \ldots, x_{1_{SQ}}^i\}$
Параметри подібності для характеристик X_1, X_2, X_3	$\rho_1(X_l, X_m) = \sqrt{\sum_{w=1}^{W}(x_l^w - x_m^w)^2}$, $\rho_2(X_l, X_m) = \frac{\sum_{j=1}^{D}(x_l^j - \overline{x}_l)(x_m^j - \overline{x}_m)}{\sqrt{\sum_{j=1}^{D}(x_l^j - \overline{x}_l)^2 \sum_{i=1}^{D}(x_m^j - \overline{x}_m)^2}}$, $\rho_3(X_l, X_m) = \lvert x_l^s - x_m^s \rvert$, де $l, m = \overline{1, N}$; W – кількість елементів, які утворюють характеристику X_1; D – кількість елементів, які утворюють характеристику X_2; SQ – кількість елементів, які утворюють характеристику X_3.
Діапазони значень обмежень на метрики ρ_2, ρ_3	$[z_2^0; z_2^1], [z_3^0; z_3^1]$
Правила обчислення критеріїв ефективності	$\Phi_{затр_t} = K_t \times CST$, $\Phi_{якості_t} = \frac{F_{0_t}}{F}$, де $F = \sum_{i=1}^{N} f(O_i)$, $F_{0_t} = \sum_{i=1}^{K_t} f(S_t)$, S_t – розподіл множини $\{O_i\}, i = \overline{1, N}$ на кластери; K_t – кількість кластерів в S_t, $t = \overline{1, T}$; T – кількість пробних точок.

Основними критеріями, важливими з точки зору результативності роботи депутатів, є кількість поданих законопроектів, прийняття і якість їх імплементації.

Якість імплементації проектів прямо залежать від кількості кластерів - об'єктів управління.

$$\Phi_t = K_t \times CST \to \min,$$

де CST – кількість прийнятих законопроектів певної тематики одним депутатом; K_t - кількість кластерів, і, відповідно, кількість об’єктів управління;

Якість управління виражено у вигляді імплементації законопроектів при використанні кластерів в порівнянні з вихідною ситуацією, коли об'єктом управління є кожен законопроект:

$$\Phi_{якості_t} = \frac{F_{0_t}}{F} \to \min \ ,$$

де величина F - аналіз законопроектів за відсутності групування об'єктів по кластерах - постійна і не змінюється в залежності від t,

$$F_{0_t} = \sum_{i=1}^{K_t} f(S_t)$$

Приведемо опис інструментальних засобів кластеризації законопроектів. Описано основні функції програми (рисунки 3.17-3.18). Перелічені основні користувачі.

Проведено апробацію алгоритму: ініціалізація інтервалів значень обмежень, вибір пробних точок, кластеризація при різних значеннях обмежень, побудова парето-ефективного рішення.

На рисунку 3.18 результат роботи алгоритму представлений в просторі критеріїв. Парето-ефективні точки з’єднані кривою. На рисунку 3.19 множина парето-ефективних рішень відображено в просторі обмежень.

На рисунку 3.20 відображені розрахункові показники ефективності впровадження модуля кластеризації в систему підтримки прийняття рішень при ухваленні рішень.

