Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)
Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти____перший (бакалаврський)_____
(рівень вищої освіти)

Система розпізнавання зразків у великих даних на основі векторного методу
(тема)

Виконав: здобувач IV курсу, групи КІУКІ-21-7
_____Прохорчук В.І.
(прізвище, ініціали)
Спеціальність *123 _Комп'ютерна інженерія*
Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_____
_____*Комп'ютерна інженерія*_____
(повна назва освітньої програми)

Керівник Чумаченко С.В
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри АПОТ _____ Чумаченко С.В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123_Комп'ютерна інженерія
(шифр і назва)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри АПОТ

Чумаченко С.В.
(підпис)
«_06_»___05_____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Прохорчуку В'ячеславу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Система розпізнавання зразків у великих даних на основі векторного методу
затверджена наказом по університету від "21 "_____05_____2025 р. № 403 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії _____15.06.2025_____
3. Вихідні дані до роботи (проекту)
Моделі та алгоритми для пошуку та розпізнавання
Мова програмування С++

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
Огляд технологій
Аналіз моделей та алгоритмів ідентифікації зразків (шаблонів)
Розробка системи розпізнавання зразків на основі векторного методу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) __21__слайд_____

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження	06.05.2025 - 06.05.2025	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів. Складання аналітичного огляду стану технологій. Аналіз останніх досліджень та публікацій. Написання вступу та огляду моделей й алгоритмів.	07.05.2025 -10.05.2025	
3	Опис обраних моделей, методів та алгоритмів	10.05.2025 -17.05.2025	
4	Розробка системи розпізнавання зразків (шаблонів) на основі векторного методу	18.05.2025 -23.05.2025	
5	Опис програмного засобу, розробка структури автомату	24.05.2025 -30.05.2025	
6	Складання прототипу програми, тестування	01.06.2025 -05.06.2025	
7	Оформлення пояснювальної записки	06.06.2025 -10.06.2025	
8	Перевірка виконаного проекту керівником, допуск до захисту	11.06.2025 -12.06.2025	
9	Захист проекту	26.06.2025	

Дата видачі завдання 06.05.2025

Здобувач _____ _____ Прохорчук В.І.
(підпис)

Керівник роботи _____ проф.Чумаченко С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 46 с., 12 рисунків, 10 табл., 1 дод., 14 джерел.

СИСТЕМА, АВТОМАТ, КОМП’ЮТИНГ, ПОШУК, ІДЕНТИФІКАЦІЯ, МЕТРИКА, ФУНКЦІОНАЛЬНІСТЬ, МОДЕЛЬ, ТЕСТУВАННЯ, ВЕРИФІКАЦІЯ, ВЕКТОРНИЙ МЕТОД, ЧАСОВИЙ ІНТЕРВАЛ,

Робота по’вязана з підвищенням ефективності пошуку зразків (шаблонів) за рахунок розробки системи їх розпізнавання у великих даних на основі детермінованих методів.

В роботі розглянуто стан технологій; виконано аналіз публікацій, моделей та алгоритмів; розробка системи розпізнавання зразків, що містить реалізацію векторного завдання символьних послідовностей (векторного методу), унітарно кодовану матричну модель як декартів добуток множин-примітивів, синтезовану логіку алгоритмів для пошуку, ідентифікації та класифікації зразків (шаблонів); виконано тестування програми; розроблено структура автомату для розпізнавання різниці у зразках на основі матричного аналізу великих даних.

ABSTRACT

Bachelor’s thesis contains 46 pages format A4, 12 figures, 10 tables, 1 application, 14 sources.

SYSTEM, AUTOMAT, COMPUTING, SEARCH, IDENTIFICATION, METRICS, FUNCTIONALITY, MODEL, TESTING, VERIFICATION, VECTOR METHOD, TIME INTERVAL

The work is related to increasing the efficiency of searching for samples (templates) by developing a system for their recognition in big data based on deterministic methods.

The work considers the state of technology; analyzes publications, models and algorithms; develops a system for pattern recognition, which includes the implementation of the vector problem of character sequences (vector method), a unitarily encoded matrix model as a Cartesian product of primitive sets, synthesized logic of algorithms for searching, identifying and classifying samples (templates); tests the program; structure of the automat for recognizing differences in samples based on matrix analysis of big data.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
ВСТУП 8
1 СТАН ТЕХНОЛОГІЙ 9
1.1 Тенденції розвитку технологій 9
1.2 Топ-технології від Gartner 10
1.3 Аналіз останніх досліджень та публікацій 15
1.4 Висновки до розділу 1 20
2 МОДЕЛІ ТА АЛГОРИТМИ ІНДЕНТИФІКАЦІЇ ЗРАЗКІВ 21
2.1 Модель пошуку та індентифікації зразків 21
2.2 Синтез моделі 25
2.3 Матричний паралельний аналіз великих даних 28
2.4 Алгоритм класифікації 31
2.5 Алгоритм створення універсуму 31
2.6 Висновки до розділу 2 35
3 СИСТЕМА РОЗПІЗНАВАННЯ ЗРАЗКІВ НА ОСНОВІ ВЕКТОРНОГО МЕТОДУ 36
3.1 Визначення класів еквівалентностей за макро-алгоритмом 36
3.2 Алгоритми визначення кластерів 37
3.3 Опис програмного засобу 40
3.4 Автомат для визначення різниці у зразках 44
3.5 Висновки до розділу 3 45
ВИСНОВКИ 46
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 47
ДОДАТОК А Графічна частина 49

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence (штучний інтелект);

AI TRiSM – Artificial Intelligence Trust, Risk, and Security Management (управління довірою, ризиками та безпекою за допомогою штучного інтелекту);

AffectNet – великий набір даних про вирази обличчя;

BU-3DFE – 3D facial expression database (база даних 3D-виразів обличчя);

CCPR – Control Cards Pattern Recognition (розпізнавання образів контрольних карток);

CTEM – Continuous Threat Exposure Management (безперервне управління впливом загроз);

DC – детермінований цифровий комп’ютинг;

FER – facial expression recognition (розпізнавання виразу обличчя);

FML – Federated Machine Learning (федеративне машинне навчання);

GenAI – Generative AI (генеративний ШІ);

JPR – розпізнавання образів перешкод;

MDV – Matrix Deductive Vectors (матриця дедуктивних векторів);

MIV – Matrix Input Vectors (матриця вхідних векторів);

MS-CCPR – схема розпізнавання образів багатомасштабних контрольних діаграм;

RAF-DB – Real-world Affective Faces Database (база даних реальних афективних облич);

SFEW – Static Facial Expressions in the Wild (статичні вирази обличчя в дикій природі);

ГСА – граф-схема автомата;

ШІ – штучний інтелект.

ВСТУП

Тематика роботи стосується питань створення системи розпізнавання зразків шляхом реалізації детермінованих методів, а саме методу пошуку та розпізнавання зразків (шаблонів) на основі векторного завдання символьних послідовностей, які ідентифікують бізнес-зразки у потоках великих даних. Виконується верифікація знайденої сукупності зразків на основі метрики перетину/об'єднання, тестування та верифікація програми.

Мета дослідження – підвищення ефективності пошуку зразків (шаблонів) за рахунок розробки системи їх розпізнавання у великих даних на основі векторного методу для заданих часових інтервалів.

Задачі: огляд стану технологій; аналіз публікацій, аналіз моделей та алгоритмів; розробка системи розпізнавання зразків, що містить реалізацію векторного завдання символьних послідовностей (векторного методу), унітарно кодовану матричну модель як декартів добуток множин-примітивів, синтезовану логіку алгоритмів для пошуку, ідентифікації та класифікації зразків (шаблонів); тестування програми; розробка структури автомату для розпізнавання різниці у зразках на основі матричного аналізу великих даних.

1 СТАН ТЕХНОЛОГІЙ

1.1 Тенденції розвитку технологій

Детермінований цифровий комп'ютинг традиційно вважається переважаючим над будь-якою ймовірнісною технологією, такою як машинне навчання, нечітка логіка, штучний інтелект, нейронні мережі та еволюційні алгоритми. Основне завдання всіх технологій штучного інтелекту полягає у досягненні рівня детермінізму, властивого класичному комп'ютингу. Важливими показниками для досягнення цієї досконалості є критичні параметри: Yield – показник якості обчислень або розпізнавання, Time-to-Market – час, необхідний для досягнення зазначеної якості. В цьому контексті особливий інтерес викликає співвідношення між штучним інтелектом (AI) і детермінованим комп'ютингом (DC) у рамках точності (якісного виконання завдань) та термінах Yield і Time-to-Market при розробці обчислювальних систем, придатних для практичного застосування.

Таким чином, DC характеризується як інтелектуальна творчість (human intelligence), коли потрібно інтелектуальне напруження, щоб отримати точний розв’язок, а AI – є рутинне навчання (routine training), що призведе до ймовірнісного неточного розв’язку за тривалий час. Іншими словами, штучний інтелект не вимагає природного інтелекту, а детермінований комп'ютинг обов'язково використовує високий рівень природного інтелекту. Виходить, що дедуктивні методи, які мають апріорну точність логічних виразів або аналітичних формул, є кращими для їх імплементації в практику, ніж індуктивні методи, що вимагають тривалого навчання та верифікації для впровадження у виробничі процеси. Але залишається велике поле невизначених (неповних, нечітких) знань про процес чи явище, де тимчасовим виходом може служити ймовірнісний AI-розв’язок задачі.

1.2 Топ-технології від Gartner

Метрично визначити належність задачі до потенційного детермінізму чи ймовірнісного підходу її вирішення – було, є і буде моментом істини у творчості на полі emerging computing. Деяким підтвердженням сказаного є трендова картина топ-технологій від компанії Gartner [1], що було спрогнозовано ще у 2021 році, представлена на рис. 1.1.

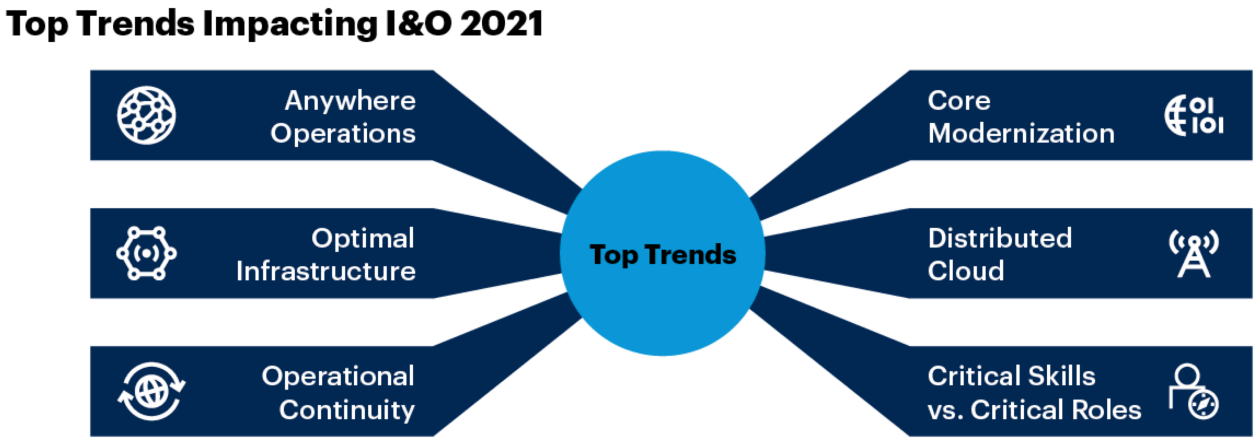


Рисунок 1.1 – Напрямки впливу моделей інфраструктури та операцій на бізнес згідно до Gartner 2021 [1]

Цикл Гартнера – концепція, запропонована американською консалтинговою компанією Gartner Inc. Вона описує життєвий цикл інновацій, щоб розробники мали змогу вибудовувати ефективну стратегію просування продукту різних етапах його розвитку: від виходу ринок до повного виведення.

