Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти: другий (магістерський)

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ У НЕСТРУКТУРОВАНИХ НАБОРАХ ДАНИХ З ДАТЧИКІВ

(тема)

Виконав:
студент 2 курсу, групи ІНФм-18-2
Трембовецький Є.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

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

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Кузьомін О.Я.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Путятін Є.П.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Трембовецькому Єгору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: дослідження та розробка інформаційно-пошукової системи у неструктурованих наборах даних з датчиків

затверджена наказом по університету від «21» жовтня 2019 року № 1506Ст

2. Термін подання студентом роботи до екзаменаційної комісії 26 листопада 2019 р.

3. Вихідні дані до роботи *Дані з часовими послідовностями*
теоретичні відомості про методи аналізу часових послідовностей

4. Перелік питань, що потрібно опрацювати в роботі
1. Визначення переліку властивостей індексування
2. Визначення алгоритмів для порівняння послідовностей
3. Визначення структури даних індексу
4. Локація та класифікація аномалій

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів): діаграми часових послідовностей, діаграми вартостей і теплові діаграми, діаграми оптимального шляху, лістинги алгоритмів, зображення перетворення часових послідовностей, графіки залежностей параметрів експериментів, структурні діаграми, діаграми класів, зображення представлення даних, схематичне зображення структур даних.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	21.10.2019	
2	Аналіз завдання, підбір літератури	22.10.19-23.10.19	
3	Аналіз літератури з досліджуваної проблеми	24.10.19-29.10.19	
4	Аналіз технічних засобів	30.10.19-30.10.19	
5	Розробка методу	01.11.19-10.11.19	
6	Програмна реалізація	11.11.19-17.11.19	
7	Оформлення пояснювальної записки	18.11.19-26.11.19	
8	Перевірка на плагіат	09.12.19	
9	Рецензування	09.12.19	
10	Підготовка презентації та доповіді	09.12.19	
11	Занесення роботи в електронний архів	10.12.19	
12	Попередній захист атестаційної роботи	11.12.19	

Дата видачі завдання 21 жовтня 2019 р.

Студент _____
(підпис)

Керівник роботи _____ проф.Кузьомін О.Я.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до атестаційної роботи: 95 с., 2 табл., 31 форм., 58 рис., 1 дод., 40 джерел.

ЧАСОВІ ПОСЛІДОВНОСТІ, ВИТЯГНЕННЯ ВЛАСТИВОСТЕЙ, ПОШУКОВА СИСТЕМА, ПОРІВНЯННЯ ЧАСОВИХ ПОСЛІДОВНОСТЕЙ, ДЕТЕКТУВАННЯ АНОМАЛІЙ.

Метою роботи є розробка методів, що здатні ефективно індексувати часові послідовності для створення системи пошуку у багатовимірних часових послідовностей.

Об’єктом дослідження є дані експериментів забраних на верстатах для обробки металів. Був проведений аналіз та обґрунтування отриманих результатів, а також був зроблений висновок щодо доцільності використання даної системи.

У результаті атестаційної роботи було розроблено систему для пошуку часових послідовностей та детектування типу пересувного засобу. Був здійснений огляд, розбір та аналіз існуючих методів пошуку, індексування та представлення даних.

TIME SERIES, PROPERTIES EXTRACTION, INFORMATION RETRIWVAL SYSTEM, COMPARISON OF TIME SERIES, ANOMALY DETECTION.

The purpose of the work is to develop methods that can effectively index time series to create a search engine in multidimensional time series.

The object of the study is the data of experiments taken away on machine tools for metalworking. The results were analyzed and substantiated, and a conclusion was drawn as to the feasibility of using this system.

As a result of the appraisal work, a system was developed to search for time sequences and detect the type of vehicle. The existing search, indexing and presentation methods were reviewed, parsed and analyzed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SDA - shape description alphabet - алфавіт опису форми

DFT - discrete Furier transformation - дискретне перетворення Фур'є

DTW - dynamic time warping - динамічне викривлення часу

LCSS - longest common subsequence - найдовша спільна підпослідовність

TAM - Time Alignment Measurement - вимірювання з вимірюванням часу

STB - shape to bit vector - метод опису послідовностей за допомогою бітів

ASL - austarlian sign language - австралійська мова жестів

ЧП, S - часова послідовність

k - кількість точок даних у S

s_i - $i^{та}$ точка даних в S

Q - послідовність запиту

l - довжина Q

B - бітова строка, яка є міткою для корзини

b - кількість біт в B

ВСТУП

Протягом історії людство вдосконалювало свою промисловість, не лише покладаючись на сталий розвиток техніки, але і заново винаходячи її, оскільки нові ресурси створювали нові технічні засоби. Тому промисловість розвивалася якісними стрибками, які іноді були настільки інтегровані у певний проміжок часу і мали такий переважний вплив, що їх назвали "революціями".

Після повільного періоду протоіндустріалізації, перша революція охоплює період з кінця 18 століття до початку 19 століття. Вона була свідком появи механізації, процесу, який замінив сільське господарство промисловістю, як основою економічної структури суспільства. Масовий видобуток вугілля одночасно з винаходом парової машини створили новий тип енергії, який надав поштовху всім процесам завдяки розвитку залізниць та прискоренню економічного, людського та матеріального обміну.

Майже століття пізніше, наприкінці 19 століття, нові технологічні досягнення стали ініціатором появи нового джерела енергії: електроенергії, газу та нафти. Металургійна промисловість почала розвиватися і одночасно експоненціально розвивалися потреби в сталі. Способи зв'язку також були переосмислені з винаходом телеграфа і телефону, і те саме відбулося зі способами транспортування з появою автомобілів і літаків на початку 20 століття.

Майже через століття, у другій половині XX століття, відбулася третя промислова революція з появою нового типу енергії, потенціал якої перевершив всіх її попередників: ядерна енергія. Ця революція стала свідком піднесення електроніки - на базі транзисторів та мікропроцесорів - що дало поштовх зростанню телекомунікацій та комп'ютерів. Ця нова технологія призвела до виробництва мініатюрних виробів, які відкрили двері, насамперед до космічних досліджень та біотехнологій.

Четверта революція відбувається зараз. Її початок поклала поява Інтернету. Це перша промислова революція, коріння якої полягає в новому технологічному явищі - оцифровці - а не в появі нового типу енергії. Ця оцифровка дозволяє нам побудувати новий віртуальний світ, з якого ми зможемо керувати фізичним світом. Промисловість має на меті об'єднати всі виробничі засоби, щоб забезпечити їх взаємодію в режимі реального часу. Виробництва 4.0 роблять можливим комунікацію між різними гравцями та підключеними об'єктами на виробничій лінії завдяки таким технологіям, як Cloud, Big Data Analytics та Industrial Internet of Things [1].

Розумне виробництво - це технологічний підхід, який використовує підключені до Інтернету машини для контролю виробничого процесу. Мета РВ - виявити можливості для автоматизації операцій та використання аналітики даних для підвищення продуктивності виробництва [3]. Іншими словами, таке виробниче середовище, будучи спроектоване та побудоване, здатне самостійно та інтелектуально здійснювати вибір щодо технологій та алгоритмів виробництва, вирішувати виникаючі виробничі проблеми під час виробничого процесу та нарешті виходити з готовим продуктом. Для досягнення цієї мети потрібно виконати безліч проміжних завдань. Ці завдання включають здатність машин самостійно оцінювати якість своєї роботи та вміння самостійно коригувати технологічні процеси.

Машинне навчання - це один з підходів до створення штучного інтелекту (ШІ), що надає системам можливість автоматично навчатись та вдосконалюватись із досвіду без явного програмування [4]. Іншими словами, ШІ - це підхід, керований даними, який використовує дані для створення моделей ШІ. Модель ШІ може бути математичним поданням реального процесу [5]. У такому підході дані, що використовуються для навчання, мають дуже велике значення. Щоб створити якісні моделі ШІ, нам потрібно забезпечити алгоритми якісними даними.

Сучасні виробничі середовища створюють велику кількість даних, зібраних за допомогою вимірювань з датчиків, генерованих за допомогою контролерів та мережевих вузлів, доповнених метаданими, що надаються із зовнішніх областей. В безлічі промислових даних інженерам різних профілів потрібен інструмент для пошуку даних, вивчення знайдених даних, їх характеристик та виконання конкретних і специфічних операцій над цими промисловими даними. В даній роботі зроблено перший крок по створенню інформаційно-пошукову систему в великих неструктурованих даних з датчиків. Замовниками проекту, в рамках якого виконана дана робота, є співробітники Інституту технологій та машинобудування (IFW) університету ім. Лейбниця, м. Ганновер [6]. В результаті роботи Ця інформація потрібна їм для вирішення задач підвищення якості продукції, що виробляється, для вдосконалення технологій. Головна мета даної роботи, розробити методи роботи з часовими послідовностями промислових експериментів, що можуть бути покладені в основу інформаційно-пошукової платформу для промислових даних.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОБРОБКИ ЧАСОВИХ ПОСЛІДОВНОСТЕЙ

Пошукові системи стали важливою частиною при роботі з інформацією, починаючи з пір, коли для пришвидшення пошуку книг у бібліотеках ввели індекс присутніх видань і до теперішнього моменту з сучасними пошуковими системами. Насправді, з часом були розроблені нові втілення системи, але ідея підходу, що лежить в основі цих систем, залишилась незмінною. Сучасні пошукові системи дають змогу шукати інформацію різної природи. Прикладами таких систем є надзвичайно широко відомі пошукова система Google для текстових документів, а також для зображень, відеохостинг YouTube який має функцію пошуку орієнтовану на відео і Shazam для звукової інформації. Існують інші підвиди інформації. Час четвертої індустріальної революції характеризується створенням великої кількості інформації, яка була отримана при спостереженні за виробничими процесами. Такі дані представляють собою часові послідовності. У даній області не надто легко зробити огляд існуючих прикладів. Немає сумнівів що такі системи існують, але їх документація і детальний опис створює ризик розкриття таємниці інтелектуальної власності. Попри подібну специфіку, існують запатентовані розробки, до яких належать розробки описані в роботах [1-3].

З областю розумного виробництва пов’язана область інтернету речей. Дана область є прямою спадкоємицею розумного виробництва. Головна ідея, що перейшла до цієї область полягає у тому, що завдяки датчикам у побутових приборах ми можемо контролювати реальний світ використовуючи його віртуальну модель побудовану на даних з датчиків. Тому, розробки в даній області є адекватними при використанні їх у зворотньому напрямку. Однією з найбільш яскравих розробок є робота [11] у якій автори розробили повну архітектуру такої системи. Одним із найважливіших внесків роботи, на нашу думку, є опис складових і понять, потрібних для такої системи. До них належать

представлення даних, властивості, що описують часові послідовності, алгоритми індексації, структури даних та інше.

Без сумніву, базовим аспектом роботи з часовими послідовностями є ознаки, або як буде названо у даній роботі, властивості, якими ми описуємо дані. ЧП можна описати статистичними метриками, такими як як діапазон і середнє значення, дисперсія і середнє квадратичне відхилення та інші, геометричні, коли форма послідовності описується геометричними фігурами, наприклад мінімальний окреслюючий паралелепіпед використаний у роботах [5, 12]. Найбільш природним шляхом опису властивостей ЧП, є опис його амплітуди, у було зроблено у роботах [8-10]. Існують і більш специфічні підходи для опису патернів у послідовностях, наприклад у роботі [6] було наведено описи концепції подій і перетворення Фур'є з розділу математичного аналізу.

Традиційною структурою даних для пошуку є дерево і його різноманітні модифікації, тому що пошук відбувається за логарифмічний час, що є найкращим з можливих. Індекс може підтримувати просторові дані для опису геометричної форми, за допомогою R-дерев як описано у роботі [6]. Також у цій роботі було розроблено та запропоновано різновид дерева, названого PTrie. Це дерево використовується в одновимірному випадку, але в ньому зроблена оптимізація пам'яті. Воно нагадує звичайну структуру Trie, в якій кожен вузол зберігає більше ніж одне значення і дозволяє скоротити час пошуку і пам'ять процесу. Оскільки в даній роботі планується працювати з багатовимірними послідовностями, потрібно згадати роботу [5], у якій автори розробили багатовимірне дерево для багатовимірних часових послідовностей. У роботі [11] була створена модифікація RtGR-дерев які підтримує інформацію про час на рівні структури даних.

В області пошуку багатовимірних патернів у часових послідовностях значною мірою слід відзначити роботу [8], в якій адаптували відомий алгоритм пошуку K найближчих сусідів до багатовимірного випадку. Для цього

багатовимірні дані представляються як матриця, запит як матриця і пошук заключається в пошуку підматриці матриці даних найбільш що не відрізняється від запиту не більше ніж на задану порогову величину. Для цього проводиться аналіз запиту на наявність або відсутність в ньому запиту до тих чи інших вимірів. Відсутні виміри у запиті автоматично виключають відповідні виміри з даних, на яких відбувається пошук. Для пошуку патерну використовується пошук найдовшої підпослідовності, як у роботі [10], у якій алгоритм пошуку найдовшої спільної підстроки також поширили на багатовимірний випадок. Головний внесок їх роботи полягає у відмови від зміни структури індексу при додаванні нових вимірів, що не були присутні у перших доданих до нього об'єктах.

Міра подібності надзвичайно важлива при пошуку інформації у створеному індексі. Існують підходи, в яких одновимірні випадки розширені на більшу кількість вимірів. Історично із найперших методів порівняння ЧП є метод динамічного викривлення часу (DTW) докладно описаного в роботі [9]. Він був створений за прикладом відомого алгоритму Левенштайна для текстових послідовностей. За допомогою деякого вікна алгоритм створює матрицю, яка є таблицею відстаней на які треба змінити певні ділянки обох послідовностей, аби отримати іншу послідовність. Результатом є вектор, по якому можна судити, скільки змін треба внести до послідовності. Чим менша відстань редагування, тим більше схожі послідовності. Алгоритм DTW настільки вдалий, що для нових застосунків потребує лише деяких модифікацій. У роботі [6] одним із головних внесків роботи була нова міра подібності, що дозволяла користувачу вказувати ступінь нечіткості пошукового критерію, що дозволяло звужувати або розширювати перелік результатів. У роботі [7] автори створили модифікацію алгоритму Time Alignment Measurement for Time Series (TAM), що враховує викривлення не тільки у просторовому, а і часовому вимірі. У роботі [8] автори розробили модифікацію алгоритму DTW, що називається

DTW незмінний до перестановок, для пошуку багатовимірних патернів у багатовимірних ЧП.

Стратегія індексування об’єднує всі попередні розробки у єдиний робочий процес. Перш за все, характер створення індексу може підтримувати оновлення індексу під час роботи додатку або підтримувати створення всього індексу при наявності одразу всіх даних. Стратегії, що підтримують оновлення, у свою чергу можуть робити це у режимі реального часу або підтримувати інкременти час від часу під час перерви роботи додатка. В роботі [5] автори розробили метод, який дозволяє використовувати різні міри подібності при побудові індексу, не змінюючи при цьому структуру індексу, це дозволяє змінювати алгоритми порівняння не втрачаючи попередню роботу. В роботі [6] автор зробив огляд адаптації декількох методів індексування для тексту до задачі індексування часових послідовностей. В роботі [10] автори розробили стратегію індексування залежну від довжини індексованих ЧП. В роботі [11] одним із головних внесків даної роботи є розробка алгоритму індексування даних в режимі реального часу, що дозволяє оновлювати індекс новими даними. У роботі [12] головним внеском роботи є розробка структури індексу яка зберігає інформацію на декількох рівнях, для того, щоб при різній довжині запиту можна було повернути найбільш точний результат.

1.1 Постановка задачі дослідження

З огляду на існуючі досягнення в областях споріднених з пошуком у ЧП, для досягнення мети даної роботи було сформульовані наступні задачі:

- Розробити формат для зберігання даних для індексу;
- Розробити алгоритми витягнення властивостей;
- Розробити структуру даних для індексу багатовимірних даних;
- Розробити метрику подібності у багатовимірних даних.

2 РОЗРОБКА МЕТОДІВ ДЛЯ СТВОРЕННЯ СИСТЕМИ З НЕСТРУКТУРОВАНИМИ НАБОРАМИ ДАНИХ

Розглянуте в даній роботі рішення цілком являє собою так зване рішення кероване даними (data driven solution), будується навколо даних.

Бути керованим даними означає, що всі рішення та процеси диктуються даними, на відміну від того, щоб керуватися простою інтуїцією чи особистим досвідом. Іншими словами, рішення приймається з жорсткими емпіричними доказами, а не міркуваннями або почуттями. Термін використовується в багатьох сферах, але найчастіше в галузі техніки та бізнесу. Рішення включають прийняття рішень щодо конкретних структур даних та алгоритмів. Ось чому ми даємо точні та всебічні дані, надані IFW в першій підрозділі цієї глави. Спираючись на цей опис, ми визначаємо вимоги до структур даних та алгоритмів.

У цьому розділі обговорюється вибір формату даних для часових послідовностей, алгоритми вилучення властивостей часових послідовностей, алгоритми індексації часових послідовностей, структура даних індексації.

В даному розділі розглянуто питання вибору формату даних для часових послідовностей, алгоритмів вилучення властивостей часових послідовностей, алгоритмів індексування часових послідовностей, структури даних індексу.