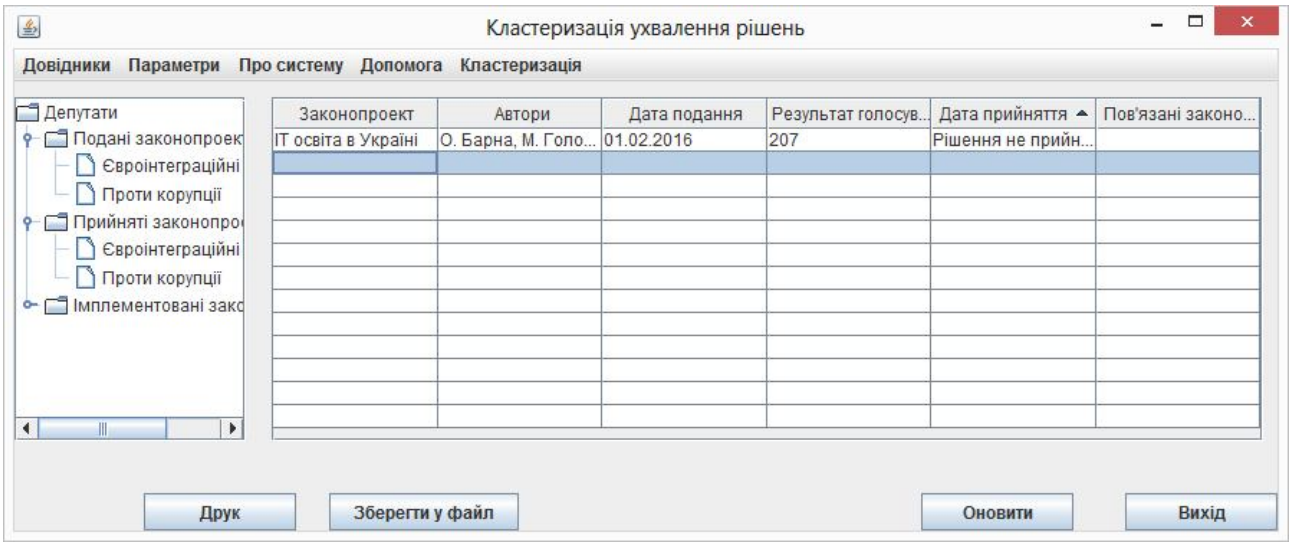

Рис. 3.17. Основна форма системи кластеризації ухвалення рішень

Рис. 3.18. Форма з результатами кластеризації

Завдяки використанню модуля кластеризації при різних якісних показниках досягається економія витрат на аналіз інформації від 5 до 70%, зниження завантаження інформаційних систем (кількість оброблюваних і

збережених даних) від 98 до 9%, скорочення термінів підготовки звітів оцінки результативності роботи депутатів від 97 до 39%.