«Крива хайпа» збирає технології певного напрямку та розподіляє їх в одну з категорій:

1. Інноваційний тригер — перші згадки та публікації про нову технологію.

2. Пік надмірних очікувань – технологію широко обговорюють і від неї чекають революції.

3. Порятунок від ілюзій – у технології виявляють недоліки і завищені очікування змінюються розчаруванням.

4. Подолання недоліків – технологія позбавляється недоліків і дитячих хвороб, стає придатною для впровадження в бізнес.

5. Плато продуктивності – технологія приймається суспільством і стає нормою.

Ще в «кривій» є часові позначки, коли технологія вийде на плато і прогнозований термін може досягати 10 років.

Крива Гартнера допомагає визначити перспективні напрямки розвитку технологій, щоб підготуватися до майбутніх подій, оптимізувати свої ресурси та ухвалити стратегічні рішення для подальшого зростання та розвитку, тобто, отримати конкурентні переваги перед тими, хто такого планування не проводив.

Цикл ажіотажу навколо нових технологій є унікальним серед циклів ажіотажу Gartner, оскільки він збирає ключові висновки з понад 2000 технологій та прикладних фреймворків, які Gartner щорічно аналізує, у стислий набір «обов'язкових для знання» нових технологій. Ці технології мають потенціал для досягнення трансформаційних переваг протягом наступних від 2 до 10 років.

25 революційних технологій, за якими варто стежити в рамках циклу ажіотажу Gartner, Inc. щодо нових технологій у 2024 році, поділяються на чотири ключові області: автономний ШІ, продуктивність розробників, комплексний досвід та орієнтовані на людину програми безпеки та конфіденційності (рис. 1.2) [2].

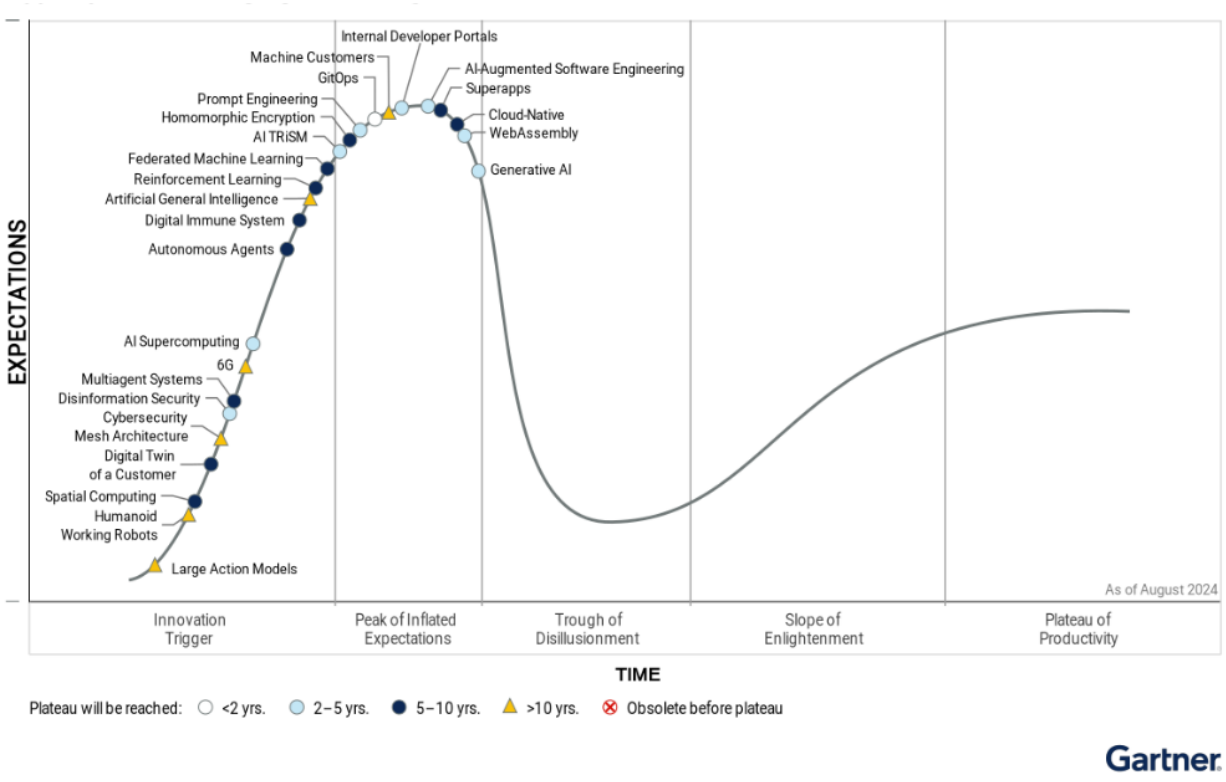


Рисунок 1.2 – Крива Гартнера 2024 (Hype Cycle for Emerging Technologies, 2024) [2]

«Генеративний ШІ (GenAI) перетнув пік завищених очікувань, оскільки фокус бізнесу продовжує зміщуватися з ентузіазму навколо базових моделей на варіанти використання, що підвищують рентабельність інвестицій», – сказав Арун Чандрасекаран, заслужений віце-президент-аналітик Gartner [2]. Це прискорює розвиток автономного ШІ. Хоча поточному поколінню моделей ШІ бракує свободи дій, дослідницькі лабораторії ШІ швидко випускають агентів, які можуть динамічно взаємодіяти зі своїм середовищем для досягнення цілей, хоча цей розвиток буде поступовим процесом.

«Навіть коли ШІ продовжує привертати увагу, IT-директори та інші IT-керівники також повинні вивчити інші нові технології з трансформаційним потенціалом для розробників, безпеки, досвіду клієнтів і співробітників і розробити стратегію використання цих технологій відповідно до здатності

своїх організацій обробляти неперевірені технології», – зазначив Чандрасекаран [2].

У 2024 році аналітики Gartner проаналізували понад 2 тисячі технологій і вибрали 25 найперспективніших, які можуть претендувати на масовість у найближчі роки. 10 головних стратегічних технологічних трендів Gartner на 2024 рік пропонують інновації, що можуть допомогти швидше досягти бізнес-цілей, особливо в епоху штучного інтелекту, що швидко розвивається [2]. В цілому, ці стратегічні технологічні тенденції відіграватимуть важливу роль у прийнятті рішень щодо бізнесу та технологій протягом наступних трьох років (рис. 1.3).

Рисунок 1.3 – Схема щодо топ-10 стратегічних технологічних трендів від Gartner на 2024 рік [2]

Gartner настійно рекомендує оцінювати вплив і переваги кожної з цих технологічних тенденцій, щоб визначити, яка інновація – або стратегічна комбінація – матиме найбільш значний вплив на успіх певної організації:

1) управління довірою, ризиками та безпекою за допомогою штучного інтелекту (AI TRiSM);

2) безперервне управління впливом загроз (CTEM);

3) сталі технології;

4) платформна інженерія;

5) розробка доповненої реальності за допомогою штучного інтелекту;

6) галузеві хмарні платформи;

7) інтелектуальні додатки;

8) демократизований генеративний ШІ;

9) інтелектуальна підтримка співробітників;

10) машинні клієнти.

Кожна з цих тенденцій пов'язана з однією або кількома ключовими бізнес-темами: захист і збереження минулих і майбутніх інвестицій, розробка правильних рішень для потрібних клієнтів у потрібний час і створення цінності в мінливому середовищі як для внутрішніх, так і для зовнішніх клієнтів.

До 2026 року генеративний ШІ суттєво змінить 70% проектування та розробки нових веб- та мобільних додатків.

До 2027 року 80% IT-директорів матимуть показники продуктивності, засновані на стійкості IT-організації.

Як зазначив Барт Віллемсен, віце-президент з досліджень Gartner [2], технологічні катастрофи та соціально-економічна невизначеність вимагають готовності діяти сміливо та стратегічно підвищувати стійкість, а не покладатися на ситуативні заходи. IT-лідери повинні дбати про прораховані ризики та робити надійні та довгострокові інвестиції для забезпечення сталого внутрішнього та зовнішнього створення цінностей.

1. Щорічне дослідження Gartner ключових стратегічних технологічних тенденцій допомагає визначити пріоритетність інвестицій, особливо в епоху штучного інтелекту.

2. Тенденції 2024 року забезпечують одну або кілька ключових переваг: захист інвестицій, оптимізацію зростання творців інтелектуальних додатків/рішень і збільшення цінностей.

3. Складання правильної комбінації технологій для досягнення цілей CEO та IT-директорів на найближчі роки.

1.3 Аналіз останніх досліджень та публікацій

Тут акцент зроблено на операційній локальності виконання бізнес-функцій в оптимальній кібер-фізичній інфраструктурі, що забезпечується співробітниками, які володіють не просто певними ролями, а критичними знаннями про нові технології, включаючи хмарні федеративні послуги машинного навчання FML (Federated Machine Learning) [3, 4].

Візуалізація структурованих даних, які з часом розвиваються в ієрархічному порядку, стає проблематичною. Відомі кілька способів її вирішення [5]. Наприклад, анімовані або співставлені деревоподібні візуалізації недостатньо ефективні для надання загального огляду часових рядів і не можуть повноцінно відобразити зміни, що відбуваються з часом. Вкладені потокові графи пропонують кращу інтерпретацію еволюції даних, однак вони не забезпечують чітке уявлення про ієрархічні структури на конкретному часовому етапі. Ці підходи часто обмежуються статичними ієрархіями або не враховують складні зміни в ієрархії даних, що звужує можливості їх застосування. Запропонований тут метод забезпечує плавний перехід між деревоподібними картами і вкладеними графами потоків, що дозволяє досліджувати баланс між динамічною поведінкою і ієрархічною структурою. Завдяки здатності обробляти топологічні зміни будь-якого типу, цей підхід підходить для широкого спектра додатків. Його корисність

демонструється через кілька прикладів та оцінюється за результатами користувацького дослідження, з доступністю повного вихідного коду.