2.1 Представлення даних

Визначення 2.1: Процес - фіксований у часі проміжок часу, протягом якого верстат виконував роботу над заготівкою під керуванням керуючого пристрою згідно запрограмованому завданню.

Процес являє собою сукупність даних з датчиків та метаданих. Процес у значній мірі залежить від конкретного верстата, а саме перелік його датчиків.

Загальна модель повинна бути достатньо гнучкою. Крім даних, що знімаються безпосередньо з датчиків, існує два види метаданих: метадані процесу і метадані керування.

До метаданих процесу належать: швидкість шпинделя, обрану для процесу, тип матеріалу, діаметр фрези, ширина та глибина розрізу.

До метаданих керування належать: координати положення відносно основних координат, швидкість переміщення відносно координат, прискорення переміщення відносно координат, струм живлення.

Підсумовуючи всі описи, сформулюємо вимоги до представлення процесу. Воно має:

- підтримувати довільну кількість полів метаданих, дозволяючи додавати або видаляти довільні пари <назва,значення>;

- підтримувати довільну кількість часових послідовностей в своєму складі;

- підтримувати довільну щільність часових послідовностей.

Для підтримки балансу ефективності використання пам'яті та не змушуючи користувача заповнювати надто багато полів для конфігурації за замовчуванням модель процесу матиме наступний вигляд:

$$\{\text{<process metadata>}, \text{<control metadata>}, [\text{<sensor model>:<observation model>}]\}, \quad (2.1)$$

де process metadata - метадані процесу;
control metadata - метадані керування;
sensor model - представлення датчика;
observation model - модель даних.

Представлення метаданих процесу:

$$\{PID; [\langle p_1\rangle:\langle pv_1\rangle, \langle p_2\rangle:\langle pv_2\rangle,\ldots, \langle p_n\rangle:\langle pv_n\rangle]\}, \quad (2.2)$$

де PID - ідентифікатор процесу;

$\langle p_n \rangle : \langle pv_n \rangle$ - пари ключ:значення;

p_n - властивість процесу n;

pv_n - значення властивості n.

Модель метаданих керування:

$$\{CID; [\langle c_1 \rangle : \langle cv_1 \rangle, \langle c_2 \rangle : \langle cv_2 \rangle, \ldots, \langle c_n \rangle : \langle cv_n \rangle]\}, \quad (2.3)$$

де CID - ідентифікатор процесу:

$\langle c_n \rangle : \langle cv_n \rangle$ - пари ключ:значення;

c_n - властивість контролю n;

cv_n - значення властивості n.

Під час роботи, на верстаті працюють датчики, що одночасно збирають показники процесу. Датчики вимірюють прискорення переміщення, механічний тиск і звуки. Однією з найважливіших характеристик датчика є частота вибірки. Вона може значно відрізнятися для двох різних датчиків. Крім цього датчика властиво мати ідентифікатор інструмента, на якому його використано.

Визначення 2.2: Частота вибірки - це кількість замірів, отриманих за одну секунду.

Дана величина має великий вплив на кількість даних, що були згенеровані. За проміжок часу протягом хвилини, кількість зареєстрованих значень може сягати сотень тисяч.

Треба також враховувати, що нотація назв та вміст можуть бути визначені довільно тим, хто створює лабораторний звіт. В даній роботі ми спираємося лише на дані, надані нам конкретною персоною з IFW.

Дані експортовані у csv форматі, файл якого називається "N5750AE8AP05_6Stahl_langD8".

Нотація його назви містить метадані про процес, де:

- N - швидкість шпинделя (об/хв);

- AE - ширина розрізу (мм);

- AP - глибина розрізу (мм);

- D - діаметр фрези (мм).

В даному файлі зняті дані з наступних датчиків:

- <axis>_Machine_CS - позиція по осі axis

- <axis>_Machine_V - швидкість по осі axis

- <axis>_Machine_a - прискорення по осі axis

- <axis>_Machine_I - струм живлення по осі axis

Слідом за метаданими розташовані безпосередньо часові послідовності. Таблиця 2.1 містить узагальнену інформацію про характеристики часових послідовностей. В таблиці міститься 9583 строки з частотою вибірки 1 раз/мс.

Таблиця 2.1 - загальні відомості про часові послідовності

Тип датчика	Діапазон	Тип даних
[X,Y,Z,C,B]_Machine_CS	[79.3099:79311], [120.3905:224.9453], [-171.0505:-166.6795], [359.9624:359.9649], [0:0.0001]	REAL64
[X,Y,Z,C,B]_V_Machine_CS	[-0.6152:0.4052], [-0.1123:12.0166], [-1.8115:0.8496], [-0.5686:0.987], [-0.0799:0.0799]	REAL64
n_Spindle	[5246,086:5255,928]	REAL64

Продовження таблиці 2.1

[X,Y,Z,C,B]_a_Machine_CS	[-383,3007:310,0585], [-471,1914:285,6445], [-1540,5273:1547,8515], [-560,5816:523,0307], [-66,6646:66,6646]	REAL64
[X,Y,Z,C,B]_I	[-2,498:-0,239], [-3,325:-0,082], [-1,176:0,801], [-0,188:3,069], [-0,189:0,451]	REAL64
VARIABLES.SIGNAL_FORCE [1,2,3]	[-0,004638:-0,004638] [-0,002319:-0,002319] [0,0009422:0,0009422]	REAL64
ChatterScore	[11,86369:670,01879]	REAL64
IsChatter	[0,1]	BIT
AE	[0:0]	REAL64
AP	[0:0]	REAL64
LastChatterScore	[11,8636932:670,0188]	REAL32
A_OVERSAMPLING_INPUT_[1-6]	[-341:435] [-300:271] [-4468:4130] [-579:534] [-452:341] [-8:7]	REAL16
IsChatterSVM	[-1,1]	REAL64

Представлення датчика у даному дослідженні дуже подібна моделі процесу. Різниця полягає у відсутності структурних одиниць залежних від неї.

$$\{SID;\ [\langle s_1\rangle{:}\langle sv_1\rangle,\ \langle s_2\rangle{:}\langle sv_2\rangle,\ldots,\ \langle s_n\rangle{:}\langle sv_n\rangle]\}, \tag{2.4}$$

де $\langle s_n \rangle : \langle sv_n \rangle$ - властивості датчика,

s_i - це ім'я властивості i;

sv_i - це значення величини p_i.

Представлення часових послідовностей:

$$\{TID, [<ts_i:tsv_i>]\}, \quad (2.5)$$

де TID - ідентифікатор часової послідовності

ts_i - часова позначка у відносному форматі в момент i;

tsv_i - значення датчика у момент i.

2.2 Загальний опис роботи системи

За допомогою запита користувач формулює опис бажаної інформації. Спершу запит аналізується та трансформується для пошуку в індексі. Індекс представляє собою структуру даних, призначену для прискорення встановлення місцезнаходження ділянок часових послідовностей що відповідають запиту. Це досягається тим, що дані в індексі впорядковані за деяким принципом, який дозволяє відкидати наперед нерелевантні варіанти від час порівняння з запитом. Наступним кроком відбувається пошук у індексі, в основі якого лежить порівняння запиту з записами індексу. Порівняння відбувається відносно ознак часових послідовностей, названих у даній роботі властивостями. Пошук відбувається із деякою мірою нечіткості. Ця вимога обумовлена тим, що, по-перше, точне співпадіння часових послідовностей не представляє жодної цінності у контексті поставленої мети і, по-друге, ймовірність існування майже однакових послідовностей надто мала, і надто жорсткі критерії релевантності знизять якість результатів.

Спираючись на наведений опис, сформулюємо питання, на які буде дана відповідь в даному розділі:

1. Які властивості потрібно вилучити з часових послідовностей;
2. Який алгоритм використовується для порівняння послідовностей, як регулюється нечіткість;
3. Яку структуру даних індексу обрати;
4. У якій формі потрібно задати запит;
5. Яким чином можна знайти аномалії та як їх класифікувати.

Подальша структура розділу підпорядкована переліку питань.

2.3 Витягнення властивостей

В даній роботі попри багатовимірний характер даних, витягнення властивостей з часових послідовностей здійснюється з одного виміру, а потім результати об’єднуються при подальшій обробці. Тому у даному підрозділі йде мова про методи вилучення властивостей розроблених для одновимірних послідовностей.

Як буде обговорено далі, у даній роботі увага була зосереджена на запитах за формою. Запити за формою - це запити що описують форму бажаного результату, наприклад "Знайти всі послідовності збережені в системі, які містять форму аналогічну заданій в запиті формі". Цей запит передбачає, що користувач вводить шукану форму у аналітичний або графічний спосіб. Це не єдина форма запитів, яку можна задати на часових послідовностях. Інші типи запитів включають запити на час, умовно "У який час часовий ряд мав значення ‘5’? Або навпаки - на значення, наприклад "Яке значення мали часові ряди в момент ‘7’?". Тип запитів, які ми збираємося здійснити в системі, є важливим. Оскільки під час вилучення властивостей ми вибираємо лише певну інформацію про часовий ряд, те, як ми обираємо цю інформацію, матиме

великий вплив на інші частини системи.

Для прикладу процесу вилучення властивостей уявіть, що часовий ряд – це потік автомобілів на шосе, і видобуток цієї властивості виконує людина, що стоїть поруч із шосе. Особа, відповідальна за вилучення властивостей, виділить властивості проїжджаючих машин, які є важливими для нього. Однією з особливостей, яку він може відзначити, є колір автомобілів, що проходять. Потім ми зможемо пізніше запитати його, наприклад, чи були у потоці сині машини. Але якщо це єдина особливість, яку він витягує з потоку автомобілів, ми не зможемо запитати у нього, чи були в вантажному потоці вантажівки.

Те саме стосується і часових послідовностей. Ми повинні вирішити, який аспект часової послідовності ми маємо намір запитувати (шукати), щоб визначитися з процесом вилучення властивостей. Логічне питання тоді, чому ми не зберігаємо всі аспекти часового ряду. Ще одна причина для вилучення властивостей - це видалення «неважливої» інформації, щоб зменшити кількість інформації, яку ми маємо індексувати. Зазвичай опис сигналу після вилучення властивостей значно менший, ніж сигнал.

Під час роботи з часовими послідовностями, як правило, індексувати абсолютні значення та відстані в сигналі є поганою ідеєю. У багатьох випадках користувач зацікавлений лише у пошуку послідовностей, схожих або таких же, як послідовність у запиті. Якщо системі надано запит, наприклад, у вигляді послідовності [1.1, 1.4, 1.0, 0.8,], користувач, ймовірно, хоче знайти послідовності [1.1, 1.4, 1.0, 0.9] і [1.2, 1.4, 0.9, 0.8], якщо вони є в індексованому матеріалі, навіть якщо вони не точно відповідають послідовності в запиті. Це схоже на проблему, виявлену при індексації тексту. Якщо в запиті включено дієслово, зазвичай це не важливо, чи є він у теперішньому чи минулому часі, важлива річ, описана дієсловом. Те саме стосується часових послідовностей. Запит призначений зазвичай не для знаходження точної послідовності значень, а для всіх послідовностей, які певним чином схожі на послідовність запиту. Є ще одне важливе питання, коли ми говоримо про часові послідовності. Часова

послідовність будується шляхом вибірки безперервного сигналу. Якщо ми зберігаємо різні сигнали з різними частотами вибірки, отриманий часові ряд буде різним, попри те, що вони можуть мати однакову суть, наприклад координати положення. І це одна з причин того, що важливо, щоб механізм запитів знаходив обидві часові послідовності, оскільки користувач не має контролю над процесом вибірки.

Вища математика дає нам декілька інструментів для опису поведінки та форми сигналу, які допомагають витягувати функції із сигналу. Спочатку наведемо короткий огляд методів, а далі більш докладний.

Перший метод, що використовується для передачі поведінки сигналу - це обчислення дискретного перетворення Фур'є часової послідовності (DFT). Після цього сигнал розглядається не як послідовність значень у часовій області, а як сукупність частот. Якщо ми хочемо отримати загальну форму сигналу та уникнути шуму і невеликих змін значень, ми можемо позбутися всіх високочастотних компонентів DFT та просто зберегти перші кілька частотних компонентів. Ці компоненти можуть дати нам досить гарне наближення форми сигналу. Якщо наближення потрібно вдосконалити, потрібно зберегти більше частотних компонентів.

Другий метод передачі поведінки сигналу полягає в наближенні коротших сегментів сигналу до поліномів. Якщо нас просто цікавить, збільшується чи зменшується загальна тенденція сигналу, ми можемо наблизити сигнал до поліному першого ступеня. Це дуже грубе наближення, але його можна легко уточнити, використовуючи поліноми 2-го чи 3-го ступеня. Проблема такого підходу полягає в тому, що важко сегментувати сигнал, щоб розриви, задані розбиттям сигналу, не заважали нашій здатності запитувати послідовність поліномів.

2.3.1 Вилучення властивостей за допомогою похідних часу

Для отримання властивостей з вхідного сигналу ми використовуємо процес, який неформально можна описати так: потрібно побудувати похідну часу сигналу та оцінити його кількісно. Спочатку ми скануємо сигнал. Для кожної точки часової послідовності, ми обчислюємо різницю між значенням у точці та значенням у наступній точці. Потім це значення відображається на символ із алфавіту. Отримана строка дуже інтуїтивно описує форму вхідного сигналу.

Ми використовуємо алфавіт SDA для опису сигналу. Розмір SDA важливий, оскільки він визначає деталізацію нечіткості в системі. Коли ми використовуємо позначення SDA, ми маємо на увазі SDA взагалі, але якщо ми маємо на увазі конкретну версію SDA, наприклад, з п’ятьма символами, то ми використовуємо нотацію SDA(5). Оскільки ми відображаємо реальне значення, похідну часу сигналу в кожній точці, на символ з алфавіту (таблиця 2.2), ми вводимо певну нечіткість в систему. Часова послідовність, описана як строка символів, буде мати обвідну область, і всі часові послідовності, що знаходяться в межах області, будуть відповідати одній строці символів. Обвідна зображена на рисунку 2.1 описується не як просте значення відступу рівного Δ навколо сигналу, а як дещо інша область, оскільки це похідна часу послідовності, а не часова послідовність, яка відображається на алфавіт. Якщо часова послідовність укладається в область огину іншого часової послідовності, вони вважаються подібними. Обвідну для трьох різних послідовностей можна побачити на рисунках А.1, А.2 та А.3 в додатку А.

Таблиця 2.2 - приклад SDA(5)

Символ	Значення	Визначення
a	Швидко зростаючий перехід	$\frac{d}{dt} > 5$
b	Повільно зростаючий перехід	$5 \geq \frac{d}{dt} > 2$

Продовження таблиці 2.2

c	Сталий перехід	$2 \geq \frac{d}{dt} \geq -2$
d	Повільно спадаючий перехід	$-2 > \frac{d}{dt} \geq 5$
e	Швидко спадаючий перехід	$\frac{d}{dt} < 5$

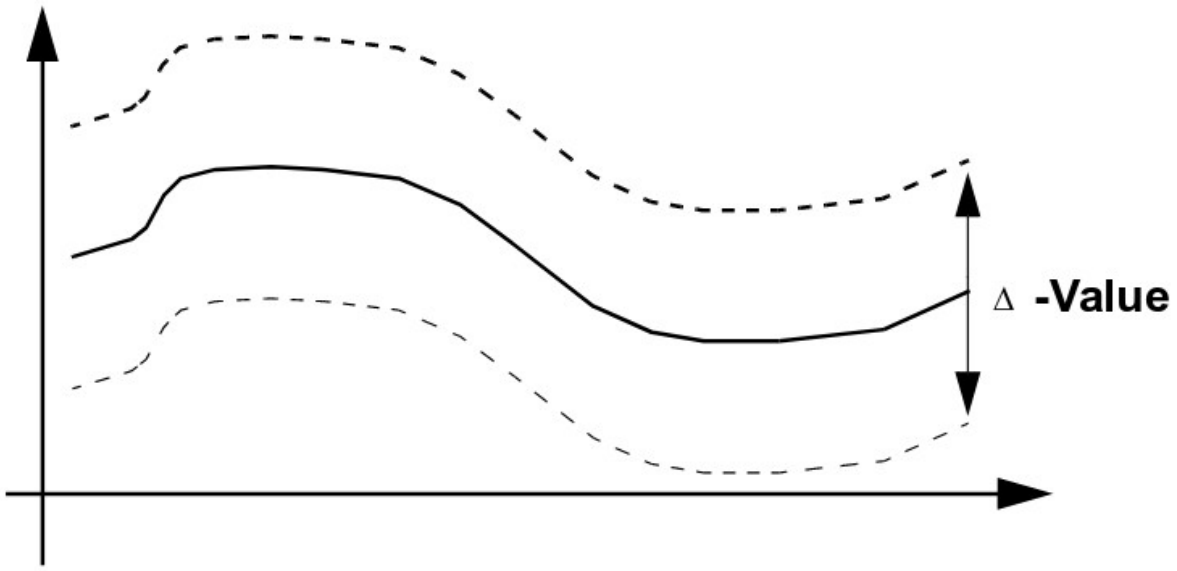


Рисунок 2.1 - Опис зміни обвідної області

У цьому відображенні вводиться певний рівень нечіткості. Якщо текстові строки використовуються як поняття подібності, можна сказати, що дві послідовності схожі, якщо вони відображаються в одному текстовому рядку. Якщо ми розглянемо послідовність на рисунку 2.2, [0, 1, 4, 10, 14, 15, 15, 11, 10, 20, 14, 8, 2, 0], вона буде відображатись у рядку «cbabccdcaeeec» (використовуючи SDA(5) з таблиці 2.2). Але ми можемо бачити, що послідовність, показана на рисунку 2.3, [-10, -11, -6, 1, 6, 4, 2, -1, -3, 15, 9, 3, -3, -1] також буде відображатись у тому ж рядку. Ці дві послідовності, таким чином, вважаються системою подібними. Таким чином деталізація нечіткості контролюється розміром алфавіту.