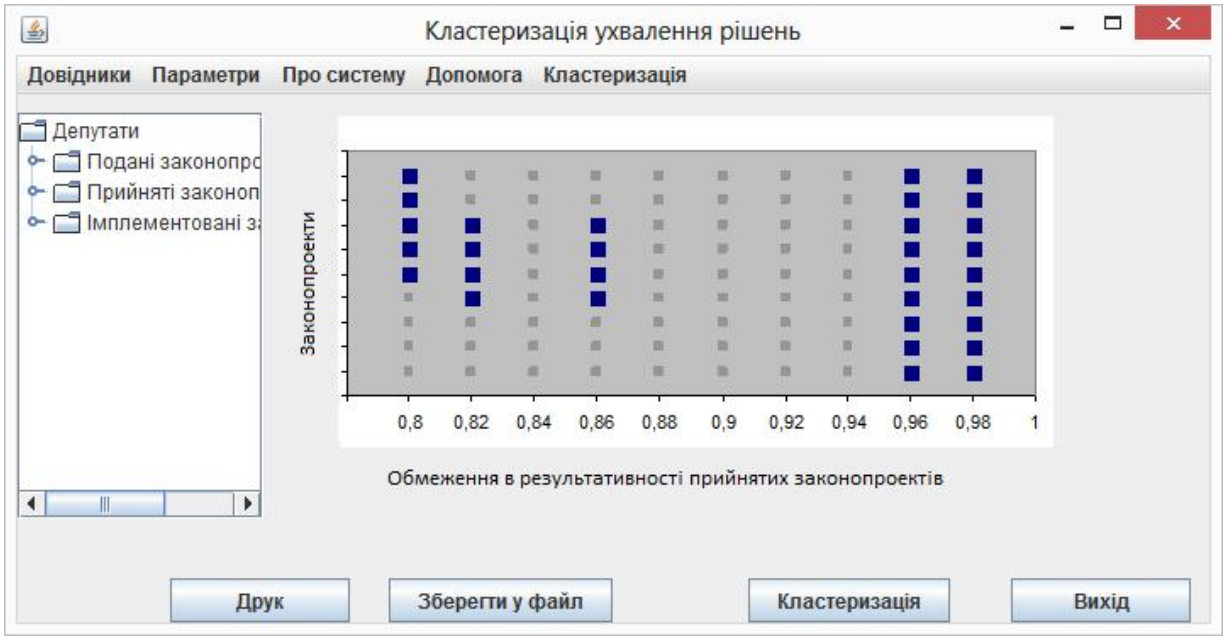

Рис. 3.19. Область компромісного рішення в просторі обмежень

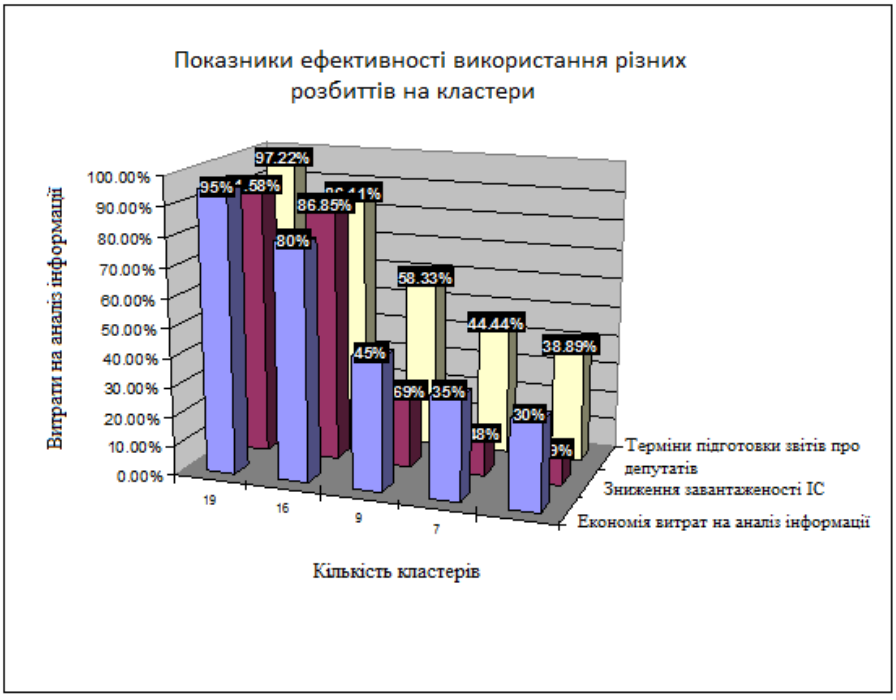

Рис. 3.20. Розрахункові показники ефективності впровадження системи кластеризації

В роботі проведено розрахунок ефективності впровадження системи кластеризації.

Висновки до розділу 3

В даному розділі проведено практичну реалізацію розроблених методів та алгоритмів, реалізована програмна система структурування багатовимірних даних з використанням запропонованого алгоритму кластеризації. Здійснено програмну реалізацію системи, проведено апробацію та тестування.

На основі проведеного аналізу та експертної оцінки результатів впровадження програмного модуля виявлено ефективність використання розробленого алгоритму в системах підтримки прийняття рішень.

ВИСНОВКИ

В процесі виконання магістерської роботи отримані наступні наукові та практичні результати:

1. Проведено огляд методів автоматичної класифікації, кластерного аналізу. Проведено огляд публікацій, присвячених методам кластеризації з використанням інформації, отриманої від експертів. Обгрунтовано обрані методи кластеризації, застосовні до об'єктів багатопараметричної структури.

2. Формалізовано задачу багатопараметричної кластеризації. Визначено модель об’єктів, що розглядаються в різних просторах ознак. Запропоновано дворівнева схема вибору пари найбільш близьких об’єктів в процесі багатопараметричної кластеризації. Розроблено алгоритми багатопараметричної кластеризації для різних типів експертних оцінок: обмеження на значення метрики, обмеження на характеристики кластерів.

3. Запропоновано алгоритм кластеризації, розроблений на основі діалогового алгоритму Соболя-Статнікова. Запропоновано схему етапів проведення кластеризації. Виявлено переваги та недоліки запропонованого методу.

4. Реалізована програмна система структурування багатовимірних даних з використанням запропонованого алгоритму кластеризації.

5. За допомогою розробленого програмного модуля вирішена практична задача автоматичної кластеризації законопроектів та депутатів з метою ухвалення рішення майбутніми виборцями. Проведено аналіз результатів кластеризації. Визначено переваги методу в порівнянні з експертною оцінкою.

6. На основі проведеного аналізу та експертної оцінки результатів впровадження сситеми виявлено ефективність використання розробленого алгоритму в системах підтримки прийняття рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Самойлов, В. Д. Модельное конструирование компьютерных приложений / В. Д. Самойлов. - К.: Наукова думка. - 2007. - 198 с.

2. Лаврищева, Е. М. Методы программирования: теория, инженерия, практика / Е. М. Лаврищева. - К.: Наукова думка. - 2006. - 451 с.

3. Чмир, І.О. Моделювання систем у середовищі UML (Unified Modeling Language) : навч. посібник / І. О. Чмир, М. Ф. Ус ; Черкаськ. акад. менеджменту. - Черкаси : ЧАМ, 2004. - 100 с.