У дослідженні [6] розпізнавання образів контрольних карток (CCPR) у більшості попередніх методів використовувався класифікатор для позначення аномальних карток. Проте в довгострокових даних контрольної картки часто зустрічається велика кількість незначних аномальних патернів, які відрізняються від загальної картини. Крім того, існує вагома ймовірність, що локальні аномальні патерни потребують уваги та аналізу. Представлено новий підхід до розпізнавання образів у багатомасштабних контрольних діаграмах MS-CCPR, який орієнтований не на класифікацію даних однієї діаграми, а на аналіз закономірностей у часових рядах. Запропонована схема базується на використанні гістограмного представлення даних разом із зіставленням підпослідовностей часових рядів для виявлення аномалій на різних масштабах у довгих рядах контрольних карт. Результати експериментів підтверджують ефективність цієї методики у виявленні графічних патернів на різних рівнях масштабу, перевершуючи сучасні алгоритми зіставлення часових послідовностей.

У роботі [7] розглянуто проблему розпізнавання образів перешкод (JPR), коли деякі образи не мають навчальних вибірок. Існуючі роботи у цьому напрямі описують розпізнавання лише відомих моделей перешкод і розглядають ситуації із невідомими перешкодами. Автори пропонують схему JPR на основі навчання з нульовим пострілом (ZSL). Спочатку впроваджується контрольований процес навчання для дослідження уявлення прихованих ознак відомих шаблонів перешкод. Далі пропонується неконтрольований підхід до класифікації для розпізнавання різних моделей перешкод. У результаті як відомі, так і невідомі шаблони перешкод класифікуються в просторі прихованих ознак. Результати моделювання свідчать, що запропонована схема демонструє чудову продуктивність при вирішенні відкритих задач JPR.

Останніми роками розпізнавання виразів обличчя (FER) привертає значну увагу завдяки своєму широкому застосуванню. Незважаючи на певні досягнення, зокрема завдяки появі глибокого навчання, проблема зміни поз залишається невирішеною. Більшість традиційних підходів фокусуються на FER у контрольованих лабораторних умовах, тоді як розпізнаванню виразів обличчя в реальному середовищі приділяється менше уваги. Реалізація FER у таких умовах можлива за допомогою моделі розпізнавання виразів, що не залежить від пози, але цьому перешкоджає нестача навчальних даних. Достатні навчальні дані з надійними мітками виразів для FER зазвичай недоступні. У статті [8] обговорюється проблема моделювання варіацій пози на зображеннях обличчя, а також використання зашумлених даних з Інтернету з метою покращення продуктивності розпізнавання емоцій (FER). Представлена модель реалізована наскрізно з обмеженим контролем та пропонує низку переваг. По-перше, вона ефективно використовує великі обсяги зашумлених даних, щоб покращити роботу класифікатора FER, який пройшов навчання на невеликому наборі чистих даних. По-друге, вводиться інноваційна мережа для моделювання поз, яка адаптивно вирівнює відмінності у просторі представлень глибинних зображень облич за різних положень голови. Запропонована модель сприяє вивченню варіантів виразів, незалежних від пози. Для оптимального використання корисної інформації із зашумлених даних розроблена мережа моделювання шуму, яка спроможна навчатися відображенню простору ознак шляхом визначення залишків між чистими та зашумленими мітками. Запропонований підхід перевірено на чотирьох загальнодоступних тестах FER: AffectNet, RAF-DB, SFEW та BU-3DFE. Експерименти підтверджують, що запропонований метод кращий за інші сучасні методи. Іноді розпізнавач сміттєвих даних працює ефективніше, ніж розпізнавач функціонального ряду.

Прогноз фінансових часових рядів [9] є основним об'єктом вивчення у галузі фінансової економетрики. На першому етапі проводиться відокремлення будь-яких систематичних змін цих рядів від їх випадкових

коливань. Систематичні зміни можуть бути результатом трендів, сезонних і циклічних коливань. Економетричні моделі мають різний рівень складності для відтворення різних закономірностей. Однак алгоритми машинного навчання можна застосовувати для прогнозування нелінійних часових рядів, оскільки вони здатні навчатися і адаптуватися до змін на фінансових ринках. Найбільш стандартним економетричним підходом до прогнозування тенденцій фінансових часових рядів є методологія Бокса та Дженкінса (1970). Розглянутий підхід заснований на визначенні відповідних систематичних варіацій часового ряду (тренд, сезонні або циклічні ефекти).

В роботі [10] запропоновано універсальну метрику пошуку даних у кіберпросторі, яка базується на використанні параметрів подібності-різниці та матричної структури у двійковій формі. Описано метод аналізу матричних структур даних з використанням метрики подібності-різниці для виявлення несправностей у цифрових системах. Метод різницевої діагностики характеризується виконанням трьох логічних операцій над двійковими станами матриці-векторів-рядків та орієнтований на одиночні та множинні несправності в цифрових системах та програмних додатках. Кубіт-різницевий метод виявлення несправностей характеризується використанням векторних паралельних логічних операцій для формування діагнозу багатозначного технічного стану. Показано перевагу кубіт-векторного подання даних у комірках матриці, що дозволяє виконувати логічні операції паралельно у трьох автоматичних циклах для отримання необхідного результату. Описано метод виявлення одиночних та множинних несправностей у цифрових програмно-апаратних системах, орієнтований на апаратну реалізацію паралельних логічних регістрових операцій, які забезпечують значне підвищення продуктивності порівняно з існуючими аналогами. Запропоновано метод та апаратну реалізацію секвенсора для визначення подібності-різниці-включення, що характеризується отриманням більш точної структурованої оцінки взаємодії двох об'єктів.

В роботі [11] запропоновано моделі, методи та алгоритми для кіберсоціальних обчислень, що використовують метрику подібності-різниці унітарної кодованої інформації для обробки великих даних з метою генерації адекватних сигналів виконавчих механізмів для керування критичними кіберсоціальними системами. Розроблено теоретико-множинний метод пошуку даних, заснований на подібності-різниці частотних параметрів примітивних елементів, який дозволяє визначати подібність об'єктів, стратегію перетворення одного об'єкта в інший, а також виявляти рівень спільних інтересів, конфлікту. Наведено визначення фундаментальних понять у галузі обчислювальної техніки на основі метричних співвідношень між взаємодіючими процесами та явищами. Запропоновано програмний додаток для розрахунку подібності-різниці об'єктів на основі формування частотних векторів двох наборів примітивних даних. Показано високий рівень кореляції між результатами застосування та відомою системою визначення плагіату.

У [12] розглянуто ідентифікація шаблонів та об'єктів за допомогою ключових слів та даних. Проаналізовано такі проблеми: пошук ідентифікаторів об'єктів на основі таблиці істинності, де ключові слова формуються у стовпцях, що визначають подібність та відмінність об'єктів у потоці; визначення класів еквівалентності ключових слів на основі аналізу стовпців таблиці істинності, побудованої на основі оцифрованого потоку даних; генерація моделі бізнес-процесу у вигляді логічного вектора на основі двійкової матриці шаблонів; використання моделі бізнес-процесу для перевірки дій операторів, що виконують регуляторну функціональність на підприємстві. Запропоновано перенесення архітектури фон Неймана в пам'ять та заміну потужного процесора читання-запису на логічні векторні транзакції для зменшення енергетичних та часових витрат під час обробки великих даних. Розробляються інтелектуальні структури даних та прості алгоритми на основі транзакцій читання-запису для великих даних у пам'яті як адресні обчислення.

Запропонований у статті [13] метод застосовує унітарне кодування інформаційних примітивів і даних, які потім представляються у формі послідовності або двійкового вектора. Цей вектор відображає бізнес-патерн або функціональність, що виконується оператором впродовж робочого дня.

1.4 Висновки до розділу 1

Проаналізовано технологічні тенденції, що визначені компанією Gartner за 2021-2024 роки. Проведено аналіз сучасних досліджень та публікацій. Робиться висновок про те, що дедуктивні методи, які мають апріорну точність логічних виразів або аналітичних формул, є кращими для їх імплементації в практику, ніж ймовірнісні методи ШІ, що вимагають тривалого навчання та верифікації для впровадження у виробничі процеси, хоча вони є зараз на часі.

Мета дослідження – підвищення ефективності пошуку зразків (шаблонів) за рахунок розробки системи їх розпізнавання у великих даних на основі векторного методу для заданих часових інтервалів, що семантично утворюють бізнес-функціональності у часовому фреймі із зайвими даними.

2 МОДЕЛІ ТА АЛГОРИТМИ ІНДЕНТИФІКАЦІЇ ЗРАЗКІВ

Розглядаються моделі пошуку та ідентифікації зразків (шаблонів) та аналізуються алгоритми синтезу моделей, класифікації та створення універсуму примітивів на основі [10-14].

2.1 Модель пошуку та індентифікації зразків

Модель для синтезу логіки алгоритму пошуку та ідентифікації слугує базовою або початковою матрицею, яка є форматованою в метриці параметрів <string-primitives – business-flow>, що представлено у табл. 2.1.

Таблиця 2.1 – Матриця у метриці параметрів
<рядок-примітив – бізнес-функціональність>

Рядок	Зразки (шаблони)											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1				1							
D		1		1								
C			1					1				
D							1					
E					1							
F						1						
G												
H									1			
K											1	
L										1		1

У даному контексті кожен стовпець може містити лише одну одиницю, тому що скріншот не має й не повинен містити два активних вікна різних функцій. Рядки, у свою чергу, відображають сумарну кількість одиниць, що вказують на участь примітиву в одному або декількох патернах протягом певного заданого інтервалу часу, наприклад, вісім годин. Зазвичай послідовність стовпців-векторів з одиницями формує патерн завдяки своїм

літерним (символьним) ідентифікаторам. Тривалість окремого патерну визначається часовим інтервалом, що володіє специфічною статистичною обробкою даних. Часовий інтервал реалізації бізнес-функціональності виступає у ролі верифікатора та роздільника патернів, які можуть мати однакові множини символьних ідентифікаторів для формування бізнес-функціональностей. Прикладом реалізації цієї процедури є табл. 2.2.

Таблиця 2.2 – Модифікована матриця після розбиття за параметрами

Рядок	Зразки (шаблони)											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1											
D		1										
C			1									
D						1						
E					1							
F				1								
G							1					
H								1				
K									1			
L												1
M											1	
N										1		

Елементи спочатку діляться на менші підмножини, а потім ці підмножини об'єднуються в більші групи, організовані по стовпцях (1-3, 4-6, 7-9, 10-12). Ці групи не мають спільних елементів при розгляді їх перетинів по рядках і стовпцях. Всі ці групи разом утворюють бізнес-потік (табл. 2.3).

Знайти кореляцію між примітивами, що становлять патерн, шляхом застосування алгоритмів класифікації – є сутність завдання. Для підвищення продуктивності навчання можна використовувати розподілену структуру, представлену на рис. 2.1.