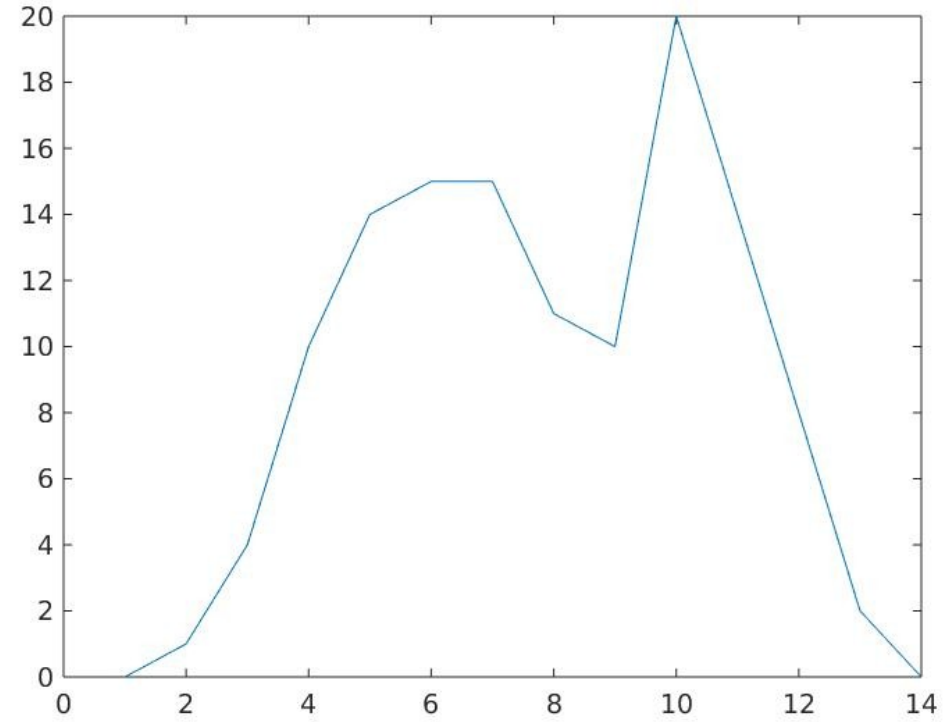


Рисунок 2.2 - Приклад строки 1

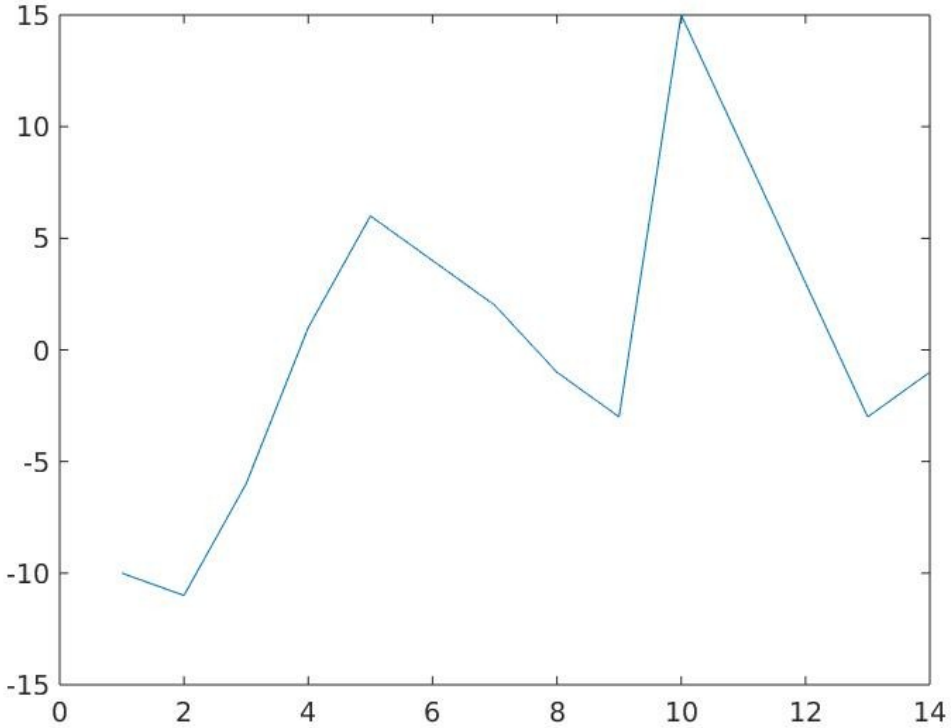


Рисунок 2.3 - Приклад строки 2

Ця концепція подібності дуже приваблива, оскільки є дуже простою, інтуїтивно зрозумілою і близькою до того, як люди думають про подібність. Вона також має бажану властивість бути нечутливою до амплітуди.

Але такий підхід має недоліки. Якщо це єдине використовуване поняття подібності, а розмір алфавіту невеликий, кожен запит призведе до великої кількості колізій. Кожне співпадіння буде відображено на деякий текстовий рядок, але схожість між сигналами може бути занадто розпливчастою, щоб людина сприйняла подібність. Для усунення цього недоліку, розмір алфавіту можна збільшити. Використовуючи таку ж концепцію подібності, як і раніше, кількість збігів, отриманих за допомогою запиту, тепер буде меншою, оскільки зараз є менше сегментів сигналів, які відображаються в одну строку.

Невелика модифікація методу вилучення властивостей за допомогою похідних часу, полягає у використанні кривих замість похідної часу. З точки зору нашої структури індексування, цей метод є ідентичним методу вилучення властивостей за допомогою похідної часу, описаної вище, оскільки є результатом процесу вилучення властивостей є послідовність символів, яка може бути оброблена так само, як якщо б вона була створена за допомогою похідної часу.

2.3.2 Вилучення властивостей за допомогою DFT

Це широко використовуваний метод вилучення особливостей. Якщо задача полягає в індексації колекцій коротших послідовностей, перетворення, наприклад, дискретне перетворення Фур'є, застосовується до кожної послідовності, і обираються два або чотири компоненти для представлення всієї послідовності. Ці компоненти формують наш вектор властивостей.

Оскільки ми витягуємо дуже мало інформації з кожної послідовності, стиснення буде дуже хорошим. Цей метод дозволить нам шукати набір

послідовностей для даної послідовності, але якщо послідовності довші, може бути цікавим знайти підпослідовності в довгій послідовності, рисунок 2.4.

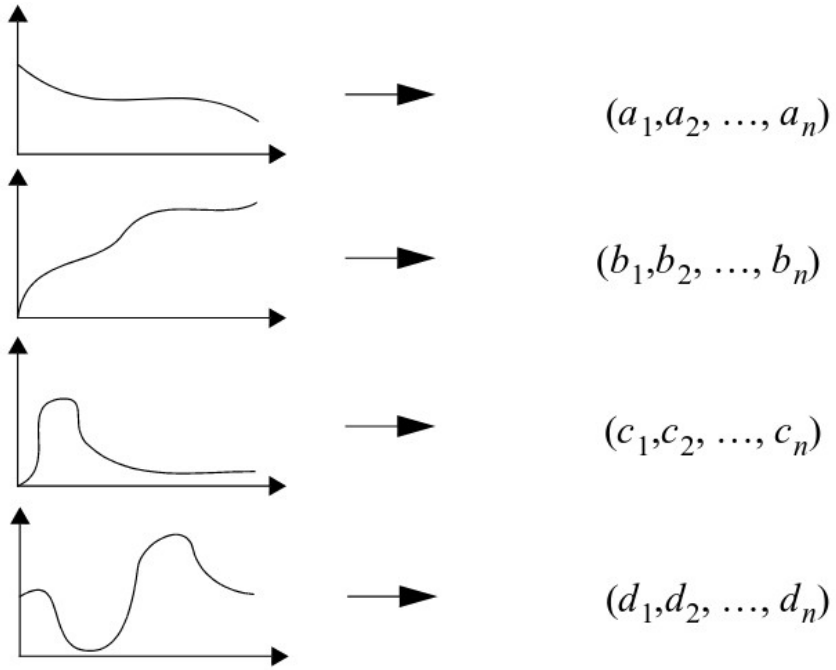


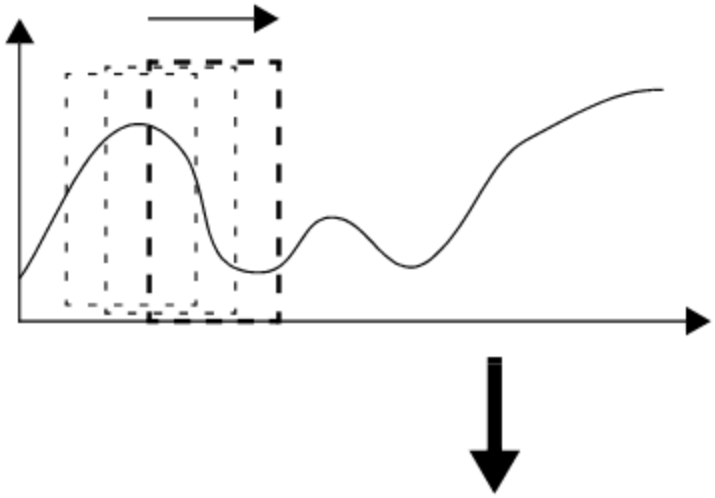
Рисунок 2.4 - Витягнення властивостей за допомогою DFT, приклад 1

Якщо нам потрібно знайти послідовності в більшій послідовності, ми повинні отримати більше інформації з кожної послідовності. Для цього вводимо ковзаюче вікно шириною n. На початку вікно розміщується над першими n значеннями часового ряду:

1. Перетворити n значень у вікні за допомогою DFT для отримання вектора властивостей;
2. Зберегти x компонентів з DFT, як правило, два або три;
3. Перемістити вікно на крок вперед у часовій послідовності;
4. Повторити крок 1.

Вся часова послідовність після цього процесу булі відображена у послідовність векторів властивостей, рисунок 2.5. Тоді можна буде знайти будь-яку послідовність довжини n у послідовності. Неможливо знайти більш короткі послідовності, але, розділивши запит довжиною m, за умови що m > n, на m - n

+ 1 запитів, кожен довжиною n, можна знайти всі послідовності довші за n.



$$(a_{11}, \ldots, a_{1n}), (a_{21}, \ldots, a_{2n}), \ldots, (a_{k1}, \ldots, a_{kn})$$

Рисунок 2.5 - Витягнення властивостей за допомогою DFT, приклад 2

2.3.3 Витягнення властивостей за допомогою подій

Якщо ми представляємо послідовність як ряд цікавих подій з нецікавими послідовностями між ними, і якщо припустити, що кількість подій є низькою порівняно з тривалістю послідовності, іншим підходом можна спробувати знайти ці цікаві події та витягнути їх разом з їх координатами у вихідній послідовності.

Якщо часова послідовність складається з відносно небагатьох подій, які повторюються у великій кількості разів, слід мати можливість використовувати ці події як слова, а потім використовувати метод індексації тексту без змін для індексації часових послідовностей.

Здійснення вилучення подій може, звичайно, поєднуватися з декількома способами вилучення властивостей, описаними вище. Спочатку послідовність конвертується у текстовий рядок, але цілком можливо спершу знайти всі події, а потім обчислити DFT для кожної знайденої події.

Коли ми говоримо про пошук подій у послідовності, важливо спершу вирішити, що таке подія. Неформально це досить просто, подія - це колись щось відбувається в послідовності. Формально подія описується трьома основними параметрами, які мають вплив на пошук подій: нечіткість, упередження (look-a-head) та мінімальний розмір події. Розглянемо ці параметри докладніше.

Нечіткість задає порогове значення, перш ніж нова подія буде розпізнана. З "нечітким співпадінням" ми розпізнаємо подію, як тільки знаходимо варіацію послідовності з заданою точністю, тобто два сусідніх елемента в послідовності символів відрізняються не більше ніж на вказану величину нечіткості. З більшою величиною нечіткості ми повинні відбутися більші зміни, перш ніж нова подія буде розпізнана. Наприклад якщо у нас є послідовність [aaabccdfaeee...dc] і використовуємо нечітке співпадіння, перший початок події, який ми знайдемо, породжується "abc". Якщо ми збільшимо параметр нечіткості до 2, початком знайденої події буде "cdf", оскільки це перший відрізок, знайдений там, де максимальна відстань між символами в послідовності більша за 1.

Упередження описує, на скільки кроків попереду функція “дивиться”, щоб визначити, починається чи закінчується подія. Наприклад припустимо, що у нас є послідовність [aaabccdfaeee...dc], і, скажімо, що ми допускаємо нечіткість рівну 1 без наявного початку події у вікні. Якщо ми встановимо упередження на 1 і почнемо шукати подію, перша подія, яку ми знайдемо - це "dfae...", оскільки "aa", "ab", "bc", "cc" та " cd "відрізняються лише на один елемент. Якщо ми встановимо упередження на 3, першою подією, яку ми знайдемо, є "abc..." Оскільки "aab" відрізняється лише в одному елементі, вона буде пропущена, але на наступному кроці, "abc", відрізняється на 2 елементи.

Як тільки ми знайшли початок події, ми видаляємо перші символи, які не є частиною події. У наведеному вище прикладі, коли упередження встановлене з пороговим значенням 2, якщо "aab" запускає початок нової події, нова подія

буде збережена починаючи з просто "ab". Потім ми продовжуємо, поки не знайдемо нову послідовність, яка не містить змін. У наведеному вище прикладі (look-a-head встановлений рівним 2 з точним співпадінням), якщо "aab" є початком нової події, подія буде тривати, поки не буде знайдено "eee", а повна подія рівна "abccdfae".

Ще один важливий параметр керування - мінімальний розмір події. Оскільки ми можемо отримати величезну кількість дуже малих подій, ми можемо задати нижню межу щодо тривалості подій, які ми приймаємо як цікаві події. Це простий механізм фільтрації. Подія розпізнається, і якщо розмір виявленої події менший за попередньо встановлене значення, воно відкидається і пошук подій триває.

2.4 Метрики подібності

2.4.1 Алгоритм DTW

Алгоритм DTW завоював свою популярність, будучи надзвичайно ефективна як міра подібності часових послідовностей, яка мінімізує наслідки зсуву та спотворень у часі, дозволяючи "еластичне" перетворення часових рядів з метою виявлення подібних фігур на різних фазах, рисунок 2.6. Дано часовий ряд $X = (x_1; x_2; ... x_N)$, $N \in N$ і $Y = (y_1; y_2; ... y_M)$, $M \in N$ представлений послідовностями значень (або кривими, представленими послідовностями вершин) DTW дає оптимальне рішення за час O(MN), яке можна у подальшому вдосконалювати. Єдине обмеження до послідовностей даних, полягає в тому, що вони повинні бути відібрані в рівновіддалені моменти часу.

Якщо послідовності беруть значення з певного простору функцій Ф, тоді для порівняння двох різних послідовностей $X, Y \in Ф$ потрібно використовувати міру відстані, визначену як функція:

$$d: \Phi \times \Phi \rightarrow R >= 0. \quad (2.6)$$

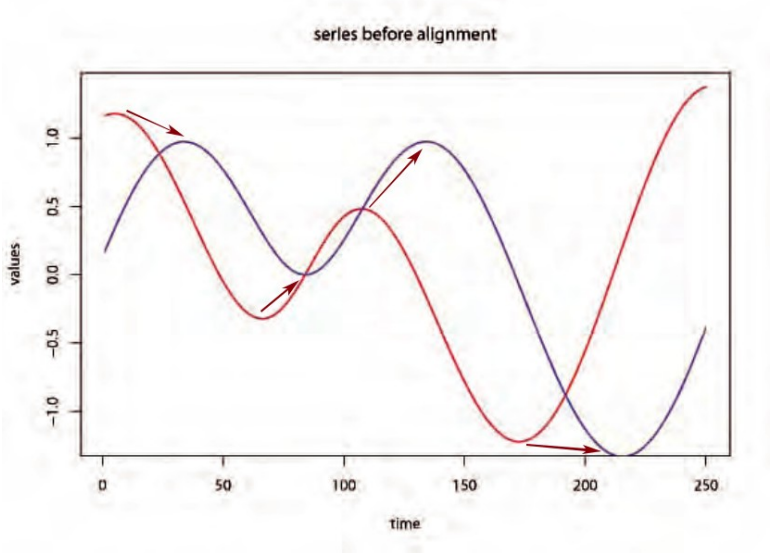


Рисунок 2.6 - Необроблена часова послідовність, стрілки показують бажані точки вирівнювання

Інтуїтивно d має невелике значення, коли послідовності схожі, і велике значення, якщо вони різні. Оскільки в основі DTW лежить алгоритм динамічного програмування, прийнято називати цю функцію відстані функцією вартості, а завдання оптимального вирівнювання послідовностей стає завданням упорядкування всіх точок послідовності шляхом мінімізації функції вартості (або відстані).