4. Шатовська Т. Б. Комбінований ієрархічний підхід кластеризації документів [Електронний ресурс] / Т. Б. Шатовська, І. В. Каменєва // Харківський національний університет радіоелектроніки. – 2009. – Режим доступу до ресурсу:http://visnyk.vntu.edu.ua/index.php/visnyk/article/view/696/695.

5. Buasilovsky, P. Adaptive and intelligent Web-based educational systems / P. Buasilovsky, C Peylo // International Journal of Artificial Intelligence in Education. Special Issue on Adaptive and Intelligent Web-based Educational Systems -2003. -№13 (2-4). -P. 159-172.

6. Murray, T. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art / T. Murray. // International Journal of Artificial Intelligence in Education. -1999.-№10-P. 98-129

7. Андон, Ф. И. Логические модели интеллектуальных информационных систем / Ф. И. Андон, А. Е. Яшунин, В. А. Резниченко. - К: Наукова думка.-1999.-397 с.

8. Краковецкий А. Кластеризация: алгоритмы k-means и c-means [Електронний ресурс] / Александр Краковецкий // Habrahabr. – 2009. – Режим доступу до ресурсу: http://habrahabr.ua/post/67078/.

9. Marshall, B. Convergence of Knowledge Deputatagement and E-Learning: the GetSmart Experience [Електронний ресурс] / Marshall, B., et al. // JCDL. - Houston, 2003. - Режим доступу: http://ai.bpa.arizona.edu/go/intranet/

Publication/JCDL-2003-Marshall.pdf.

10. Семикин, В. А. Семантическая модель контента образовательных электронных зданий: Автореф. дис. … канд. тех. наук: 05.13.18 / Семикин Виктор Алексеевич ; Тюменск. гос. ун-т. - Тюмень, 2004. - 21 с.

11. Титенко, С. В. FreshKnowledge - система управління навчальним Веб-контентом на семантичному рівні / С. В. Титенко, О. О. Гагарін // VII международная конференция «Интеллектуальный анализ информации ИАИ-2007», Киев, 15-18 мая 2007г. : Сб. тр. / Ред. кол. : С. В. Сирота (гл.ред.) и др. - К.: Просвіта, 2007. - С. 342-352.

12. Олецький О. В. Застосування формальних моделей онтологій для формалізації інформаційних потоків у системах управління контентом / О. В. Олецький //Теоретичні та прикладні аспекти побудови програмних систем. Матеріали міжнародної конференції TAAPSD'2005. Київ, 7-9 грудня 2005 р. - С. 26-29.

13. Lewisand D. Lewisand D. Acomparison of two learning algorithms for text categorization [Електронний ресурс] / D. Lewisand, M. Ringuette // In Third Annual Symposium on Document Analysis and Information Retrieval. – 1994. – Режим доступу до ресурсу: http://www.research.att.com/~lewis/papers/lewis94b.ps.

14. FreshKnowledge CMS - онтологічно-орієнтована система керування контентом, розроблена здобувачем [Електронний ресурс]. - Режим доступу : http://www.freshknowledge.net.

15. Елизаренко, Г. Н. Проектирование компьютерных курсов обучення: концепция, язык, структура / Г. Н. Елизаренко. - К.: НТУУ «КПИ», 2001.

16. Люгер, Джордж, Ф. Искусственный интеллект: стратегии и методы решення сложных проблем / Люгер, Джордж, Ф. - 4-е издание.: Пер. с англ.- М.: Издательский дом «Вильяме», 2005. - 864 с.

17. TENCompetence - European research project for lifelong competence development [Електронний ресурс]. - Режим доступу : http:// www.tencompetence.org/.

18. Валгина, Н. С. Теория текста: Учебное пособие / Н. С. Валгина. -

Москва. Изд-во МГУП «Миркниги» - 1998. - 210 с.

19. Дивак М. П., Шпінталь М.Я., Шевчук Р.П., Козак О.Л., Пукас А.В., Спільчук В.М., Гончар Л.І. Методичні рекомендації до виконання та захисту дипломної роботи на здобуття освітньо-кваліфікаційного рівня магістр за спеціальностями: 8.05010301 “Програмне забезпечення систем” та 8.05010302 “Інженерія програмного забезпечення” // Тернопіль : Економічна думка, 2011. — 31 с.