Таблиця 2.3 – Групування зразків за стовпцями

Рядок	Зразки (шаблони)											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1											
D		1										
C			1									
D						1						
E					1							
F				1								
G							1					
H								1				
K									1			
L												1
M											1	
N										1		

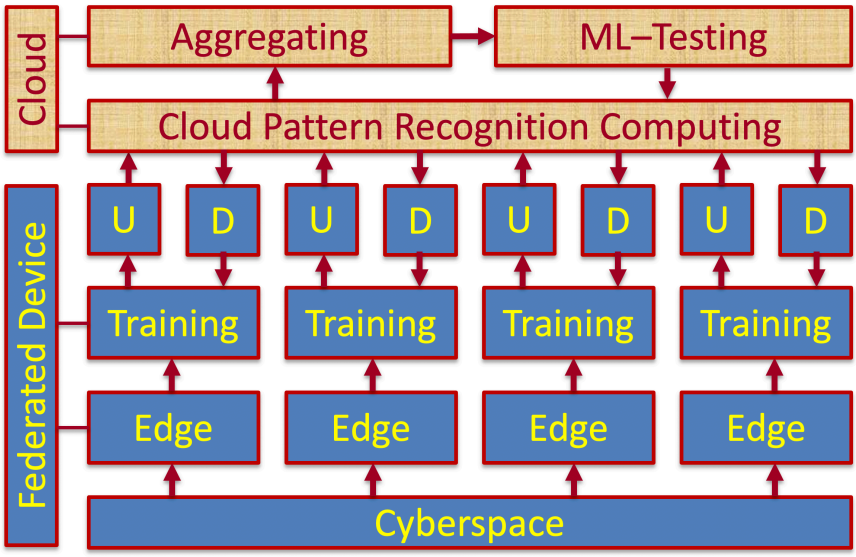


Рисунок 2.1 – Масове машинне навчання FML [13]

Далі потрібно проаналізувати, як часто різні літери-примітиви зустрічаються разом. Для цього будується спеціальна матриця, яка показує зв'язки між літерами. Аналіз цієї матриці дозволяє виявити групи літер, які утворюють певні патерни. Важливо, щоб ці групи частково перекривалися, але разом охоплювали всі можливі літери (еквівалентування).

Для послідовності примітивів $G_i=ID_i(F_1, F_2, \dots, F_j, \dots, F_n)$, що можуть розглядатися як фрагменти, з одним цифровим ID-ключом, кожен фрагмент символів є рядком, який має початок та кінець:

$$F_i=(start, a_1, a_2, \dots, a_i, \dots, a_m, end - ID_i). \quad (2.1)$$

Сукупність із n послідовностей дій за вісім годин формує бізнес-потік: $G=(G_1, G_2, \dots, G_i, \dots, G_n)$. Створення функціонально подібних послідовностей і їх об'єднання в єдиний клас передбачає розробку set-моделі бізнес-процесу, що є більш наочною і простою для аналізу. Потрібно сформувати k різних класів функціональностей у потоці G та провести верифікацію цих k класів еквівалентних послідовностей дій. Важливо пам'ятати, що еквівалентні послідовності можуть мати схожість до 90 відсотків або менше.

Еквівалентування являє собою процес виявлення та визначення підмножин послідовностей дій, або функціональних фрагментів, бізнес-процесу, які характеризуються однаковими метричними властивостями, такими як рефлексивність, симетричність і транзитивність. Поняття set-вектора описує набір компонентів у вигляді впорядкованої послідовності символів чи чисел. Усі множини або підмножини визначаються через еквівалентність щодо належності своїх елементів. Дві підмножини вважаються еквівалентними, якщо вони мають однаковий набір елементів. У матричній формі підмножина зображується через квадрат оф діагональних елементів.

Метрика вводиться за наступним правилом [13]: відношення між кінцевим числом об'єктів є еквівалентним, якщо циклічна сума відстаней між ними дорівнює нулю:

$$\oplus_{\mathbf{i=\overline{1,n}}}\mathbf{d_i} = \mathbf{0}. \quad (2.2)$$

Матриця множини є двовимірною структурою, де всі координати визначені як одиниці.

2.2 Синтез моделі

Для алгоритму синтезу моделі передбачаються наступні кроки [13]:

1. Перетворення тест-векторів, що є вхідними даними та які мають початок та кінець-ідентифікатор, на set-вектор $S=(S_1, S_2, \ldots, S_i, \ldots, S_n)$. .

2. Для кожного set-вектора (послідовності дій) будується рядок бінарної матриці $\mathbf{M_{ij}}(Si)=1$, що матиме m рядків (за кількістю set-векторів) і стільки стовпців, скільки унікальних елементів міститься у вектор-універсумі U. Останнім залишиться індекс p, який визначає потужності унікальних символів у всіх послідовностях дій.

$$\text{if Symbol} \notin U \text{ then } p=p+1. \tag{2.3}$$

Таблиця 2.4 – Формування рядків для бінарної матриці set-векторів

		G=	A	B	C	D	E	F	K	L	M	N	P	T
		U=	1	2	3	4	5	6	7	8	9	10	11	12
S1=ABC		S_1	1	1	1									
S2=ABC		S_2	1	1	1									
S3=DEF		S_3				1	1	1						
S4=KLM		S_4							1	1	1			
S5=ABC		S_5	1	1	1									
S6=NPT		S_6										1	1	1
S7=KLM		S_7							1	1	1			
S8=DEF		S_8				1	1	1						

Переставлення порядку рядків у двійковій матриці дозволяє виявити чітку діагональну структуру, де кожна діагональ відображає сумарний обсяг певного виду діяльності, виконаного оператором протягом зміни (табл. 2.5).

Таблиця 2.5 – Блоково-діагональна структура матриці

G=	A	B	C	D	E	F	K	L	M	N	P	T
U=	1	2	3	4	5	6	7	8	9	10	11	12
S_1	1	1	1									
S_2	1	1	1									
S_5	1	1	1									
S_3				1	1	1						
S_8				1	1	1						
S_4							1	1	1			
S_7							1	1	1			
S_6										1	1	1

Слід відмітити, що перетин діагональних підматриць, які формують еквівалентні послідовності дій, дорівнює порожній множині, а їх об'єднання складає бізнес-процес цілком.

3. Формування оцінок подібності шляхом паралельного xor-порівняння всіх рядків матриці між собою

$$S(S_i, S_j) = \sum_{i=1}^{n}(S_i \bigwedge_{i=\overline{1,n}} S_j) / \sum_{i=1}^{n}(S_i \bigvee_{i=\overline{1,n}} S_j). \quad (2.4)$$

Результат зводиться до квадратної матриці, розмірність якої обумовлена потужністю множини послідовностей дій у бізнес-потоці (табл. 2.6).

Таблиця 2.6 – Оцінки подібності

Sim	S1	S2	S3	S4	S5	S6	S7	S8
S1	1	1,0			1,0			
S2	1,0	1						
S3			1					1,0
S4				1			1,0	
S5	1,0				1			
S6						1		
S7				1,0			1	
S8			1,0					1

4. Групування послідовностей дій (представлених set-векторами) в класи еквівалентності на основі їхньої максимальної подібності. Процес полягає в наступному.

– Визначення класу еквівалентності: береться перший set-вектор (рядок матриці). Знаходяться всі координати в цьому векторі, які мають значення 1. Ці координати ідентифікують стовпці, які належать до одного класу еквівалентності з цим рядком.

– Виключення знайдених послідовностей: усі рядки та стовпці, що відповідають послідовностям дій, які були згруповані в клас еквівалентності (наприклад, S1, S2, S5), обнуляються.

– Оптимізація (необов'язково): Замість обнулення, можна фізично видаляти ці рядки та стовпці з матриці, зменшуючи її розмірність на кожному кроці. Цей підхід проілюстровано в таблиці 2.7.

Таблиця 2.7 – Групування (еквівалентування)

Sim	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
S_1	1	1,0			1,0			
S_2	1,0	1						
S_3			1					1,0
S_4				1			1,0	
S_5	1,0				1			
S_6						1		
S_7				1,0			1	
S_8			1,0					1

Sim	S_3	S_4	S_6	S_7	S_8
S_3	1				1,0
S_4		1		1,0	
S_6			1		
S_7		1,0		1	
S_8	1,0				1

Таким чином, алгоритм знаходить найбільш схожі послідовності дій, групує їх разом (еквівалентує), а потім виключає їх з подальшого розгляду (або обнуляє їхні відповідні рядки/стовпці, або видаляє їх з матриці), щоб знайти наступну групу схожих послідовностей. Цю ітерацію алгоритму можна реалізувати я коді:

$$M_{ij},\ i,j{+}{+};\ \text{if } M_{ij} = 1{,}0 \text{ then } k{+}{+};\ E_k = j;\ M_i - \text{delete};\ M_j - \text{delete}. \qquad (2.5)$$

5. Виконується перехід до обробки наступного рядка матриці (після видалення рядків та стовпців завжди береться перший рядок). У цьому рядку шукається перша координата зі значенням 1. Процес повторюється до тих пір, поки всі координати матриці не будуть обнулені (викреслені). Це означає, що всі класи еквівалентних послідовностей, представлені в табл. 2.8, були успішно побудовані.

Таблиця 2.8 – Побудова всіх класів еквівалентостей

Sim	S_3	S_4	S_6
S_4	1		1,0
S_6		1	
S_7	1,0		1

Sim	S_6
S_6	1

6. Найпростіший алгоритм, що полягає у викреслюванні рядків і стовпців за суттєвими координатами, дозволяє згрупувати матрицю в класи еквівалентності: E={{S1, S2, S5}, {S3, S8}, {S4, S7}, {S6}}. Проте, використання двійкової матриці послідовностей дій для еквівалентування має обмеження: воно працює лише при повній ідентичності послідовностей. Якщо потрібна часткова подібність, необхідно визначити мінімальний рівень подібності, наприклад, 80%, для визначення еквівалентності послідовностей дій.

7. Перевірка правильності результату здійснюється шляхом переконання, що перетин будь-яких двох еквівалентних класів є порожньою множиною (тобто, вони не перетинаються). При цьому, об'єднання всіх еквівалентних класів повинно повністю охоплювати весь бізнес-потік.