Алгоритм починається з побудови матриці відстані $C \in R^{N*M}$, що представляє всі попарні відстані між X і Y. Ця матриця відстані називається матрицею локальних витрат для вирівнювання двох послідовностей X і Y:

$$C_l \in R^{N*M}: c_{i,j} = \| x_i - y_i \|, i \in [1:N], j \in [1:M]. \quad (2.7)$$

Після того, як матриця локальних вартостей побудована, алгоритм знаходить шлях вирівнювання, який проходить через райони низьких вартостей - "долини" на матриці вартостей, рисунок 2.7. Цей шлях вирівнювання (або

деформації, або функції деформації) визначає відповідність елемента $x_i \in X$ до $y_j \in Y$ дотримуючись граничної умови, яка присвоювала перші і останні елементи X і Y один одному, рисунок 2.8.

Формально кажучи, шлях вирівнювання, побудований DTW, є послідовністю точок $p = (p_1; p_2; ...; p_K)$ з $p_l = (p_i; p_j) \in [1: N] * [1: M]$ для $l \in [1: K]$ який повинен відповідати наступним критеріям:

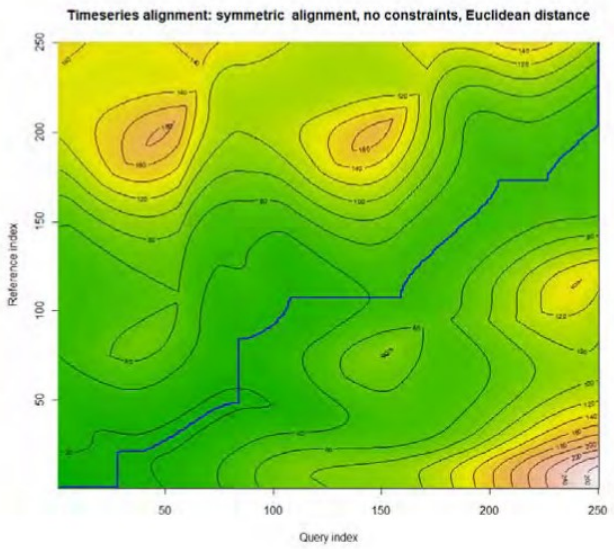


Рисунок 2.7 - Вирівнювання часових рядів, теплова карта матриці

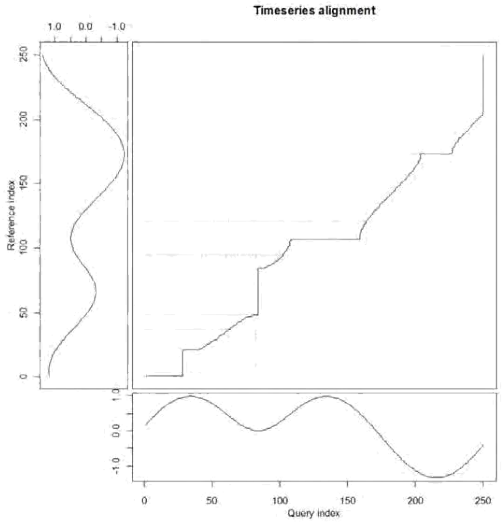


Рисунок 2.8 - Оптимальний шлях викривлення, вирівнюючи часовий ряд із рисунка 2.7

- гранична вимога: $p_1 = (1;1)$ і $p_K = (N;M)$. Початковою і кінцевою точками деформації шляху повинні бути перша і остання точки вирівняних послідовностей;

- вимога монотонності: $n_1 \leq n_2 \leq ... \leq n_K$ і $m_1 \leq m_2 \leq ... \leq m_к$. Ця умова зберігає часовий упорядкування балів;

- вимога розміру кроку: цей критерій обмежує деформацію шляху від тривалих стрибків (зміщення у часі) при вирівнюванні послідовностей. Поки що буде використано основну умову розміру кроку, сформульовану як $p_{l+1} - p_l \in \{(1; 1); (1; 0); (0; 1)\}$.

Функція вартості, пов'язана з викривленням шляху, обчислена відносно локальної матриці витрат (яка представляє всі попарні відстані), буде:

$$c_p(X,Y) = \sum_{l=1}^{L} c\left(x_{n_l}, x_{m_l}\right). \tag{2.8}$$

Шлях деформації, який має мінімальні витрати, пов'язані з вирівнюванням, називаються оптимальним деформацією шляху. Ми будемо називати цей шлях P.

Дотримуючись визначення оптимального шляху викривлення для того, щоб знайти його, нам потрібно перевірити всі можливі деформації шляху між X і Y, що може бути обчислювально складною задачею через експоненціальне зростання кількості оптимальних шляхів у міру зростання довжин X і Y лінійно. Щоб подолати цей виклик, DTW використовує підхід, заснований на динамічному програмуванні, зі складністю лише O(MN).

Динамічна частина алгоритму DTW використовує функцію відстані

$$DTW(X,Y) = c_p*(X,Y) = min\, c_p(X,Y),\ p \in P^{N*M}, \tag{2.9}$$

де P^{N*M} - сукупність усіх можливих шляхів деформації та будує матриця накопиченої вартості або загальна матриця витрат D, яка визначається наступним чином:

1. Перший рядок: $D(1,j)=\sum_{k=1}^{j} c(x_1,y_k), j\in[1,M]$.

2. Перший стовпець: $D(i,1)=\sum_{k=1}^{i} c(x_k,y_1), i\in[1,N]$.

3. Усі інші елементи:

$D(i,j)=min(D(i-1,j-1),D(i-1,j),D(i,j-1)+c(x_i,y_j), i\in[1:N], j\in[1:M])$.

Часова вартість побудови цієї матриці дорівнює O(NM), що дорівнює вартості наступного алгоритму 2.1, рисунок 2.9, де X і Y - вхідні часові послідовності, а C - матриця локальних витрат, що представляє всі попарні відстані між X і Y:

Algorithm 2.1: Accumulated Cost Matrix (X, Y, C)

Input : X and Y — time series, C — local cost matrix
Output: *DTW*

```
1  n ← |X|
2  m ← |Y|
3  dtw[] ← new [n * m]
4  dtw(0, 0) ← 0
5  for i = 1; i ≤ n; i++ do
6  |  dtw(i, 1) ← dtw(i-1, 1) + c(i, 1)
7  end
8  for j = 1; i ≤ m; j++ do
9  |  dtw(1, j) ← dtw(1, j-1) + c(1, j)
10 end
11 for i = 1; i ≤ n; j++ do
12 |  for j = 1; j ≤ m; j++ do
13 |  |  dtw(i, j) ← c(i, j) + min(dtw(i-1, j), dtw(i, j-1), dtw(i-1, j-1))
14 |  end
15 end
16 return dtw
```

Рисунок 2.9 - Лістинг алгоритму матриця акумульованої вартості

Як тільки побудована матриця накопиченої вартості побудована деформація шлях може бути знайдений простим зворотним відстеженням від точки p_{end} = (M; N) до p_{start} = (1; 1) дотримуючись жадібної стратегії, описаної в алгоритмі 2.2, рисунок 2.10.

Algorithm 2.2: Optimal Warping Path

```
1 path ← new []
2 i = rows(dtw)
3 j = columns(dtw)
4 while (i > 1) & (j > 1) do
      if i == 1 then
        j = j - 1
      else
        if j == 1 then
          i = i - 1
        else
          if dtw(i-1, j) == min(dtw(i-1, j), dtw(i, j-1), dtw(i-1, j-1)) then
            i = i - 1
          else
            if dtw(i, j-1) == min(dtw(i-1, j), dtw(i, j-1), dtw(i-1, j-1)) then
              j = j - 1
            else
              i = i - 1
              j = j - 1
            end if
          end if
        end if
      end if
5 end
6 path.add((i; j))
7 return path
```

Рисунок 2.10 - Лістинг алгоритму оптимальний шлях викривлення

2.4.2 Алгоритм LCSS

Оригінальний алгоритм LCSS стосується одновимірних послідовностей; тому його потрібно поширити на багатовимірний випадок. Важливо те, що парадигма LCSS здатна працювати з дискретними значеннями; однак у нашому випадку ми хочемо також дозволити співпадіння, коли значення знаходяться в певному діапазоні в просторі та часі. Зауважимо, що це також допомагає уникнути віддалених та вироджених співпадінь.

Припустимо, що вимірювання проводяться через фіксований та дискретний проміжок часу, що справедливо для нашого випадку.

Нехай A і B - двовимірні часові послідовності з розмірами n і m відповідно, де A = (($a_{x,1}$, $a_{y,1}$), ... , ($a_{x,n}$, $a_{y,n}$)) і B = (($b_{x,1}$, $b_{y,1}$), ... , ($b_{x,m}$, $b_{y,m}$)). Де функція визначена як Head (A) = (($a_{x,1}$, $a_{y,1}$),.., ($A_{x,\,n-1}$, $a_{y,\,n-1}$)).

Визначення 2.3 Враховуючи цілі числа δ і ε, визначити $LCSS_{\delta,\varepsilon}$(A, B) наступним чином:

$$LCSS_{\delta,\epsilon}(A,B)=\begin{Bmatrix} 0,if A\vee Bisempty \\ 1+LCSS_{\delta,\epsilon}(Head(A),Head(B)), \\ if\,|a_{x,n}-b_{x,m}|<\epsilon\wedge|a_{y,n}-b_{y,m}|<\epsilon\wedge|n-m|\le\delta \\ max\left(LCSS_{\delta,\epsilon}(Head(A),B),LCSS_{\delta,\epsilon}(A,Head(B))\right)otherwise \end{Bmatrix}. \qquad (2.10)$$

Константа δ контролює ступінь співпадінь у часовій області, а константа ε - поріг співпадінь у просторі. Цю модель LCSS можна обчислити, використовуючи алгоритм динамічного програмування. Його часова складність має порядок O(δ(n+m)), якщо дозволити лише відповідне вікно δ у часі. Екземпляр виконання динамічного програмування між двома траєкторіями зображений на рисунку 2.11. Заповнення масиву починається з нижнього лівого кута, а обчислене значення у верхньому правому куті задає остаточну схожість між двома траєкторіями.

Значення масиву в положенні [i, j] збільшується на максимум з-поміж суміжних значень масиву ([i-1, j], [i, j-1], [i-1, j-1]), лише якщо значення траєкторії A в час i та траєкторії B в час j не відрізняються більше, ніж на ε у всіх вимірах (для цього прикладу два виміри). У тому випадку, коли часове вікно δ також накладено, тоді потрібно обчислювати лише елементи масиву, які знаходяться до δ позицій від діагоналі. У прикладі з рисунку 2.11 сіра область позначає елементи масиву, які будуть обчислені під відповідним вікном δ = 5.

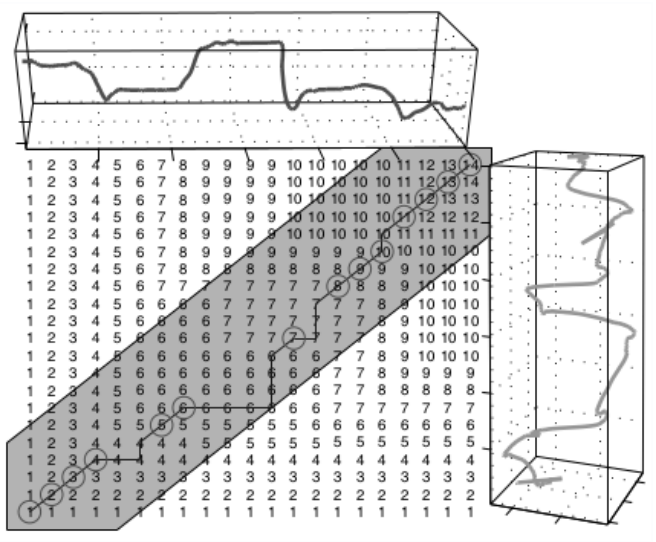


Рисунок 2.11 - Обчислення динамічного алгоритму LCSS. Відповідні вікна за часом позначаються сірою областю (δ = 5)

Обчислене значення LCSS не обмежене і залежить від довжини порівняних послідовностей. Для підтримки послідовностей змінної довжини значення повинні бути нормалізовані. Можна вивести нормоване відстань на основі подібності LCSS таким чином:

Визначення 2.4: Відстань $D_{\delta,}$ виражена в термінах подібності LCSS між двома траєкторіями A і B, задається як:

$$D_{\delta,\epsilon}(A,B)=1-\frac{LCSS_{\delta,\epsilon}(A,B)}{min(n,m)} \quad (2.11)$$

Приступимо до розширення базової функції DTW для багатовимірних траєкторій. Для простоти буде наведено приклад двовимірних траєкторій.

L_p з формули 2.11 - будь-яка L_p-норма. Для обчислення DTW використовується відхід динамічного програмування, подібна до LCSS. Знову ж таки, обмежуючи відповідну область в межах δ, час, необхідний для обчислення DTW, знаходиться в порядку O(δ(n+m)), близько до LCSS. Щоб змістовно порівняти відстані між послідовностями різної довжини, можна нормалізувати загальну відстань DTW за довжиною деформації. На рисунку 2.12 показаний

приклад відповідності DTW між двома траєкторіями. Для використання подібного алгоритму на великих даних, потрібно застосовувати оптимізації. До них належить обмеження області викривлення.

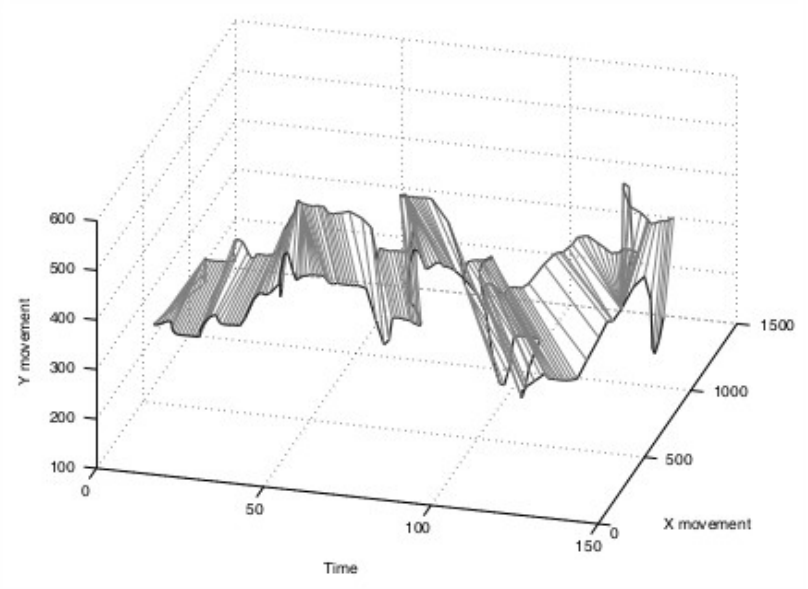


Рисунок 2.12 - Порівняння DTW між двома двовимірними траєкторіями

Визначення 2.5: Відстань DTW між двовимірними часовими послідовностями A і B визначається наступним чином:

$$DTW(A, B) = L_p((a_{x,n}, a_{y,n}), (b_{x,m}, b_{y,m})) + \min\{DTW(Head(A), Head(B)), DTW(Head(A), B), DTW(A, Head(B))\}. \quad (2.12)$$

Обмеження дозволеної довжини викривлення у часі (параметр δ) може істотно прискорити виконання алгоритму динамічного програмування для моделей DTW та LCSS. Однак ми продемонструємо на прикладі, що обмежене викривлення у вікні δ надає більше переваг:

- для більшості наборів даних деформація відносно повної довжини не потрібна для досягнення високої точності співпадіння. На практиці обмеження часу викривлення не більше 20% довжини послідовності виявляється достатнім у більшості застосувань. Зазвичай збіг в парній подібності між послідовностями

спостерігається після викривлення певної довжини. Отже, дозволивши більш широке викривлення за рахунок тривалого часу виконання не призведе до суттєвих змін у обчислюваній схожості.