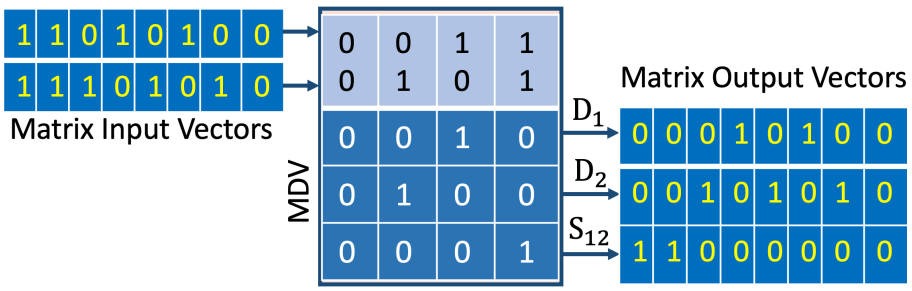
2.3 Матричний паралельний аналіз великих даних

Для швидкої класифікації великих обсягів даних, що надходять у вигляді потоку екранів комп'ютера, використовується метод матричного паралельного аналізу. Він дозволяє визначати схожість та відмінність між

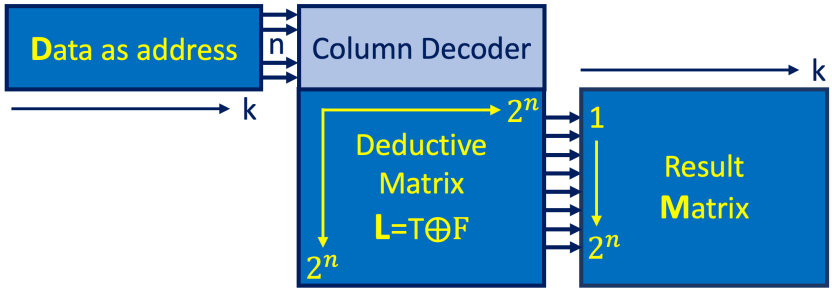
екранами, що важливо для аналізу бізнес-процесів. Вхідними даними є текст, розпізнаний за допомогою OCR на зображеннях екранів.

Математична модель використовує спеціальні вектори, які діють як фільтри, щоб визначити, наскільки схожі або різні зображення на екранах. Інформація на екранах кодується у вигляді двійкових векторів, які використовуються для вилучення відповідної інформації з цих векторів-фільтрів.

Паралельний аналіз двійкового вхідного сигналу (0100 1101 0001) за допомогою матриці трьох векторів (рис. 2.2) дозволяє одночасно обчислювати вектори відмінностей (D1, D2) та подібності (S12). Для адресації стовпців матриці використовується комбінація сигналів з першого та другого вхідних векторів, що формує код стовпця (00 01 10 11). Цей код паралельно вибирає відповідний стовпець векторної матриці, що генерує вихідні вектори.



а



б

Рисунок 2.2 – Матричний паралельний аналіз великих даних як адрес: перетворення вхідних даних; б – структура аналізу

Операція логічного добутку для трьох векторів (D1∧D2∧S12)=000000000000 дає нульовий вектор, при цьому диз'юнкція вхідних векторів дає результат: (D1∨D2∨S12) =111001111110. Обробка векторних даних далі здійснюється за допомогою матричних обчислень з великими обсягами даних.

На рис. 2.2,б показано, як працює матричний аналіз великих даних для визначення подібності. Він обчислює різницю між вхідними векторами, які потім потрапляють у матричний секвенсор. Кількість входів цього секвенсора, що відповідає числу рядків у матриці D, завжди збігається з кількістю паралельних потоків даних, які потрібно обробити.

Кожен рядок матриці L, завдяки своїм одиничним значенням, задає певну функцію обробки вхідних даних, наприклад, визначає схожість або відмінність, і використовує їх як адреси. Вся матриця L, яку можна назвати дедуктивною матрицею, формує метрику властивостей. Кількість цих властивостей дорівнює кількості виходів секвенсора (матриці M). Іншими словами, кількість векторів у матрицях L та M є однаковою.

Можна розглядати такий комп'ютинг як xor-відношення трьох матриць [14]:

$$T \oplus F \oplus L = 0, \tag{2.6}$$

де T – вхідні дані (Data as Address), F – функціональність (Deductive Matrix L), M – результат аналізу (Result Matrix), що також можна подати наступними формулами:

$$L = T \oplus F; \tag{2.7}$$

$$M = F_T. \tag{2.8}$$

Формула (2.7) визначає спосіб побудови матриці векторного дедуктивного аналізу. Формула (2.8) інтерпретується як комп'ютинг транзакцій на пам'яти, де великі дані розглядаються як адреси: $M=L_D$. Отже,

загальне рівняння векторного дедуктивного матричного комп’ютингу для паралельного аналізу великих даних має вигляд:

$$M=(T \oplus F)_D. \quad (2.9)$$

2.4 Алгоритм класифікації

Алгоритм складається з наступних пунктів:

1) вибір наступної послідовності дій для знаходження спільного елементу з іншими послідовностями. Цей спільний елемент (непорожній перетин) стає основою (ядром) нової групи;

2) об'єднання унікальних елементів різних послідовностей в межах однієї групи. Це досягається шляхом знаходження симетричної різниці між цими послідовностями;

3) перехід до наступної послідовності в списку, що залишився, і повторення процесів пошуку спільного елементу та об'єднання унікальних елементів для формування нової групи.

Фактично, алгоритм працює у два етапи: спочатку він групує рядки-послідовності, знаходячи найбільш схожі на них рядки з таблиці істинності (схожість має бути не менше 80%); потім для кожної групи алгоритм виявляє, чим саме відрізняються рядки один від одного, використовуючи операцію XOR для порівняння їхніх послідовностей дій.

2.5 Алгоритм створення універсуму

2.5.1 Алгоритм для створення примітивів (символів) складається з наступних етапів [13].

1. Обробка назви дії (Activity Name): кожна назва дії перевіряється на наявність у списку унікальних примітивів (U). Якщо назва дії нова, вона

додається до списку U, і їй присвоюється наступний порядковий номер у цьому списку.

2. Присвоєння номера дії вектору: цей унікальний номер, отриманий з U, присвоюється відповідній назві дії у векторі P. Вектор P містить номери примітивів, і його довжина відповідає кількості рядків у таблиці бізнес-процесу.

3. Повторне використання існуючих номерів: якщо назва дії вже існує у таблиці бізнес-функціональності, їй присвоюється відповідний номер з універсуму примітивів U, який був присвоєний їй раніше. Процес присвоєння номерів з U триває до тих пір, поки кожній унікальній назві дії не буде присвоєно її унікальний номер примітиву.

Після визначення базових елементів (універсуму примітивів) та вектора ідентифікації P, що описує етапи бізнес-процесу, будується двійкова матриця M(ij)=P(k). Кожен рядок цієї матриці представляє окрему послідовність дій. Одиниці в рядку вказують на використання певних базових елементів, згідно з вектором P, в цій послідовності. Початок і кінець кожної послідовності визначаються відповідними ідентифікаторами.

У подальшому матриця використовується для знаходження схожих послідовностей, що об'єднуються в класи еквівалентності. В межах кожного класу виявляються унікальні характеристики кожної послідовності, що дозволяють відрізнити їх за функціональністю.

При аналізі послідовностей текстових фрагментів необхідно брати до уваги часові мітки їх створення. Матриця представляє собою універсальний простір, де кожна координата відповідає парі "фрагмент-послідовність". Значенням кожної координати є сумарний час, витрачений на генерацію відповідного текстового фрагменту в контексті даної послідовності. У табл. 2.9 зображено матрицю універсуму та послідовності.

Таблиця 2.9 – Таблиця матриць універсуму/послідовності

G=	A	B	C	D	E	F	K	L	M	N	P	T	1	2	3	4	5	6	7	8
U=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
S_1	2	4	7										5							
S_2	4	3	2											5						
S_5	4	5	5												6					
S_3				3	5	4										7				
S_8				2	5	7											4			
S_4							6	7	7									3		
S_7							5	4	3										6	
S_6										5	3	7								3

При цьому кожна координата визначається сумою часу формування літер/рядків у всіх фрагментах однієї послідовності

$$\mathbf{M_{ij}} = \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{a_j}\ , \tag{2.10}$$

де $a_j \in G_i$. Оцінка схожості між рядками матриці базується на перетвореній формулі подібності, спеціально адаптованій для двійкових матриць:

$$S(S_i, S_j) = \sum_{i=1}^{n}(S_i \wedge_{i=\overline{1,n}} S_j) / \sum_{i=1}^{n}(S_i \vee_{i=\overline{1,n}} S_j) \approx$$

$$\approx S(S_i, S_j) = \sum_{i=1}^{n}[\min_{i=\overline{1,n}}(S_j, S_i)] / \sum_{i=1}^{n}[\max_{i=\overline{1,n}}(S_j, S_i)] \tag{2.11}$$

Приклад 2.1. Визначення подібності між першою парою рядків матриці:

$$S(S_i, S_j) = \sum_{i=1}^{n}[\min_{i=\overline{1,n}}(2,4)(4,3)(7,2)(5,0)(0,5)] /$$

$$/ \sum_{i=1}^{n}[\max_{i=\overline{1,n}}(2,4)(4,3)(7,2)(5,0)(0,5)] =$$

$$= \sum_{i=1}^{n}[\min_{i=\overline{1,n}}(2)(3)(2)(0)(0)] / \sum_{i=1}^{n}[\max_{i=\overline{1,n}}(4)(4)(7)(5)(5)] = 7/25 = 0{,}28$$

Літера, що ідентифікує символ-рядок, який є ключовим для формування послідовності дій (навіть якщо він з'являється на короткий час), отримує найвищий пріоритет у матриці бізнес-послідовностей, що відображається через максимальну вагу.

2.5.2 При створенні універсуму як показника бізнес-процесу, кожен символ кодується унікальним десятковим числом (рис. 2.3). Потім, для кожного екрана застосовується унітарне кодування до отриманих числових представлень символів.

1	2	3	4	5	6	7	8	9	10	11	12
ма	he	ye	in	on	no	do	go	di	ha	qu	me
1	1	0	0	0	1	0	0	0	1	0	0
1	1	1	0	0	0	1	1	1	0	1	0

Рисунок 2.3 – Вектор універсуму примитивів та унитарне кодування символів екрана

Спочатку кожен набір символів на екрані перетворюється на двійковий вектор. У цьому векторі "1" вказує на позицію, де знаходиться символ на екрані, який кодується. Ці вектори, об'єднані у вхідну матрицю MIV – Matrix Input Vectors, використовуються для подальшого аналізу. Вхідна матриця подається на матрицю дедуктивних векторів (MDV – Matrix Deductive Vectors).

Стовпці вхідної матриці використовуються як адреси для вибору відповідних стовпців з MDV, які потім формують стовпці вихідної матриці. Якщо потрібно одночасно проаналізувати, наприклад, 8 екранів, то створюється матриця для 8 входів, причому кожен дедуктивний вектор матиме певну розмірність: $\mathbf{2}^{\boldsymbol{n}}$=256 бит.

2.6 Висновки до розділу 2

Проаналізовано основні моделі та проаналізовано алгоритми пошуку та ідентифікації зразків (шаблонів), класифікації та створення універсуму примітивів. Модель для синтезу логіки алгоритму пошуку та ідентифікації є вихідною або первісною матрицею, форматованою в метриці параметрів.

3 СИСТЕМА РОЗПІЗНАВАННЯ ЗРАЗКІВ НА ОСНОВІ ВЕКТОРНОГО МЕТОДУ

Виконується реалізація методу пошуку та розпізнавання зразків (шаблонів) на основі векторного завдання символьних послідовностей, які ідентифікують бізнес-зразків у потоках великих даних.

Як початкові дані використовуються потокові файли бізнес-процесу через конкретні часові інтервали, що містять певні шаблони формування. Їх важливість визначається часовими інтервалами та компонентами, передбаченими завдання вручну та незначний час виконання.

Завдання:

1) формування моделі як структури даних для опису потоку з невизначеними сукупностями зразків та фрагментів з початковими та кінцевими ідентифікаторами;

2) розробка унітарно кодованих матричних моделей як визначених векторів шляхом прямого добутку сукупностей послідовності;

3) інтеграція логіки алгоритму для пошуку, ідентифікації та класифікації зразків, визначення початкових даних у прийнятній формі та супутній статистиці;

4) верифікація знайдених шаблонів, виявлених на основі перетинання та об’єднання;

5) тестування програму, прототипування алгоритмічного коду та опис структур даних.

3.1 Визначення класів еквівалентностей за макро-алгоритмом

1. Визначення алфавіту як універсальної множини, що утворює стовпчик матриць бізнес-процесів З цією метою переглядається весь потік даних та відокремлюється лише сукупність оригінальних символів.

2. Етап заповнення наступної матричного рядка з перебором символів у фреймах дій для формування часових визначень для кожного символу за адитивним правилом, який входить до інтегральної функціональності послідовності:

$$M_{ij}=M_{ij}+t(G_k)\leftarrow G_k=a_j, \quad (3.1)$$

де $a_j \in U$ уявляє собою символ-рядок, що форує універсум та одночасно належить послідовност $a_j \in G_i$, яка формує координату «j» в рядку «i» матриці.

3. На етапі побудови квадратичної матриці подібності використовується мінімальна метрика для розрахунку коефіцієнту схожості (Similarity).

4. Для виявлення класів еквівалентних послідовностей здійснюється пошук найбільших показників подібності між рядками, що представляють бізнес-процеси.

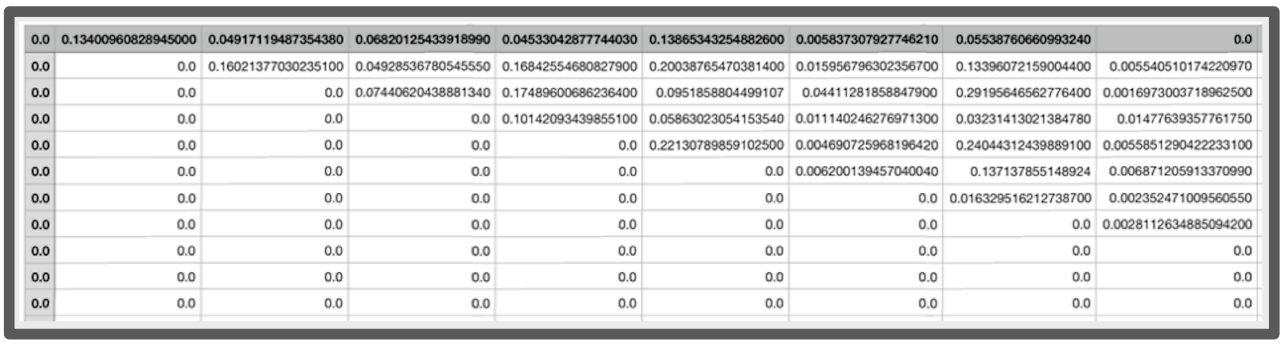
5. Етап перевірки, де класи еквівалентності піддаються операції перетину та об'єднання, яка повинна призвести до відсутності елементів (порожньої множини) та відповідати бізнес-потоку (універсуму) відповідно.

Розроблений програмний код згенерував сотню послідовностей, що містять дві тисячі базових символів, серед яких є порожні за часом. Для підвищення ефективності класифікації, потрібно зменшити обсяг алфавіту, виключивши символи, несуттєві для аналізу.

3.2 Алгоритми визначення кластерів

Алгоритм визначення кластерів як підмножини послідовностей операцій з універсуму примітивів на основі аналізу потоку даних використовує усічену кількість атрибутів.

1. Синтез матриці подібності послідовностей дій (процесів) (рис. 3.1).



0.0	0.13400960828945000	0.04917119487354380	0.06820125433918990	0.04533042877744030	0.13865343254882600	0.00583730792774621 0	0.05538760660993240	0.0
0.0	0.0	0.16021377030235100	0.04928536780545550	0.16842554680827900	0.20038765470381400	0.01595679630235670 0	0.13396072159004400	0.00554051017422097 0
0.0	0.0	0.0	0.07440620438881340	0.17489600686236400	0.0951858804499107	0.0441128185884790 0	0.29195646562776400	0.00169730037189625 00
0.0	0.0	0.0	0.0	0.10142093439855100	0.05863023054153540	0.0111402462769713 00	0.03231413021384780	0.0147763935776175 0
0.0	0.0	0.0	0.0	0.0	0.22130789859102500	0.0046907259681964 20	0.24044312439889100	0.00558512904222331 00
0.0	0.0	0.0	0.0	0.0	0.0	0.0062001394570400 40	0.13713785514892 4	0.00687120591337099 0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01632951621273870 0	0.00235247100956055 0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00281126348850942 00
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Рисунок 3.1 – Результати синтезу матриці подібності послідовностей дій

2. Видалення символів X, повторення побудови матриці подібності. Склеювання однакових цифрових ідентифікаторів послідовностей дій.

3. Розподіл потоку за днями та операторами. Повторення синтезу матриць подібностей.

4. Аналіз можливості зменшення символів алфавіту за рахунок видалення несуттєвих диференціаторів.

5. Повторення побудови матриць подібностей на мінімальній множині символів.

Результат виявлення подібності за часом дає 14 кластерів (рис. 3.2).

[[0, 1, 5, 10, 18, 24, 25, 27, 28, 29, 30, 31, 34, 35, 36, 37, 38, 43, 44, 48, 49, 54, 56, 57, 61, 62, 63, 64, 65, 69, 71, 72, 74, 75, 76, 81, 82, 84, 86, 87, 88, 90, 93, 100], [], [2, 4, 7 , 9, 13, 16, 19, 20, 21, 22, 23, 55, 60, 70, 77, 78, 80, 85], [3, 11, 12, 33, 42, 45, 52, 59, 66 , 67, 68, 79, 83, 89, 91, 94, 99], [], [], [6], [], [8], [], [], [], [], [], [14, 15, 32, 41, 53, 92], [], [], [17, 95, 96, 97, 98], [], [], [], [], [], [], [], [], [26], [], [], [], [], [], [], [], [], [], [], [], [39], [40], [], [], [], [], [], [46], [47], [], [], [50], [51], [], [], [], [], [], [], [58], [], [], [], [], [], [], [], [], [], [], [], [], [ ], [], [73], [], [], [], [], [], [], [], [], [], [], [], [] , [], [], [], [], [], [], [], [], [], [], [], [], []]

Рисунок 3.2 –Результати кластеризації

Перевірка кількості послідовностей дій. Верифікація за потоком подібності послідовностей дій, що потрапили у кластери. Інтегральний час

кожної послідовності може слугувати інструментом для виявлення кластерів. Якщо не враховувати тривалість виконання кожного символу, оцінки схожості між послідовностями значно покращяться. Необхідно спробувати визначити подібність за допомогою двійкової матриці послідовностей дій, а потім перевірити результати, враховуючи час виконання послідовності. Слід зменшити кількість атрибутів до двох (усунути хвости та зменшити розмір алфавіту символів) і підрахувати кількість кластерів. Це також призведе до підвищення оцінки подібності.

Алгоритм для визначення кластерів як підмножини літер з універсуму примітивів, що ґрунтується на аналізі потоку даних із застосуванням обмеженої кількості атрибутів:

1. Формується граф кореляції всіх пар літер з універсуму примітивів, які є сусідніми в обраному вікні бізнес-потоку. При цьому сусідніми можуть виявитися літери, що належать різним функціональностям.

2. Створюється квадратна матриця, яка відображає частоту зустрічальності сусідніх символів у поточному потоці даних, а не ступінь їхньої схожості. Важливим є те, що взаємодія між літерами, наприклад, А та Б у послідовності А…Б, може бути проігнорована, якщо ці літери опиняються за межами аналізованого фрагменту.

3. Для виявлення прихованих зразків дій, що утворюють кластери, застосовується спектральний аналіз квадратичної матриці. Отримані кластери є взаємовиключними (не перетинаються). Цей метод може бути оптимальним для кластеризації бізнес-функціональностей, представлених у вигляді послідовностей елементів, що моделюють щоденний робочий процес.

Бібліотека складається з 350 символів, що визначають 331 кластер. Оптимально, щоб обидва методи (рис. 3.3) виділяли однакові літери в межах одного кластера. Іншими словами, бажано, щоб результати обох методів збігалися щодо набору літер, які вони відносять до певного кластера.

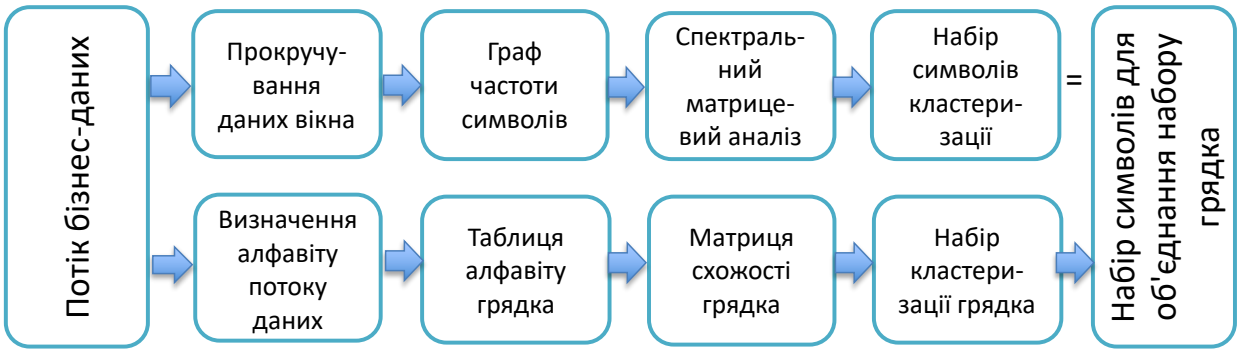


Рисунок 3.3 – Аналіз двох методів кластеризації

На виході алгоритму в першому випадку утворюється сукупність неперетинних підмножин символів, які відповідають основним функціональностям, закодованим у бізнес-потоці даних. У другому випадку здійснюється кластеризація послідовностей або бізнес-патернів, де кожен кластер містить подібні за функціональностями підмножини літер, які їх складають.