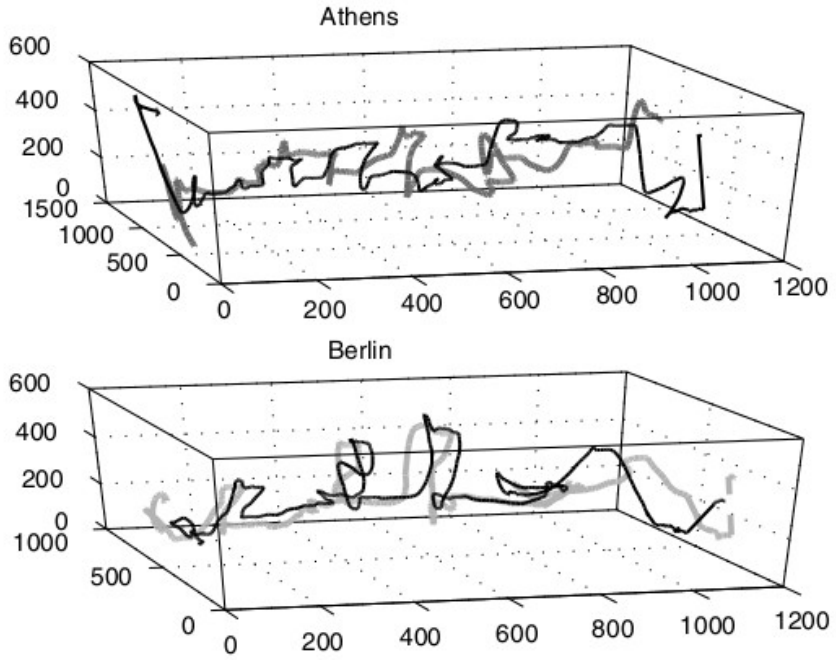


Рисунок 2.13 - Два “слова” з набору даних cameraMouse (по два екземпляри в кожному)

- у наведеному прикладі використано кілька примірників різних слів, створених за допомогою програми “cameraMouse” шляхом відстеження руху людини у часі. Наприклад, відстежуючи рух якоїсь людської властивості (наприклад, кінчика носа), користувач може використовувати «віртуальну клавіатуру» та «писати» слова.

Програма орієнтована в першу чергу на людей з обмеженими рухами, забезпечуючи більш простий комунікаційний інтерфейс з комп'ютером. Таким чином були зібрані просторово-часові послідовності, що представляють різні слова, рисунок 2.13. На рисунку 2.14 можна спостерігати тенденцію обчислюваної подібності LCSS при збільшенні довжини викривлення більше ніж п’ять парних обчислень. Очевидно, що обчислювана подібність

дотримується закону зменшення віддачі, оскільки довжина викривлення більше 20% не змінює схожість обчисленої послідовності.

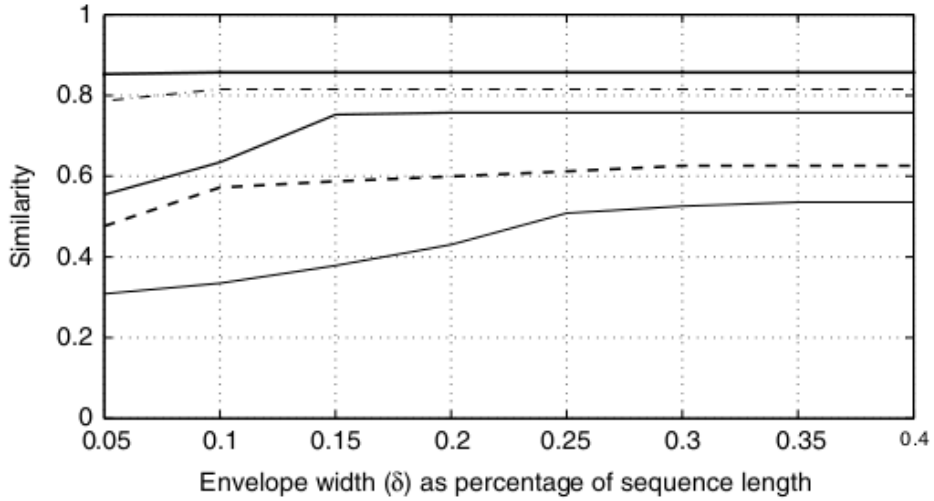


Рисунок 2.14 - Попарна подібність LCSS із зростанням вікон викривлення

Подібні результати були відзначені й для інших наборів даних часових послідовностей, які охоплюють області ботаніки, відстеження відео, розпізнавання тексту тощо.

У деяких наборах даних розширене викривлення не тільки не дає додаткових переваг, але й шкодить точності. Був проведений наступний простий експеримент: кожну послідовність тестових наборів даних позначили за допомогою класифікаційної схеми найближчого сусіда «leave-one-out». Таким чином для кожної траєкторії в наборі даних ми знаходимо найближчу відповідність відповідно до співпадіння LCSS. Класифікація вважається правильною лише в тому випадку, якщо мітка найближчого сусіда є такою ж, як і оригінальна мітка траєкторії. Загальна точність реєструється шляхом повторення експериментів для всіх послідовностей даних. Рисунок 2.15 ілюструє результати точності, отримані для набору даних cameraMouse та для підмножини набору даних австралійської мови жестів. Ми спостерігаємо, що надмірне викривлення може пошкодити продуктивність у певних наборах даних. У той час як для даних cameraMouse розширене викривлення не

покращує точність, у випадку ASL, збільшення деформації до більш ніж 30%, погіршує точність.

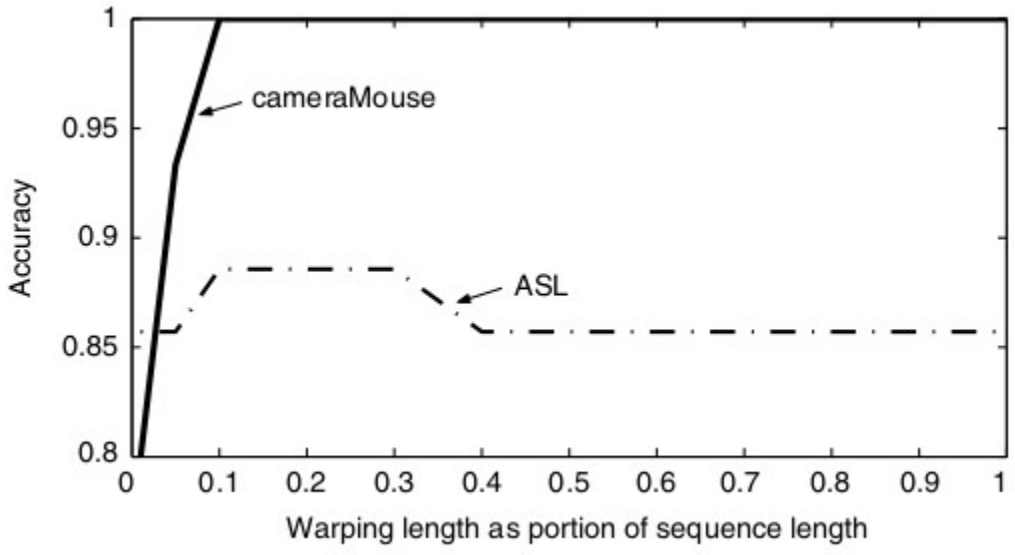


Рисунок 2.15 - Точність класифікації "leave-one-out" для двох наборів даних

Очевидно, що надмірне викривлення може спотворювати справжні відстані траєкторії та створювати штучні співпадіння, дозволяючи тривалі і вироджені відповідності. Тепер перейдемо до обговорення як саме треба налаштувати параметри моделі LCSS.

Зрозуміло, що значення для δ і ε залежать від реалізації та набору даних. Для більшості наборів даних, було виявлено, що встановлення δ на понад 10–20% від довжини траєкторій оцінку подібності суттєво не змінило. Крім того, після певного моменту подібність стабілізується до певного значення. Тому параметр δ можна спочатку наблизити за допомогою міри DTW (для якої використовується лише один параметр) та оцінки довжини викривлення, після чого точність класифікації не покращується.

Згодом параметр можна оцінити, досліджуючи кілька значень для виявленого параметра δ і записуючи те, що забезпечує найкраще розділення

класу. Використання значення в межах 10–25% від стандартного відхилення досліджуваних траєкторій, дає хороші та інтуїтивні результати.

Навіть незважаючи на те, що встановлення значення другого параметра є може зменшити зручність використання LCSS моделі, ми вважаємо за краще використовувати цей метод, оскільки він забезпечує додаткові можливості просторової фільтрації, які не може надати жодна інша міра. Наприклад, можливим запитом на сторожовій вежі аеропорту може бути: "Знайдіть усі літаки, які йшли за аналогічним маршрутом, як площина X і завжди знаходилися в радіусі ε". На такі запити можна легко відповісти, використовуючи лише модель LCSS.

2.4.3 Модифікація DTW для випадків з викривленням у часі

У цьому підрозділі представлено інтерпретацію оптимального деформаційного шляху TAM, що базується на алгоритмі DTW.

Оптимальний шлях викривлення встановлює попарну залежність між значеннями обох рядів. Ця схожість дозволяє охарактеризувати перетворення на часовій осі між ними. Незважаючи на те, що в літературі запропоновано безліч покрокових моделей для обчислення деформації шляху, ми розпочали з вивчення основної схеми кроків, яка розглядає вертикальні, горизонтальні та діагональні сегменти.

Припустимо, що можливе викривлення часу посилається на послідовність X, яка побудована на осі x. На рисунку 2.16 наведені можливі похильні. Оптимальний деформаційний шлях, накладений на акумульовану матрицю витрат, що містить усі доступні напрямки кроків. Горизонтальний, вертикальний та діагональний сегменти представляють інтервали просування, затримки та фази відповідно.

Горизонтальний відрізок позначається як $w_{k+1}-w_k = (1,0)$. У цьому випадку точка послідовності Y асоціюється з однією або декількома послідовними точками відповідної послідовності X. Ця ситуація ілюструє часове упередження, оскільки один і той самий момент часу підтримується у послідовності Y, у вихідному ряду проходить кілька часових моментів.

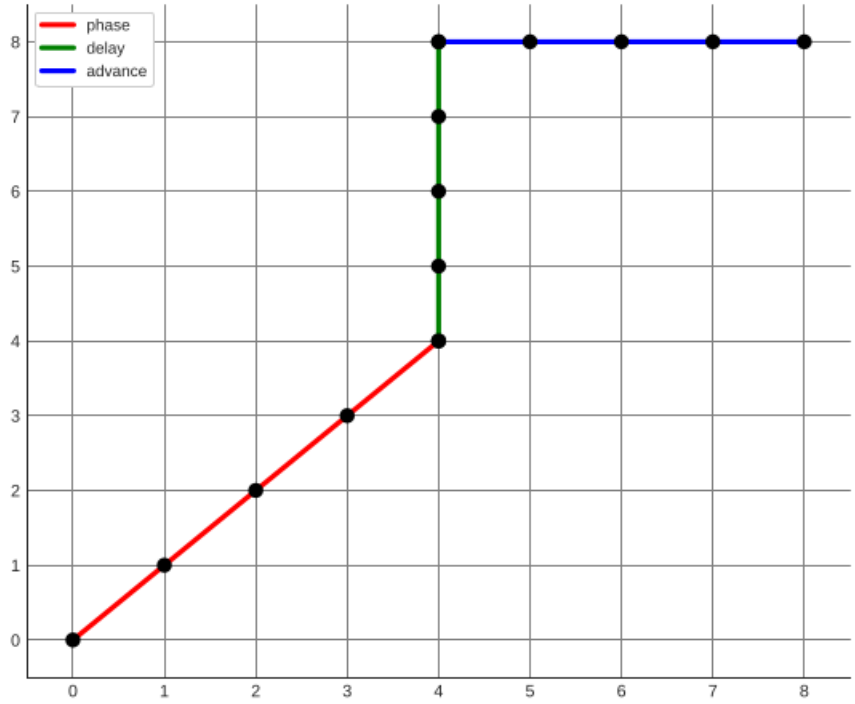


Рисунок 2.16 - Приклад штучного оптимального деформаційного шляху

Вертикальний відрізок позначається як $w_{k+1}-w_k = (0, 1)$. Такий випадок виникає, коли індекс X асоційований з одним або кількома послідовними точками послідовності Y. Тому присутня часова затримка, оскільки послідовність Y прогресує в часі, а вихідна послідовність залишається в тому самому моменті.

Діагональний відрізок позначається як $w_{k+1}-w_k = (1,1)$. У цьому випадку немає часу викривлення і сигнали можна вважати таким, що відображає фазовий зсув між ними.

Ідея, що лежить в основі запропонованої відстані, полягає в вимірюванні вартості трансформації у часі заданої ЧП відносно іншої ЧП. Використовуючи оптимальний шлях вирівнювання між двома часовими рядами, ми можемо отримати інформацію у часовому вимірі, що дозволяє характеризувати інтервали, коли ряд перебуває у фазі, упередженні або в затримці.

Розглянемо ще раз дві послідовності X довжиною N ∈ N та Y довжиною M ∈ N. Протягом повної довжини кожної послідовності сигнали можуть вважатись такими, що демонструють фазну та позафазну поведінку. У випадку, якщо сигнали поза фазою, одна послідовність може вважатися упереджуючою по відношенню до іншої. Ця характеристика є зворотною, як якщо X є упереджуючою по відношенню до Y, Y можна вважати затриманою по відношенню до X.

Якщо припустити, що Y затримується по відношенню до X, то загальний час, який Y затримується по відношенню до X, позначається як $\theta_{xy}^{\leftarrow}$, а час, який може бути упередженим, позначається як $\theta_{xy}^{\rightarrow}$. За допомогою цього запису ми можемо записати залежність між упередженням, затримкою та довжиною обох сигналів у вигляді:

$$\left(\left(\theta_{xy}^{\rightarrow} — \theta_{xy}^{\leftarrow}\right)\right) = \left(\left(M - N\right)\right) . \tag{2.13}$$

Загальний час, коли обидві ЧП знаходяться у фазі, представлений символом $\overline{\theta_{xy}}$. Під час повної довжини сигналу Y частка упередження $\psi^{\rightarrow}$, затримки $\psi^{\leftarrow}$ та фази $\overline{\psi}$ до X може бути обчислена як:

Ця відстань “штрафує” сигнали там, де є упередження або затримка, і “заохочує” ЧП, які перебувають у фазі між собою. Зі збільшенням відстані збільшується і несхожість між обома сигналами. Таким чином, якщо сигнали постійно знаходяться у фазі то справедливо $\overline{\psi} = 1, \theta_{xy}^{\rightarrow} = 0, \psi^{\rightarrow} = 0, \theta_{xy}^{\leftarrow} = 0 \text{and} \, \psi^{\leftarrow} = 0$.

$$\psi^{\rightarrow} = \frac{\theta^{\rightarrow}_{xy}}{N},$$
$$\psi^{\leftarrow} = \frac{\theta^{\leftarrow}_{xy}}{M}, \qquad (2.14)$$
$$\overline{\psi} = \frac{\overline{\theta_{xy}}}{min(N, M)},$$

де $\Gamma \in \left\{ R_0^+ | \Gamma \in [0:3] \right\}$.

Відстань ТАМ має мінімальне дозволене значення (Г = 0), і сигнали можна вважати рівними у часовій області. Найвище значення несхожості простежується при Г = 3, де $\theta^{\rightarrow}_{xy} = N \Rightarrow \psi^{\rightarrow} = 1$, $\theta^{\leftarrow}_{xy} = M \Rightarrow \psi^{\leftarrow} = 1$ і, отже, ψ = 0.

Важливо зазначити деякі міркування щодо топології запропонованого заходу:

1. Умова тотожності нерозрізних ЧП не виконується : Г(X, Y) = 0 ≠> X = Y. Насправді два сигнали можуть бути рівними за часом і мати різницю за амплітудою. Тривіальне застосування цієї відстані до двох подібних сигналів у часі зі зміщенням по амплітуді доводить це припущення.

2. Симетрія спостерігається, оскільки поняття упередження та затримки між двома часовими рядами є зворотним: $\theta^{\rightarrow}_{xy} = \theta^{\leftarrow}_{yx} = \alpha$, $\theta^{\leftarrow}_{xy} = \theta^{\rightarrow}_{yx} = \beta$ крім того, $\overline{\theta_{xy}} = \overline{\theta_{yx}} = \overline{\theta}$.

Вищезазначені міркування показують, що ТАМ не можна розглядати як метрику, оскільки вона не може гарантувати ідентичність нерозрізних ЧП. Однак ми можемо констатувати, що це майже метрична метрика, оскільки вона повністю задовольняє як умові незаперечності, так і симетрії. Важливо підкреслити, що для обчислення відстані ТАМ потрібно лише встановити попарне співвідношення між елементами кожного часового ряду. Це попарне відношення забезпечує необхідне вирівнювання для обчислення затримок, упередження та фазових періодів між сигналами. Таким чином, альтернативні

методи вирівнювання ЧП до DTW також можуть бути використані для обчислення ТАМ.

ТАМ можна обчислити безпосередньо з деформації шляху DTW на основі таких припущень:

Нехай $\Delta[W_k^*] = w^*_{k+1} - w^*_k$ кінцева різниця між двома послідовними координатами оптимального шляху викривлення в точці k, представленої як двовимірний вектор. Оскільки оптимальний шлях викривлення обмежений вертикальними, горизонтальними та діагональними відрізками, $\Delta[W_k^*]$ також обмежено значеннями $\Delta[W_k^*] \in \{(1, 1), (1, 0), (0, 1)\}$. Вертикальні відрізки матимуть значення $\Delta[W_k^*] = (0,1)$, позначене тимчасовою затримкою; горизонтальні відрізки матимуть значення $\Delta[W_k^*] = (1,0)$, що позначає часове упередження, а діагональні відрізки представлятимуть $[W_k^*] = (1,1)$, що позначає фазу між ЧП.