Якщо часові параметри алфавіту замінити на двійкові символи, значення в квадратичній матриці подібності послідовностей дій між собою суттєво зросте. Внаслідок аналізу цієї матриці було виявлено 44 кластери на повній множині послідовностей дій, кількість яких становить 100, при цьому відсутня видима кореляція часу виконання послідовностей в межах кожного кластеру. Необхідно розрахувати інтегральний час для кожної послідовності аби оцінити їхню кореляцію в кожному кластері. Інтегральний час повинен бути порівняно однорідним для всіх послідовностей в одному кластері.

3.3 Опис програмного засобу

Слід прибрати всі несуттєві компоненти для зменшення алфавіту літер за рахунок включення лише 3 атрибутів.

Для кожного з атрибутів слід призначити вагові коефіцієнти, що утворюють алфавіт відповідно до ієрархії, а саме: $\mathbf{2^4, 2^3, 2^2, 2^1, 2^0}$. Вагові коефіцієнти присвоюються згідно до порядку розташування символів:

атрибуту на першій позиції присвоюється максимальна вага, на п'ятій – мінімальна.

Розрахунок подібностей між всіма послідовностями дій потоку даних виконується за формулою (2.6).

Визначення кластерів на базі отриманого алфавіту з урахуванням вагових коефіцієнтів передбачає створення матриці частот певної обмеженої кількості знаків. Наступним кроком є формування матриці подібності. Очікується, що такий підхід виявиться ефективним, оскільки ліва частина атрибутів зустрічається частіше за праву. Це дозволяє використовувати частоту як метрику для визначення ієрархічних рівнів: чим вища частота, тим більше значення атрибута і його місце в ієрархії в послідовності дій буде вище. Координата j матриці має інкремент, якщо символ G_{ir} із сукупності атрибутів послідовності дорівнює букві алфавіту A_i на універсумі визначених примітивів:

$$M_{ij}=M_{ij}+1 \leftarrow (G_{ir})=A_j \, . \qquad (3.2)$$

Перевірка інтегрального часу для виконання послідовності дій без ідентифікаторів.

У табл. 3.1 наведено дані кореляції векторів примітивів у метриці за нормою (0–1).

Для формування послідовності дій застосовується бінарна матриця слів. Загальне число рядків у потоці даних – 19939. Кількість послідовностей дій – 100. Алфавіт слів = [176]. Результати для Classes=45 (sim>0.6) та Classes=28 (sim>0.5) наведено на рис. 3.4 і 3.5 відповідно.

Таблиця 3.1 – Кореляція векторів примітивів

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1.000	0.737	0.500	0.476	0.385	0.370	0.186	0.500	0.105	0.500	0.609	0.444	0.318	0.464	0.364	0.263	0.500	0.375	0.158	0.357	0.565
2	0.737	1.000	0.571	0.579	0.522	0.500	0.188	0.640	0.176	0.640	0.800	0.462	0.273	0.600	0.450	0.353	0.577	0.391	0.167	0.480	0.750
3	0.500	0.571	1.000	0.448	0.567	0.548	0.306	0.559	0.143	0.606	0.600	0.429	0.290	0.733	0.414	0.250	0.606	0.333	0.138	0.531	0.516
4	0.476	0.579	0.448	1.000	0.571	0.545	0.176	0.500	0.286	0.500	0.619	0.440	0.444	0.520	0.687	0.500	0.444	0.500	0.187	0.591	0.500
5	0.385	0.522	0.567	0.571	1.000	0.773	0.197	0.517	0.211	0.517	0.560	0.323	0.348	0.593	0.455	0.300	0.630	0.400	0.200	0.739	0.520
6	0.370	0.500	0.548	0.545	0.773	1.000	0.211	0.552	0.200	0.500	0.538	0.312	0.333	0.571	0.435	0.286	0.552	0.385	0.190	0.708	0.444
7	0.186	0.188	0.306	0.176	0.197	0.211	1.000	0.300	0.061	0.247	0.211	0.313	0.164	0.268	0.197	0.106	0.282	0.206	0.060	0.208	0.197
8	0.500	0.640	0.559	0.500	0.517	0.552	0.300	1.000	0.160	0.724	0.667	0.469	0.276	0.581	0.462	0.280	0.667	0.367	0.154	0.484	0.571
9	0.105	0.176	0.143	0.286	0.211	0.200	0.061	0.160	1.000	0.160	0.200	0.130	0.231	0.167	0.308	0.571	0.115	0.111	0.286	0.190	0.150
10	0.500	0.640	0.606	0.500	0.517	0.500	0.247	0.724	0.160	1.000	0.731	0.516	0.276	0.581	0.462	0.280	0.562	0.367	0.154	0.484	0.571
11	0.609	0.800	0.600	0.619	0.560	0.538	0.211	0.667	0.200	0.731	1.000	0.500	0.333	0.630	0.500	0.350	0.607	0.385	0.136	0.518	0.625
12	0.444	0.462	0.429	0.440	0.323	0.312	0.313	0.469	0.130	0.516	0.500	1.000	0.360	0.484	0.458	0.261	0.424	0.462	0.125	0.303	0.414
13	0.318	0.273	0.290	0.444	0.348	0.333	0.164	0.276	0.231	0.276	0.333	0.360	1.000	0.333	0.471	0.267	0.276	0.474	0.133	0.320	0.240
14	0.464	0.600	0.733	0.520	0.593	0.571	0.268	0.581	0.167	0.581	0.630	0.484	0.333	1.000	0.480	0.292	0.633	0.333	0.160	0.552	0.593
15	0.364	0.450	0.414	0.687	0.455	0.435	0.197	0.462	0.308	0.462	0.500	0.458	0.471	0.480	1.000	0.538	0.357	0.450	0.200	0.417	0.391
16	0.263	0.353	0.250	0.500	0.300	0.286	0.106	0.280	0.571	0.280	0.350	0.261	0.267	0.292	0.538	1.000	0.231	0.211	0.333	0.273	0.300
17	0.500	0.577	0.606	0.444	0.630	0.552	0.282	0.667	0.115	0.562	0.607	0.424	0.276	0.633	0.357	0.231	1.000	0.323	0.154	0.533	0.571
18	0.375	0.391	0.333	0.500	0.400	0.385	0.206	0.367	0.111	0.367	0.385	0.462	0.474	0.333	0.450	0.211	0.323	1.000	0.167	0.423	0.296
19	0.158	0.167	0.138	0.187	0.200	0.190	0.060	0.154	0.286	0.154	0.136	0.125	0.133	0.160	0.200	0.333	0.154	0.167	1.000	0.182	0.200
20	0.357	0.480	0.531	0.591	0.739	0.708	0.208	0.484	0.190	0.484	0.518	0.303	0.320	0.552	0.417	0.273	0.533	0.423	0.182	1.000	0.481
21	0.565	0.750	0.516	0.500	0.520	0.444	0.197	0.571	0.150	0.571	0.625	0.414	0.240	0.593	0.391	0.300	0.571	0.296	0.200	0.481	1.000
22	0.560	0.727	0.613	0.500	0.518	0.500	0.239	0.621	0.182	0.621	0.680	0.517	0.259	0.586	0.400	0.318	0.567	0.357	0.174	0.536	0.708
23	0.440	0.591	0.567	0.737	0.727	0.696	0.214	0.571	0.211	0.571	0.625	0.367	0.348	0.593	0.524	0.368	0.571	0.458	0.200	0.739	0.520
24	0.500	0.640	0.606	0.560	0.517	0.667	0.247	0.667	0.160	0.613	0.667	0.424	0.276	0.581	0.407	0.280	0.562	0.414	0.154	0.533	0.571

[[0, 1, 10, 35, 43, 56, 58, 69], [], [2, 9, 13, 16, 21, 23, 60, 78], [3, 14, 22, 38, 59 , 66, 76, 80, 81, 85, 89, 91, 92], [4, 5, 19], [], [6], [7, 77], [8], [], [], [11], [12, 46], [], [], [15, 32, 45, 51], [], [17], [18], [], [20], [], [], [ ], [24, 48], [25, 42, 53, 61], [26, 27, 29, 30, 36], [], [28, 37], [], [], [31, 55] , [], [33, 34, 54], [], [], [], [], [], [39, 79], [40], [41, 47], [], [], [44], [], [], [], [], [49], [50], [], [52, 71], [], [], [], [], [57], [] , [], [], [], [62, 87], [63], [64], [65], [], [67, 68], [], [], [70, 72], [ ], [], [73], [74], [75, 94], [], [], [], [], [], [], [82], [83], [84], [ ], [86], [], [88], [], [90], [], [], [93, 95, 96, 97, 98], [], [], [], [], [], [99], [100]]

Рисунок 3.4 – Результати кластеризації для Classes=45 (sim>0.6)

[[0, 1, 10, 20, 21, 30, 34, 35, 36, 38, 43, 44, 56, 58, 60, 61, 69, 71, 72, 76, 77, 84, 88], [ ], [2, 4, 5, 7, 9, 13, 16, 19, 22, 23, 40, 78], [3, 14, 25, 41, 46, 47, 48, 59, 66, 67, 68 , 75, 79, 80, 81, 85, 89, 91, 92], [], [], [6], [], [8, 15], [], [], [11, 42, 90] , [12], [], [], [], [], [17], [18], [], [], [], [], [], [24, 28, 29, 37], [], [26, 27, 33, 87], [], [], [], [], [31, 54, 55], [32, 45, 51, 53], [], [], [ ], [], [], [], [39], [], [], [], [], [], [], [], [], [], [49, 93, 95, 96 , 97, 98], [50], [], [52], [], [], [], [], [57], [], [], [], [], [62], [63, 64], [], [65],[], [], [], [], [70, 100], [], [], [73], [74], [], [], [], [], [], [], [], [82, 83], [], [], [], [86], [], [], [], [], [], [ ], [], [94], [], [], [], [], [99], []]

Рисунок 3.5 – Результати кластеризації для Classes=28 (sim>0.5)

Здійснити верифікацію кластеризації можна через аналіз схожості n рядків кожної послідовності дій, що входить до складу кластера. Кластери, сформовані на основі n рядків, служать точним рішенням і виступають як своєрідна специфікація. Тому процедура накладання двох матриць подібності квадратної форми дозволяє визначити якість кластеризації. Це можна зробити шляхом обчислення інтегрального показника схожості для всіх відповідних координат цих двох матриць:

$$S(M, E) = \sum_{i,j=1}^{n}[\min_{i=1,n}(M_{ij}, E_{ij})] \,/\, \sum_{ij=1}^{n}[\max_{i=1,n}(M_{ij}, E_{ij})] \,. \qquad (3.3)$$

З використанням нижньої межі подібності можна привести матриці до двійкового вигляду, що формується евристично min={0,4 ∨ 0,5 ∨ 0,6}:

If (M_{ij}>min) (M_{ij}=1) else (M_{ij}=0),

If (E_{ij}>min) (E_{ij}=1) else (E_{ij}=0).