Кількість точок при упередженні, затримці та фазі можна безпосередньо обчислити з оптимального шляху згідно рівняння 2.15:

$$\begin{aligned} \vec{\delta_k} &= \begin{cases} 1, \Delta[W_k^*] = (1,0) \\ 0, otherwise \end{cases}; \\ \overleftarrow{\delta_k} &= \begin{cases} 1, \Delta[W_k^*] = (0,1) \\ 0, otherwise \end{cases}; \\ \overline{\delta_k} &= \begin{cases} 1, \Delta[W_k^*] = (1,1) \\ 0, otherwise \end{cases}. \end{aligned} \tag{2.15}$$

Отже, ТАМ можна обчислити згідно рівняння 2.16:

$$\Gamma = \frac{1}{N}\sum_{k=1}^{K} \vec{\delta_k} + \frac{1}{M}\sum_{k=1}^{K} \overleftarrow{\delta_k} + \left(1 - \frac{1}{min(N,M)} * \sum_{k=1}^{K} \overline{\delta_k}\right). \tag{2.16}$$

Повертаючись до аналізу рисунку 2.16, ми бачимо, що шлях починається в фазі, затримується протягом чотирьох сегментів і закінчується упередженням

ще протягом чотирьох сегментів. Ми можемо записати результат $\delta^{\rightarrow}, \delta^{\leftarrow}$ and $\bar{\delta}$, а δ як:

$$\begin{aligned} \delta^{\rightarrow} &= \{0,0,0,0,0,0,0,0,1,1,1,1\}; \\ \delta^{\leftarrow} &= \{0,0,0,0,1,1,1,1,0,0,0,0\}; \\ \bar{\delta} &= \{1,1,1,1,0,0,0,0,0,0,0,0\}; \end{aligned} \tag{2.17}$$

Розглянемо випадок рівних за довжиною ЧП. Коли ми встановимо порівняння між двома часовими рядами, X і Y, з рівними довжинами N = M = L, тривалість інтервалів, де X є заздалегідь, і затримка порівняно з Y, повинна бути рівною. Ця властивість є наслідком того, що обидва сигнали мають однакову довжину. Тому, незважаючи на можливу затримку, сигнал повинен мати принаймні післяопераційний просування, щоб закінчити в ту ж мить, що й інша послідовність, і навпаки. Оскільки $\theta_{xy}^{\rightarrow} = \theta_{xy}^{\leftarrow} = \theta$, відстань TAM можна спростити безпосередньо до:

$$\Gamma(X,Y) = \frac{3\theta}{L} \,. \tag{2.18}$$

Рисунок 2.17 представляє приклад, коли набір з чотирьох штучних сигналів генерується шляхом спотворення першої послідовності. Хоча в цьому прикладі евклідова відстань збільшується зі збільшенням спотворення часу, вона не відображає значущого вимірювання. Оскільки вона чутлива лише до амплітудної подібності, у випадку, якщо послідовність має зсув на значеннях плато, вона буде створювати більшу величину відстані, незалежно від ступеня деформації часу. DTW вирівнює кожну пару сигналів перед обчисленням відстані і таким чином виробляє однакові оцінки для всіх прикладів. Однак TAM виробляє значущу оцінку, яка відображає витрати на стиснення та розширення в часі конкретного сигналу. Фактично відстань збільшується зі ступенем спотворення часу, присутнього в кожній послідовності. Сигнал A ідентичний вихідному сигналу. Плато від сигналу B має упередження в 10

секунд. Щоб закінчити в ту ж мить, сигнал входить в затримку на останньому сегменті ще на 10 секунд. Сигнал C спотворюється половину свого часу як $\theta_{xy}^{\rightarrow}=\theta_{xy}^{\leftarrow}=15\Rightarrow \Gamma=\frac{3*15}{30}$. Сигнал D являє собою значне упередження плато, яке представлене як єдиний пік з подальшим значним запізненням до кінця сигналу. Зауважимо, що у всіх прикладах два перший та останній зразки знаходяться у фазі, що заважає досягти максимального теоретичного значення ТАМ.

Розглянемо випадок різних за довжиною ЧП. Часові послідовності з різною тривалістю можуть представляти різні типи поведінки в тому, як вони відбуваються протягом часу. Рисунок 2.18 представляє групу з чотирьох спотворених у часі сигналів. Сигнали генерували на базі першої послідовності шляхом модифікації відповідних векторів часу для імітації затримки та упередження. Сигнал A розділяє однакове представлення часу з опорним сигналом. Сигнал B має початкову затримку на 10 секунд, а потім переходить у фазу на іншу частину своєї довжини. Відстань ТАМ 0,25 позначає стиснення. Більша відстань спостерігається для сигналу C, оскільки є більш значне просування.

Сигнал D відображає лінійне відставання затримки, оскільки два моменти сигналу пов'язані з одним миттю сигналу A, виробляючи значення відстані 0,5. Евклідову відстань обчислювали шляхом лінійної інтерполяції всіх сигналів для поділу однакової довжини відліку. Ця процедура гарантує, що сигнали мають однакові довжини для обчислення евклідової відстані. Хоча сигнал D відрізняється від опорного у часовій області, евклідова відстань дорівнює нулю.

Попередні приклади дозволили краще описати природу нашого основного внеску: надати нове вимірювання відстані, здатне характеризувати ступінь викривлення часу між часовими послідовностями, які можуть бути подібними за амплітудою. Модифікація DTW-D складається із відношення між DTW та евклідовою відстанню.

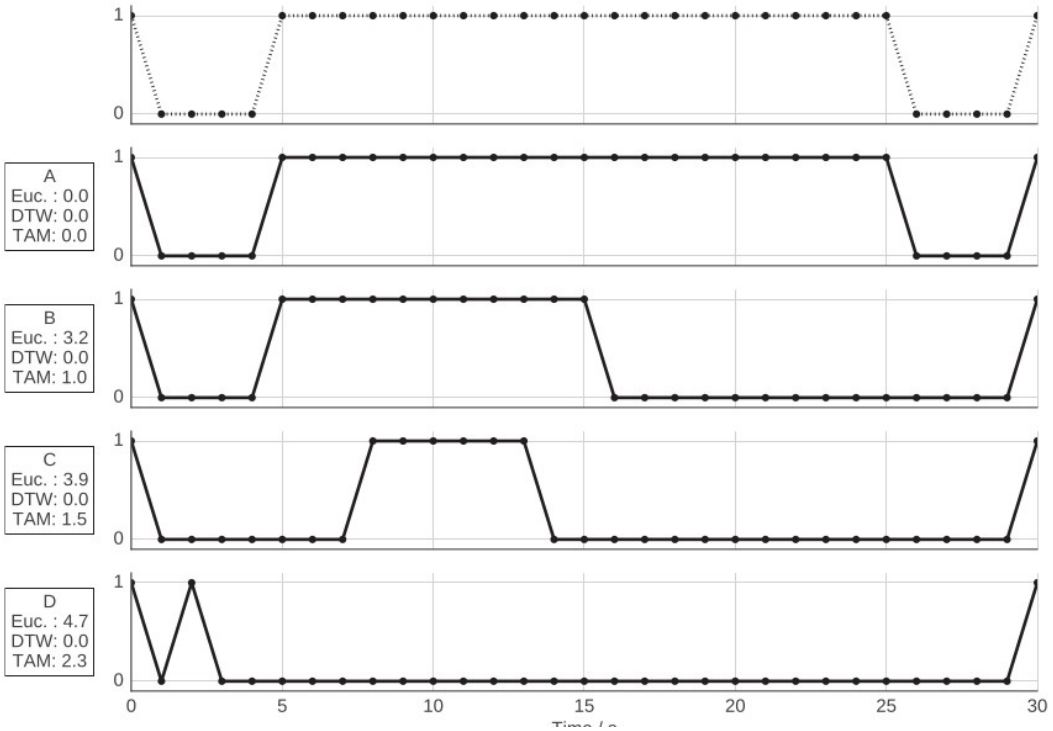


Рисунок 2.17 - ЧП однакової послідовності

Можна потенційно стверджувати, що таке співвідношення може бути наближенням для вимірювання деформації, оскільки воно вимірює кількість викривлень, необхідних для відповідності заданому часовому ряду по відношенню до евклідової відстані (яка взагалі не вимагає викривлення). Однак DTW-D зрештою не впорається з задачею у представлених прикладах додатків. Оскільки сигнали однаково схожі за амплітудою, DTW матиме значення 0 і, отже, DTW-D також не зможе забезпечити значущу оцінку.

На завершення, наївний підхід порівняння часових рядів на основі часового виміру був би виключно порівнянням довжини послідовностей. Однак TAM оцінює поведінку в часі з точки зору затримки, упередження та фази протягом часу кожної послідовності, і, отже, не обмежується лише кінцевими точками кожної послідовності.

Отже, навіть для послідовностей, що мають однакову довжину, часова інформація може бути витягнута, що було б неможливо, якби було проведено пряме порівняння між довжинами сигналу.

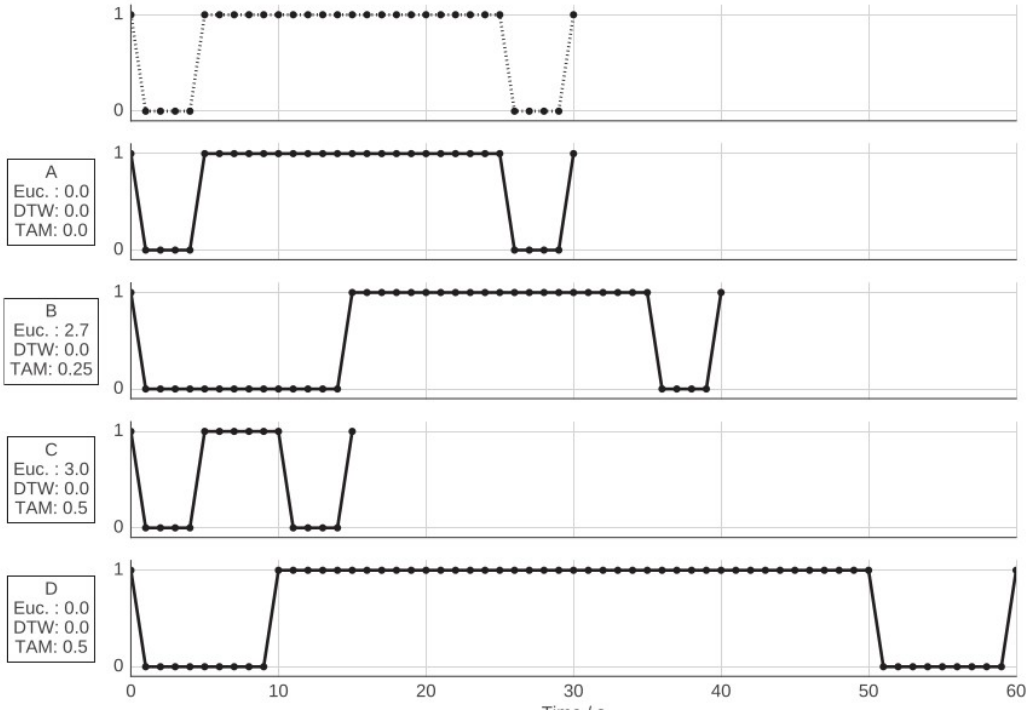


Рисунок 2.18 - ЧП різної послідовності

2.5 Побудова індексу

2.5.1 LCSS індексування

В даному підрозділі розглянуто першу з двох схем індексування, засновану на намірах подібності LCSS, DTW і TAM.

Хоча накладення обмеження на розмір пошукового діапазону δ може сприяти прискоренню пошуку, обчислення пошукового запиту все одно може займати час близький до квадратичного, якщо δ - значна частина довжини послідовності. Тому порівняння запиту з усіма послідовностями стає недосяжним для великих баз даних. Ми шукаємо способи уникнути перевірки послідовностей, дуже віддалених від заданого запиту. Це можна досягти, виявивши близьке запиту співпадіння якомога раніше. Можна використовувати індекс у вигляді дерева, що дозволяє виключити більшість віддалених співпадінь та скорочує кількість перевірок у логарифмічній залежності. Лише для деяких кваліфікованих послідовностей буде виконано дорогий (але точний)

алгоритм з квадратичним часом. Ця схема показана на рисунку 2.19. Дуже віддалені траєкторії можна відкинути, використовуючи етап швидкої попередньої фільтрації. Для решти траєкторій можна використовувати дорогу, але точну міру подібності. Існують певні етапи попередньої обробки, яких необхідно дотримуватися:

- послідовності сегментовані на мінімальні обмежуючі прямокутники (МОП), які зберігаються у R-дереві;

- враховуючи запит Q, області можливого співпадіння виявляються шляхом побудови мінімальної обмежуючої обвідної ($МОО_Q$);

- $МОО_Q$ розкладається на МОП, які перевіряються в індексі;

- на основі перетинів МОП обчислюються оцінки подібності, а точний LCSS (або DTW) виконується лише на кваліфікованих послідовностях.

Ці кроки відображені на рисунку 2.20.

Розглянемо спочатку одномірну послідовність $A = (a_{x,1}, ..., a_{x,n})$. Ігноруючи зараз параметр ε, ми хотіли б показати на наступному прикладі дуже швидке $LCSS_\delta$ співпадіння між послідовністю A і деяким запитом $Q = (q_{x,1},., q_{x,n})$. Припустимо, що ми повторюємо кожну точку $q_{x,i}$ для екземплярів δ часу до та після і. Область огину, яка включає всі ці точки, визначає області можливого співпадіння. Ніщо за межами цієї області огину ніколи не може бути співставлено.

Ця область називається мінімальною обмежуючою обвідною (МОО) послідовності. Оскільки можливість співпадіння у межах ε у просторі буде включена, нова обвідна повинна розширювати ε вище та нижче оригінальної обвідної у просторі, рисунок 2.21. Поняття обмежувальної області огину можна тривіально поширити на інші виміри.

Наприклад, МОО(δ, ε) для двовимірної траєкторії $Q = ((q_{x,1}, q_{y,1}), ..., (q_{x,n}, q_{y,n}))$ охоплює таку область:

$$EnvLow \leq MBE(\delta, \varepsilon) \leq EnvHigh, \qquad (2.19)$$

де для розмірності d на час i:

$$LCSS_{\delta,\epsilon}(A,B)=\begin{cases} EnvHigh_{d,i}=max(q_{d,i}+\epsilon), |i-j|\le\delta \\ EnvLow_{d,i}=min(q_{d,i}-\epsilon), |i-j|\le\delta \end{cases}. \quad (2.20)$$

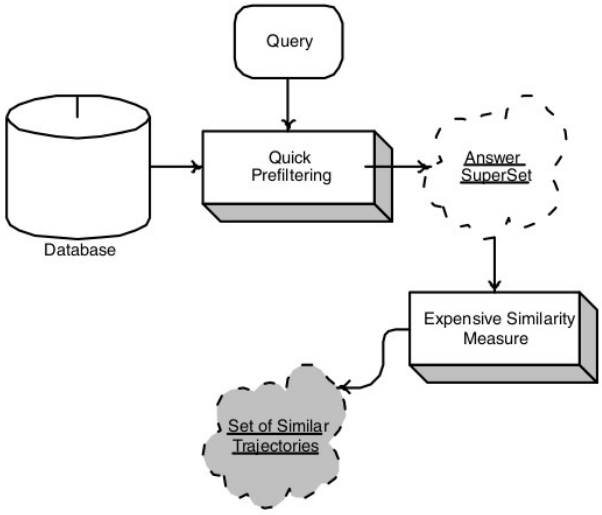


Рисунок 2.19 - Схема індексування LCSS

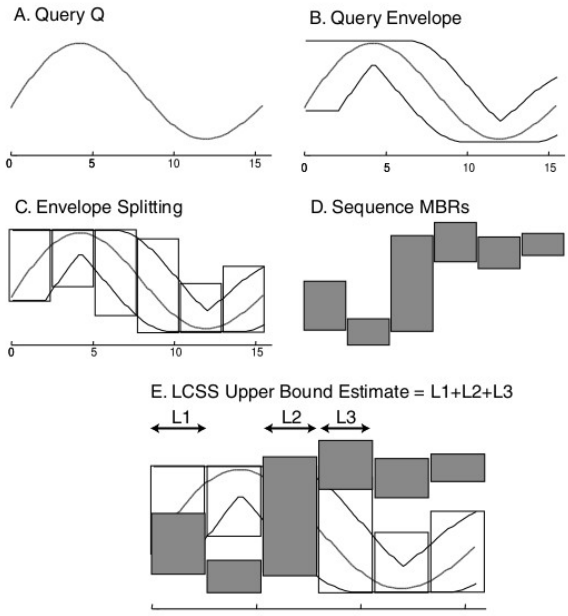


Рисунок 2.20 - Одновимірна ілюстрація запропонованого підходу. Запит поширюється на обмежувальну область, яка, у свою чергу, наближається до набору МОП. Перекриття між запитом та індексом МОП передбачає області можливого узгодження та траєкторії кандидатів на вихід.

Подібність LCSS між оболонкою Q та послідовністю A визначається як:

$$LCSS(MBE_Q, A) = \sum_{i=1}^{n} \begin{cases} 1 \; if \; a[i] \; within \; envelope \\ 0 \; otherwise \end{cases} \tag{2.21}$$

Наприклад, на рисунку 2.21 подібність LCSS між MOO_Q і послідовністю A дорівнює 46. Це значення являє завищену або верхню межу подібності LCSS запиту Q і послідовності A. MOO_Q можна також використовувати для обчислення нижньої межі відстані між траєкторіями.

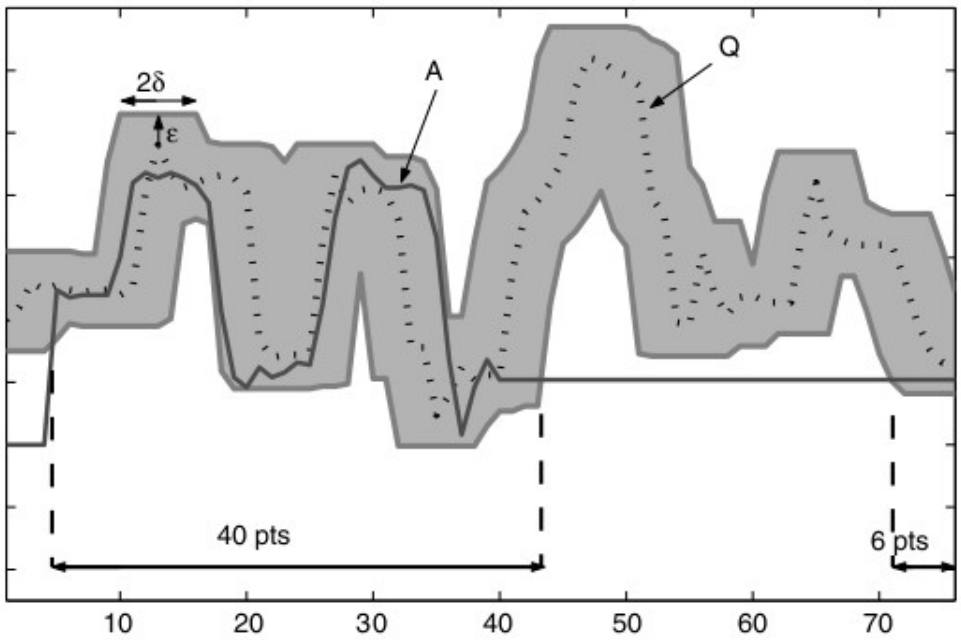


Рисунок 2.21 - Мінімальна обмежуюча обвідна (MOO) протягом δ у часовому вимірі та в просторі послідовності ε. Ніщо з того, що лежить поза цією областю огину, не може бути співставлено

Перш ніж представити нижнє обмеження для DTW, коротко переглянемо дві інші нижні межі, що з’являються в літературі. Обидві нижні обмежувальні функції спочатку були визначені лише для одновимірних часових послідовностей (і пояснюються нижче для одновимірного випадку). Однак їх розширення на багатовимірний часовий ряд досить просте.

Обмежуюча знизу функція, введена Кімом (далі LB-Kim), працює шляхом вилучення векторів з чотирьох властивостей з кожної послідовності. Властивостями являються перший і останній елементи послідовності разом з максимальним і мінімальним значеннями. Максимум між різницями у відповідних ознаках у квадраті відображається як нижня межа. Максимальна різниця у квадраті між першою (a), останньою (d), мінімальною (b) та максимальною точками (c) двох послідовностей складає як нижня межа, рисунок 2.22.

Обмежуюча знизу функція, запроваджена Yi (далі - LB-Yi), використовує спостереження, що всі точки однієї послідовності, які є більшими (меншими), ніж максимум (мінімум) іншої послідовності, повинні принаймні додавати квадрат різниці їхньої величини та максимального (мінімального) значення іншої послідовності до кінцевої відстані DTW. Сума квадратичної довжини сірих ліній представляє мінімум внеску відповідних точок у загальну відстань DTW і, таким чином, може бути повернута як міра нижнього обмеження, рисунок 2.23.

Зауважте, що хоча LB-Kim тривіально індексується, LB-Yi - ні (оскільки він вимагає початкового запиту). Корисність останнього полягає в його використанні спільно з технікою приблизної індексації DTW, яка використовує FastMap. Ідея полягає в тому, щоб вбудувати послідовності в евклідовий простір таким чином, щоб структура відстаней початкового простору була приблизно збережена. Тоді для індексації евклідового простору може бути використана традиційна багатовимірна структура індексу. Функція LB-Yi використовується для обрізання деяких неминучих хибних хітів, які будуть введені цим методом.

Намагаючись передати наближено відстань DTW, спочатку побудуємо MOO запиту Q, описаного нижньою та верхньою обвідною (як визначено вище, але налаштування = 0). На відміну від моделі LCSS, замість обчислення областей, що перекриваються, ми оцінюємо відстань MOO_Q від усіх інших послідовностей.

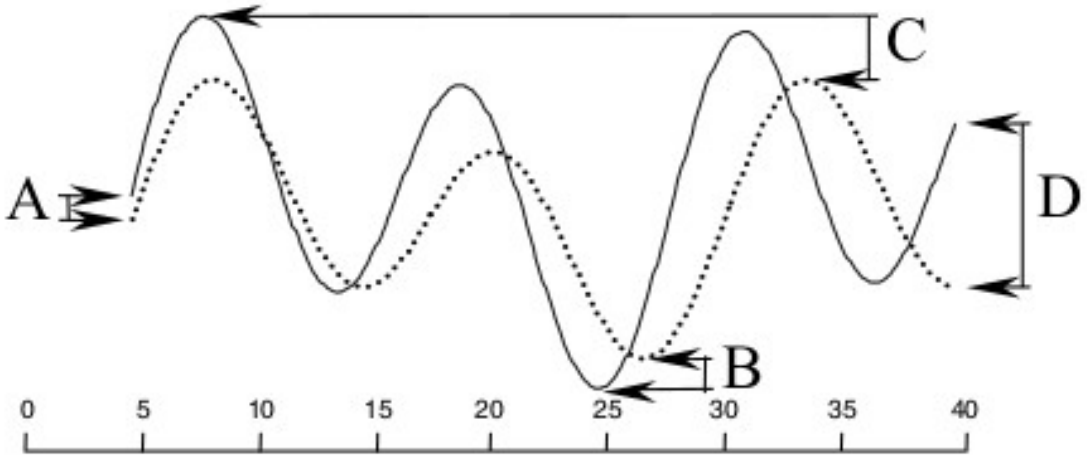


Рисунок 2.22 -Візуальне позначення нижньої межі, введена Кімом.

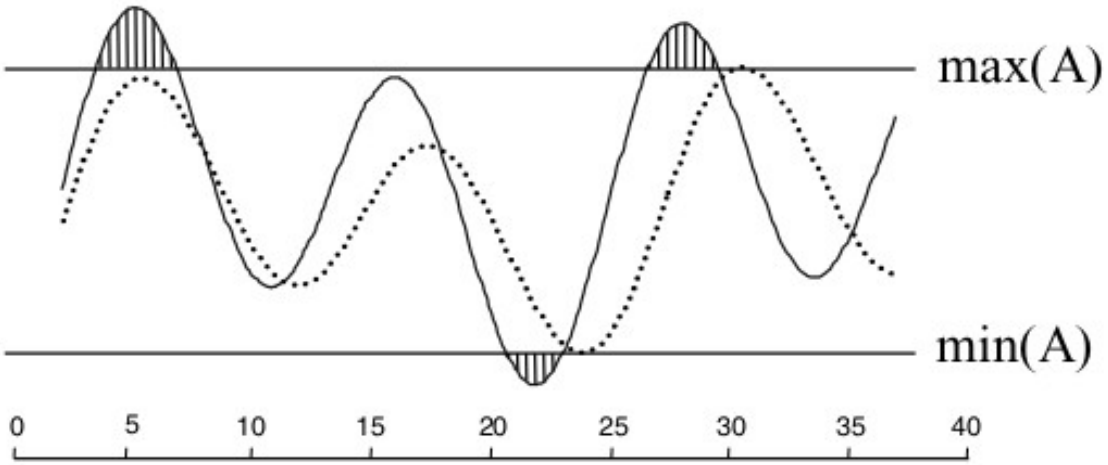


Рисунок 2.23 - Візуальне позначення нижньої межі захід, введений Yi

Для послідовностей Q і A з розмірністю D відстань між A і MOO_Q становить:

$$DTW(MBE_Q, A) = \sqrt{\sum_{d=1}^{D}\sum_{i=1}^{n}\begin{cases}(a_{d,i} - EnvHigh_{d,i})^2 \; if \; a_{x,i} > EnvHigh_{d,i} \\ (a_{d,i} - EnvLow_{d,i})^2 \; if \; a_{x,i} < EnvLow_{d,i} \\ 0 \; otherwise\end{cases}} \quad (2.22)$$

Функція може сприйматися як евклідова відстань між будь-якою частиною потенційно співпадаючої послідовності, що не потрапляє в область

огину послідовності та найближчої (ортогональної) відповідної секції області огину Q, як зображено на рисунку 2.24. Послідовність Q початкової точки (пунктирна) укладена в мінімально обмежувальну оболонку. Сума в квадраті відстаней від кожної частини послідовності A, що не потрапляє в MOO_Q, до найближчого ортогонального краю MOO_Q повертається як нижня межа. Ця нижня межа може розглядатися як узагальнення для багатовимірних часових рядів методики.

Ця відстань обмежує знизу фактичну відстань викривлення в часі. Для одновимірного випадку це вже було доведено, розширення до кількох вимірів аналогічне.

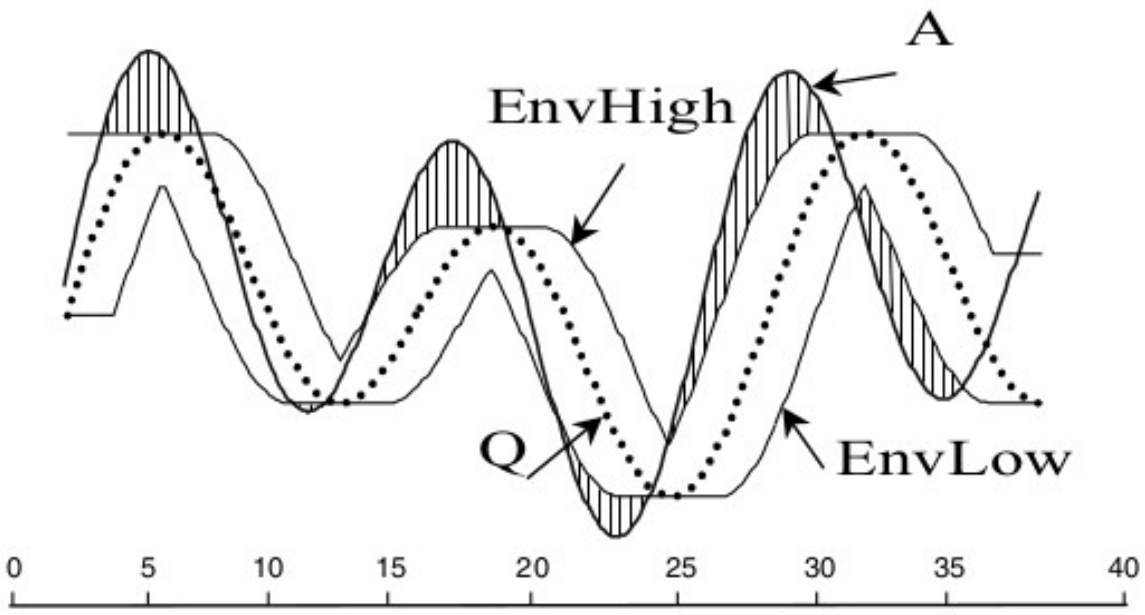


Рисунок 2.24 - Ілюстрація функції нижнього обмеження DTW

Оскільки жорсткість цієї межі пропорційна кількості та довжині сірих ліній штрихування, то, принаймні, у цьому прикладі ми можемо побачити, що нова нижня межа забезпечує більш жорстку межу, ніж LB-Kim або LB-Yi. Слід також зазначити, що запропонований захід завжди буде принаймні таким же хорошим, як LB-Yi, що є особливим випадком, що виникає, коли обмежувальна область дозволяє охопити всю довжину послідовності.

2.5.2 Метод індексації STB

Для наочності ми будемо називати необроблені часові дані як часові послідовності, а кусочне подання часової послідовності як послідовності. Часову послідовність, відібрану у k точках, позначено у вигляді великої літери A.

$$A\{AXL, AXR, AYL, AYR\}. \quad (2.23)$$

1[й] сегмент послідовності A представлений лінією між (AXL_i, AYL_i) і (AXR_i, AYR_i). Рисунок 2.25 ілюструє це позначення.

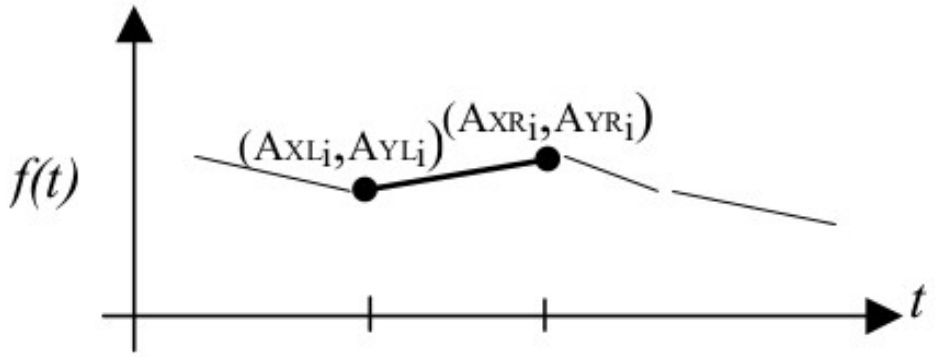


Рисунок 2.25 - Представляємо часові ряди послідовністю прямих відрізків

Однією з переваг використання кусочно-лінійних відрізків є те, що міру відстані можна обчислити приблизно в k/K разів швидше, ніж при роботі з необробленими даними. Наступною перевагою є те, що можна визначити різні міри відстані для різних областей.

Відстань між двома послідовностями A і B позначається як DS(A,B) і розрахована на те, щоб якомога точніше наблизити точність міри відстані визначену на “сирих” даних. Однак ми зазначимо, що DS - це метрика, і вона має підпорядковується правилу нерівності трикутника.

Для побудови індексу переміщуємо ковзаюче вікно фіксованої довжини вздовж послідовності даних. Вікно містить рівномірну сітку, як показано на рисунку 2.26. Сітка шаблону (показано лише 4 вікна для наочності) переміщується вздовж всієї послідовності та вирівнюється по лівому краю кожного сегмента. Підпослідовність, що потрапляє в сітку (показано жирним шрифтом), вивчається і використовується для отримання бітової строки. Цей штриховий рядок використовується для визначення того, в який корзині слід розмістити копію підстроки

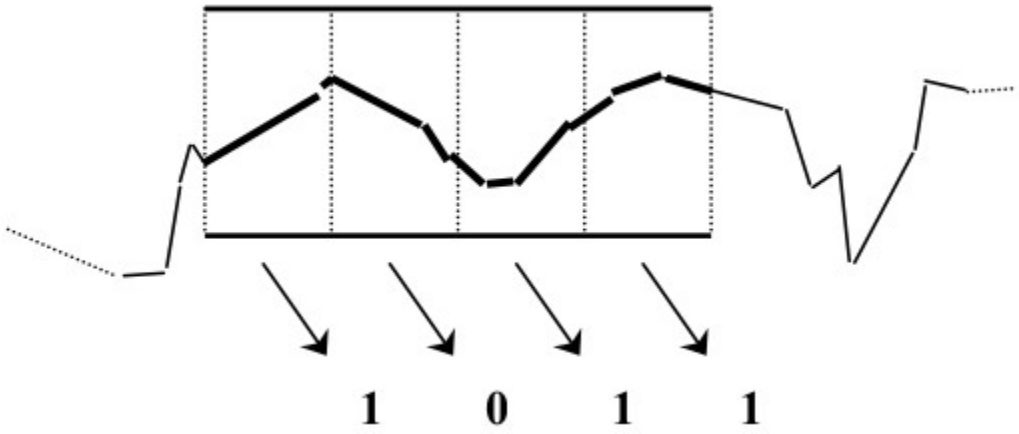


Рисунок 2.26 - Зображення методики декластеризації, яка використовується в схемі індексації

Відрізок часової послідовності, що потрапляє під кожну частину сітки, вивчається та дискретизується на два можливі класи "вгору" або "не вгору", які представлені відповідно як один або нуль. Бітове представлення, отримане при зчитуванні цих бітів зліва направо, використовується для того, щоб визначити, у яку корзину буде вказувати покажчик часової послідовності. Після того як всі підстроки скопійовані у відповідні корзини, ми порівнюємо кожну підстроку з кожною іншою підстрокою у межах однієї корзини, зберігаючи результати у матриці відстані. Після завершення індексації у нас є набір корзин, які позначені унікальними бітовими патернами. Кожна корзина містить копію однієї або декількох підстрок вихідних даних разом з попередньо обчисленою

матрицею відстані, яка містить відстань між кожною можливою парою послідовностей у цій корзині. Ця матриця відстані дозволяє скорочувати пошуковий простір у корзині за допомогою трикутної нерівності.