У двійковому форматі оцінка подібності є більш адекватною та контрастною, що створює схожість між обома матрицями:

$$S(M,E) = \sum_{i,j=1}^{n}(M_{ij} \wedge E_{ij}) \,/\, \sum_{ij=1}^{n}(M_{ij} \vee E_{ij}). \qquad (3.4)$$

У випадку двійкової S-матриці подібності, критерій S(M,E) використовується для визначення точності кластеризації послідовностей дій, які описуються специфікацією з n рядків. Подібність між матрицями обчислюється за допомогою певної формули, що базується на їх різниці:

$$S(M,E) = 1 - D(M,E) =$$

$$= 1 - \sum_{i,j=1}^{n}\left(M_{ij} \oplus E_{ij}\right) / \sum_{ij=1}^{n}\left(M_{ij} \vee E_{ij}\right) = 1 - \frac{15}{64} = 0{,}77. \qquad (3.4)$$

3.4 Автомат для визначення різниці у зразках

Рис. 3.6 демонструє структури матричних даних, що використовуються для ідентифікації відмінностей у зразках. Схема показує приклад паралельної та одночасної обробки п'яти зразків.

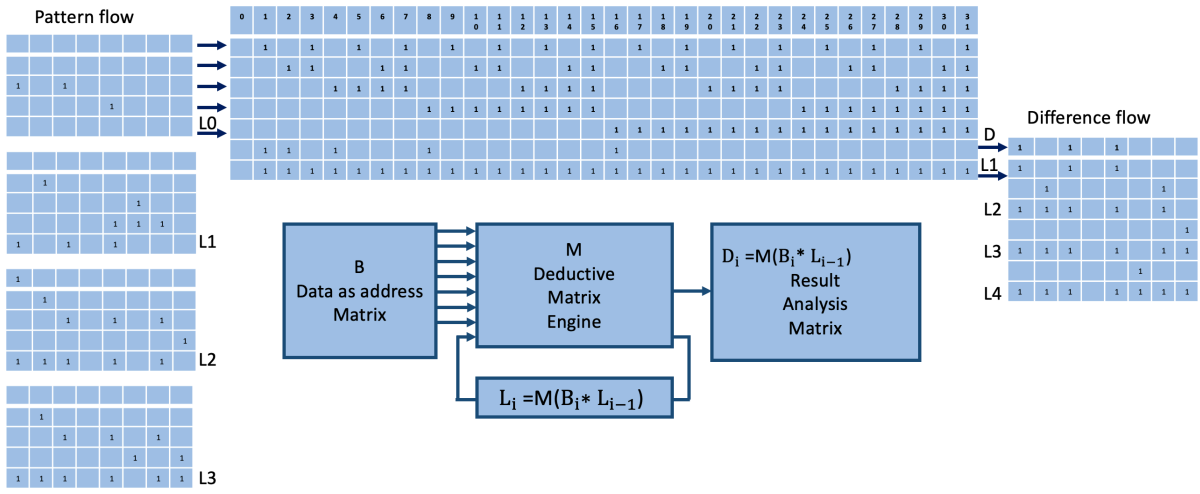


Рисунок 3.6 – Схема автомата для визначення різниці у зразках

Тут Deductive Matrix Engine реалізує паралельну обробку n=5 зразків для отримання Difference Analysis Matrix $\mathbf{D_i}$=M($\mathbf{B_i}$*$\mathbf{L_{i-1}}$), де останній вхід $\mathbf{L_i}$=M($\mathbf{B_i}$*$\mathbf{L_{i-1}}$) зв'язує або накопичує різниці у всіх вже оброблених зразках.

Конкатенація $\mathbf{B_i}$*$\mathbf{L_{i-1}}$ організовує стовпці, що містять адреси даних. Ці дані описуються чотирма (n=4) шаблонами B і вектором зв'язків L. Відсутність даних у певних місцях (порожні комірки) позначається нульовими значеннями у відповідних координатах векторів**.**

Вектор $\boldsymbol{L_{i-1}}$ виконує інтеграцію одиниць для усунення локальних відмінностей у зразку $\mathbf{B_i}$. Фрейм представляє собою кількість патернів (екранів), які обробляються одночасно і в паралельному режимі; у цьому випадку їх кількість становить чотири. Будь-який елемент, що з’являється у потоці великих даних лише один раз, буде позначений як унікальний. Після завершення аналізу потокових даних вектор зв’язку L буде дорівнювати одиниці для всіх координат, а кінцевий вектор D матиме число 1-координат у метриці примітивного універсуму, які ідентифікують глобальні відмінності в усіх даних. Слід зазначити, що під час аналізу великих даних у векторі D формуються локальні відмінності на проміжку від 1 до і кадрів. Звичайно, з ростом значення і – кількості оброблених фреймів – зменшується кількість виявлених відмінностей за метрикою примітивного універсуму.

3.5 Висновки до розділу 3

Реалізовано метод пошуку та розпізнавання шаблонів, заснований на використанні векторного подання для символьних послідовностей, що ідентифікують бізнес-зразки у потоках великих даних. Завдяки практичній реалізації векторного методу створена програма, яка розпізнає 77% зразків у потоках великих даних.

Запропоновано структуру детермінованого автомату, який дозволяє розпізнавати 100% різниці у зразках.

ВИСНОВКИ

1. Проаналізовано технологічні тенденції, що визначені компанією Gartner за 2021-2024 роки. Проведено аналіз сучасних досліджень та публікацій. Наголошено, що дедуктивні методи, які мають апріорну точність логічних виразів або аналітичних формул, є кращими для їх імплементації в практику, ніж ймовірнісні методи ШІ, що вимагають тривалого навчання та верифікації для впровадження у виробничі процеси, незважаючи на їх затребуваність.

2. Реалізовано метод пошуку та розпізнавання зразків (шаблонів) на основі векторного подання символьних послідовностей, які ідентифікують бізнес-зразки у потоках великих даних. Метод використовує унітарне кодування примітивів та даних, які у подальшому подаються як послідовності або двійкові вектори, що описує бізнес-зразок або функціональність, виконану оператором протягом робочого дня. Структури даних, засновані на унітарному кодуванні інформації, можуть бути оброблені в паралельному режимі за допомогою основних логічних операцій, що формують подібності та відмінності зразків-векторів, чим досягається висока продуктивність обробки великих даних з метою аналізу робочого дня оператора.

3. Розроблено систему розпізнавання зразків, що містить реалізацію векторного подання символьних послідовностей (векторного методу), унітарно кодовану матричну модель як декартів добуток множин-примітивів, синтезовану логіку алгоритмів для пошуку, ідентифікації та класифікації зразків (шаблонів); схему автомату для визначення різниці у зразках; тестування програми. Векторний метод дозволяє розпізнавати 77% зразків у потоках великих даних. Запропоновано структуру детермінованого автомату, який дозволяє розпізнавати 100% різниці у зразках.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Gartner Top 6 Trends Impacting Infrastructure & Operations in 2021 [https://www.gartner.com/smarterwithgartner/gartner-top-6-trends-impacting-infrastructure-operations-in-2021]

2. Gartner Top 10 Strategic Technology Trends for 2024 [https://www.gartner.com/en/articles/gartner-top-10-strategic-technology-trends-for-2024]

3. Joshi A. Machine Learning and Artificial Intelligence / A. Joshi // Springer Nature Switzerland: AG, 2020. – 261 p. Doi: 10.1007/978-3-030-26622-6

4. IEEE Guide for Architectural Framework and Application of Federated Machine Learning // *IEEE Std 3652.1-2020*. – 19 March 2021. – P.1-69. doi: 10.1109/IEEESTD.2021.9382202.

5. SplitStreams: A Visual Metaphor for Evolving Hierarchies / F. Bolte, M. Nourani, E. D. Ragan and S. Bruckner // *IEEE Transactions on Visualization and Computer Graphics*. – 1 Aug. 2021. – Vol. 27, no. 8. – P. 3571-3584. doi: 10.1109/TVCG.2020.2973564.

6. Multiscale Control Chart Pattern Recognition Using Histogram-Based Representation of Value and Zero-Crossing Rate / J.-W. Huang, P. -J. Lee and B. P. Jaysawal // *IEEE Transactions on Industrial Electronics*. – Jan. 2022. – Vol. 69, no. 1. – P. 684-693. doi: 10.1109/TIE.2021.3050355.

7. Proceed From Known to Unknown: Jamming Pattern Recognition Under Open-Set Setting / H. Han, W. Li, Z. Feng, G. Fang, Y. Xu and Y. Xu // *IEEE Wireless Communications Letters*. – April 2022. – Vol. 11, no. 4 – P. 693-697. doi: 10.1109/LWC.2021.3140145.

8. Weakly-Supervised Facial Expression Recognition в Wild with Noisy Data / F. Zhang, M. Xu, C. Xu // IEEE Transactions on Multimedia. – 2022. – Vol. 24. – P. 1800-1814. doi: 10.1109/TMM.2021.3072786.

9. McNeely C.L., Schintle L.A. Big Data Concept / C.L. McNeely, L.A. Schintler // In: Schintler, L.A., McNeely C.L. (eds) Encyclopedia of Big Data. Springer: Cham, 2022. – https://doi.org/10.1007/978-3-319-32001-4_551-3.

10. Similarity–Difference Analysis and Matrix Fault Diagnosis of SoC-components / [V. Hahanov, M. Karavay, V. Sergienko et al.] // East-West Design & Test Symposium (EWDTS), Varna, Bulgaria, 4-6 September 2020: proceedings. – IEEE, 2020. – P. 47-51. https://doi.org/10.1109/ewdts50664.2020.9224740

11. Big Data Critical Computing Based on the Similarity-Difference Metric / [V. H. Abdullayev, L. Shapa, V. Hahanov et al.] // East-West Design & Test Symposium (EWDTS), Varna, Bulgaria, 4-6 September 2020: proceedings. – IEEE, 2020. – P. 1-6. https://doi.org/10.1109/ewdts50664.2020.9225110

12. Hacimahmud V. Vector logic analysis of big data / V. Hacimahmud , H. Khakhanova, E. Litvinova // East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 22-25 September 2023: proceedings. – IEEE, 2023. – P. 1–4. https://doi.org/10.1109/ewdts59469.2023.10297032

13. Хаханова, Г.В. Векторний метод пошуку послідовностей у великих даних / Г.В. Хаханова // Сучасні інформаційні системи. – 2022. – № 6(3). – С. 13–22. https://doi.org/10.20998/2522-9052.2022.3.02

14. Faults-as-Address Simulation / V. Hahanov, S. Chumachenko, E. Litvinova, I. Hahanov, V. Ponomarova, H. Khakhanova, G. Kulak // IAES International Journal of Robotics and Automation (IJRA). – December 1, 2024. – Vol. 13, no. 4. – P. 452-468. https://doi.org/10.11591/ijra.v13i4.pp452-468.