Задаючи запит, алгоритм пошуку досягає майже лінійних часів пошуку, використовуючи три методи:

1. Скорочення повної корзини: існує проста процедура, яка, враховуючи поточну найкраще співпадіння, визначає, чи залишились у певній корзині елементи більш схожі на запит. Це дозволяє пошуку за методом гілок і меж усунути деякі корзини з розгляду, оскільки вони не можуть містити більш подібну часову послідовність;

2. Впорядкування корзин: можна замовити корзини приблизно в першому порядку. Тобто перші пошукові корзин містять найкращу співпадаючу послідовність з найбільшою імовірністю. Це дозволяє пошуку отримати додаткову користь від скорочення повних корзин;

3. Внутрішнє скорочення: після порівняння запиту з предметом у корзину, трикутна нерівність може бути використана, щоб можливо позбутися інших підпослідовностей у корзині.

Розглянемо кожен крок детальніше. Для побудови індексу необхідно визначитися з довжиною запитів, які, найімовірніше, зустрічаються. Припустимо, довжина визначена як *l*. Створюється шаблонна сітка довжиною *l* з секціями, розташованими рівномірно. Кожна секція сітки називається "вікном". Ця сітка розміщується в крайньому лівому краї кожного сегмента, а досліджується підрозділ S, що потрапляє в сітку, рисунок 2.26 ілюструє це позначення. Зауважимо, деякі сегменти можуть перетинати лінію сітки між двома вікнами. Ці сегменти "розбиті" на лінії так, що кожне вікно містить унікальний набір сегментів. k^{te} вікно містить набір цих сегментів, які ми позначаємо як W_k, $1<=k<=b$. Очевидно, що сексети є взаємовиключними, і їх перетином є просто вся послідовність всередині шаблону сітки.

Всі b наборів сегментів вивчаються і використовуються для отримання бітового рядка B довжиною b. Інтуїтивно, $k^{ий}$ біт - дорівнює 1, якщо набір сегментів у W_k "переважно зростає", і 0, якщо набір сегментів у W_k "переважно спадає" або постійний. Більш формально зауважимо, що на перший погляд може здатися, що більш швидким способом оцінювання вищевказаної функції було б просто вивчити різницю у значення лівої кінцевої точки самого лівого сегмента і правої кінцевої точки самого правого сегмента для визначення парності біта. У більшості випадків ця мітка дала би ту саму відповідь, однак вона взагалі не відповідає тому, що ми вимагаємо вирівнювання кінцевих точок послідовних сегментів (як у представленні базованому на сплайнах).

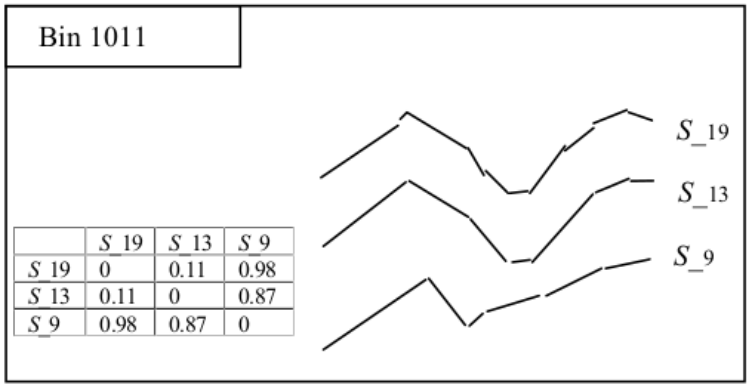
Зважаючи на те, що ми можемо створити представлення бітових строк з кожної підстроки, яка з’являється в нашій сітці шаблонів, ми використовуємо цю бітову строку щоб визначити, в який корзині розмістити копію цієї підстроки. Замість того, щоб створювати всі можливі 2^b корзин на початку алгоритму індексації, корзини створюються лише за потреби. Це робиться для запобігання зберігання мертвого простору (тобто порожніх корзин). Коли алгоритм закінчується розподілом всіх підстрок довжиною l в до корзин, може відбуватися обробка окремих корзин. Цей крок просто полягає у порівнянні кожної пари елементів та попередньому обчисленні матриці відстані для корзини. Рисунок 2.27 ілюструє структуру корзини.

Оцінку методом гілок та меж легко модифікувати для роботи з лінійними сегментами. Функція дистанції DS просто передається додатковим параметром, названим поточним best_so_far. Підсумовуючи індивідуальні внески помилок з кожного сегмента, алгоритм постійно перевіряє, чи перевищує сума досі-найкращу. Якщо так, алгоритм може відмовитися від порівняння, оскільки поточний кандидат не міг би бути ближчим, ніж найкращий поточний збіг. Зауважимо, що корисність цієї оптимізації дуже мінлива. У кращому випадку, саме перший елемент, який ви порівнюєте із запитом, може бути ідеальним збігом, і в цьому випадку ви можете відмовитися від пошуку майже відразу. У

найгіршому випадку можливо, що ніякої оптимізації методом гілок та меж отримати не вдасться, а додаткові накладні витрати трохи сповільнить пошук.

Щоб отримати значну користь від оцінювання методом гілок та меж, спочатку слід оцінити хорошу відповідність запиту. Можна використовувати представлення STB, щоб представити кандидатські підстроки в приблизному найкращому першому порядку.

Починаємо з розміщення сітки шаблону над запитом Q. Цей шаблон точно такий же, як і той, який використовується для створення початкового індексу.



	S_19	S_13	S_9
S_19	0	0.11	0.98
S_13	0.11	0	0.87
S_9	0.98	0.87	0

Рисунок 2.27 - Приклад корзини. Кожен контейнер має унікальну бітову строку як мітку. Він містить копію однієї або декількох підрядів з початкової індексованої послідовності разом з матрицею відстані, що містить відстань між кожною парою підстрок

Як і раніше, ми «розбиваємо» сегменти, які перетинають сітку для отримання b наборів сегментів. Позначимо k-ту множину цих відрізків як Q_k, 1 <= k <= b. Тепер ми можемо вивчити ці множини сегментів для отримання вектора R довжиною b. Інтуїтивно k-й елемент вектора містить реальне число, що представляє чисту кількість запиту, що піднімається в межах k-го вікна. Більш формально:

$$R_k = \sum_i QkYR_i - QkYL_i \quad . \tag{2.24}$$

Справді, якщо k[ий] набір сегментів здебільшого має тенденції до зниження, R_k буде негативним числом. Крім того, якщо в одному вікні сегменти зростають і спадають (тобто наявні вершина і долина), обидві тенденції будуть, як правило, скасовувати одна одну і R_k буде близьким до нуля.

Враховуючи вектор R, отриманий з деякого запиту Q, і бітовий рядок B, який є міткою для якоїсь корзини, ми можемо швидко обчислити нижню межу між запитом і будь-яким ймовірним збігом у цій корзині. Формула така:

$$min_{dis}(R,B)=\sum_i |(R_i)|*agree(B_i,R_i), \tag{2.25}$$

$$agree(B_i,R_i)=\begin{cases}1, if((B_i=1)\wedge(R_i>0))\vee((B_i=0)\wedge(R_i<0))\\ 0, otherwise\end{cases}. \tag{2.26}$$

Ця формула дозволяє швидко обчислити найкращий порядок, в якому представити корзини алгоритму пошуку. Почнемо з ініціалізації вільної пам’яті, яка може зберігати бітові строки довжиною b. Ми беремо список усіх міток корзин, які були використані для створення індексу, і вставляємо ці мітки на вільну пам’ять, використовуючи min_dis(R, Bi) як мітку для і-ї корзини. Порядок, в якому були переміщені ці мітки з вершини - це порядок, в якому були відібрані та оглянуті корзини. На рисунку А.4 в додатку А показано, наскільки ефективною є ця проста стратегія.

Зважаючи на те, що ми можемо замовити корзини за принципом перший-найкращий і існує така функція, яка може визначити мінімальну відстань між запитом та будь-яким із елементів, що містяться в даній корзині, ми можемо легко скоротити корзини з пошукового простору. Коли алгоритм пошуку повертає мітку чергової корзини з пам’яті, він перевіряє наступну умову: чи min_dis(R, B) досі найкращий? Якщо так, від пошуку можна відмовитися, а поточний найкращий збіг повернутий. Зауважимо, що для цієї техніки для

проведення значної обрізки важливо швидко знайти хороші співпадіння. Цей спосіб скорочення може скоротити корзину, лише вивчивши його мітку, не потрібно отримувати доступ до внутрішньої інформації корзини.

Скорочення цілої корзини дозволяє алгоритму пошуку скорочувати цілі корзини; ми також можемо використовувати трикутну нерівність, щоб (можливо) скоротити деякі об’єкти у корзині, після того як ми вивчили один або кілька об’єктів у цій корзині.

Припустимо, у нас є досі-найкращий < DS(Q,A) – DS(A,B) і DS (Q,A) - DS(A,B) < DS(Q,B). Якщо додати дві нерівності та спростити, у нас є досі-найкращий <= DS(Q,B), яка говорить нам, що послідовність B не є кращою відповіддю на наш запит Q, ніж поточна досі-найкраща версія. Однак DS(Q, A) - DS(A, B) <= DS (Q, B) - це просто перестановка визначення трикутної нерівності. Тому достатньою умовою скорочення послідовності B є:

$$best\_so\_far <= DS(Q,A) – DS(A,B). \quad (2.27)$$

Оскільки ми попередньо обчислили відстань між усіма елементами у корзині, ми можемо використовувати це правило, вивчивши кожну послідовність у корзині. На рисунку 2.28 наведено схему алгоритму внутрішнього скорочення корзини.

Після порівняння запиту Q з елементом C_j у корзині та знаходження їх на відстані D, наступним алгоритм пошуку вивчає j-й стовпець матриці корзини. Індекси рядків елементів у стовпці, менші за D - досі-найкращі, відносяться до елементів у корзині, які можна скоротити.

Наприклад, розглянемо корзину, показану на рисунку 2.27. Припустимо, у нас поточний best_so_far 0,10, і ми порівнюємо послідовність запитів Q з елементом S_13, знаходячи DS(Q, S_13) рівним 1,0. Оскільки попередньо обчислена відстань між S_13 та S_19 становить 0,11 та 1,0 - 0,10> 0,11, ми

можемо скоротити елемент S_19. Ми також можемо скоротити об’єкт S_9, оскільки 1,0 - 0,10> 0,87.

Algorithm 2.3: Burkhard and Keller algorithm

Input: Q, bin, best-so-far

let C be the set of items in the bin
while $(C \neq 0)$ **do**
    use a heuristic to pick an object C_i in C
    $update(C_i, Q, best - so - far)$
    $C = \{(C_k | DS(Q, C_i) – DS(C_i, C_k) <$
     $best - so - far) and DS(Q, C_k) is not computed)\}$
end

Рисунок 2.28 - Алгоритм (модифікований) Беркхарда та Келлера

Зауважимо, що у наведеному вище прикладі порядок, у якому ми розглядаємо об’єкти, впливає на ефективність скорочення. Якби ми спочатку подивилися на S_9, ми могли б скоротити S_13, але не S_19. Взагалі оптимальним об’єктом, який слід вивчити спочатку, є найбільш центральний об’єкт. Цей об’єкт можна швидко знайти, знайшовши стовпчик матриці відстані з мінімальною сумою.

Можна припустити, що ця методика скорочення у корзині може використовуватися сама як окрема техніка. Є дві причини, чому це не робиться:

1. Збільшиться часова та просторова складність індексу. Збільшення - лише константа, але це велика константа, щонайменше 4 або 5 порядків.

2. Ефективність цієї техніки залежить від того, наскільки швидко ви знайдете хорошу відповідність. Якщо в наборі даних дуже часто зустрічаються відповідність даному запиту, це не є проблемою. Якщо, однак, є декілька збігів, які навіть близькі до запиту в наборі даних, техніка "одна корзина" буде дуже неефективною при скороченні. На противагу нашій здатності впорядковувати корзини гарантує, що ми швидко отримаємо досить хороші відповідні співпадіння, тим самим отримуючи максимум користі від цієї техніки.

Огляд алгоритму пошуку представлений на рисунку 2.29. Параметр bin_labels - це просто список міток корзин, створених під час побудови індексу, разом із вказівниками на самі корзини.

Algorithm 2.4: Search algorithm

```
   Input: bin_labels
1  Initialize "best-so-far" to infinity get query shape Q from user
2  Let R be the discretization vector obtained from Q
3  Place bin_labels onto heap using min_dis(R,B) as heap key
4  while best-so-far > min_dis(R,B) do
5  |  Remove B from top of the heap retrieve corresponding bin
6  |  in-bin-search(Q, bin, best-so-far)
7  end
```

Рисунок 2.29 - Огляд алгоритму пошуку

2.6 Локалізація патернів у ЧП

В даній роботі використовуються запити за формою, що свідчить, що знаходження деяких патернів також дуже важлива властивість для нас. В даній роботі увага при пошуку шаблонів зосереджена на биттях інструменту. Приклад биття показаний на рисунку 2.30. Для пошуку шаблонів був використаний підхід що використовує поняття події.

Можна також розділити послідовність на більш дрібні сегменти, а потім наблизити кожен відрізок до полінома. Тоді для кожного сегмента ми повинні зберігати початкову та кінцеву позицію разом із коефіцієнтами полінома. Якщо сегменти є достатньо великими, можна досягти дуже хорошого співвідношення розмірів між початковим файлом даних та вилученими даними.

Однією з переваг цього методу є те, що вилучене наближення є безперервним і дуже просто визначити, чи дві послідовності схожі або чи містить довша послідовність більш коротку послідовність запиту.

Проблема такого підходу полягає в тому, що якщо послідовність довга, може бути дуже важко зблизити всю послідовність одним поліномом. Рішення полягає в тому, щоб розбити сигнал на кілька менших сегментів, а потім наблизити кожен відрізок до полінома.

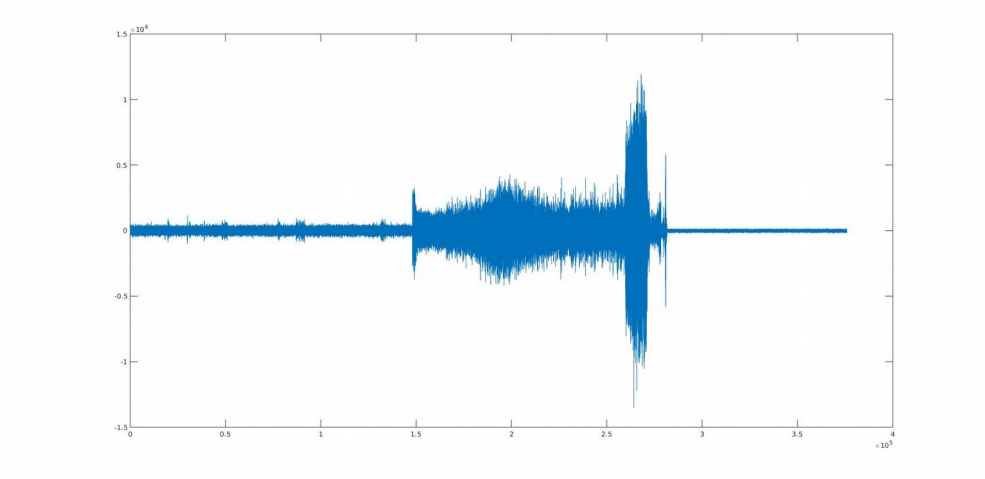


Рисунок 2.30 - Приклад биття

Проблема полягає в тому, щоб вирішити, де розірвати велику послідовність, щоб ніякі цікаві особливості не «загубилися» в розриві, і щоб отримані послідовності можна було з великою точністю наблизити.

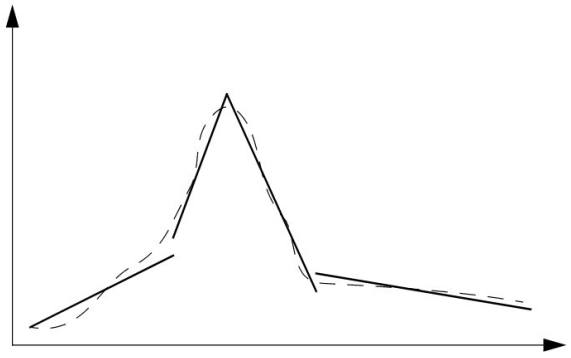


Рисунок 2.31 - Послідовність та поліноміальне наближення

На рисунку 2.31 послідовність зображена разом з простим наближенням послідовності першого ступеня. Кожен сегмент буде описуватися чотирма

параметрами: спочатку початкове та кінцеве положення відрізка, а потім значення k і m з рівняння прямої y = kx + m, оскільки в цьому прикладі послідовність наближається до многочлена першого ступеня.

2.7 Представлення запитів

Для здійснення оглядового пошуку серед наборів даних для інженерів потрібно сформувати перелік запитів, якими вони можуть користуватися. Пошук запитів, якими можуть користуватися інженери є одним з питань, на яке шукається відповідь у даній роботі.

Більшість запитів зосереджено навколо "що" (у тому числі "де", "коли" і "хто") замість "чому" або “як”. Серед усіх потенційних пошукових об’єктів, що з’являються в різних застосунках, три найпопулярніші класи інформації, які з’явилися в сучасних дослідженнях - це значення спостереження, просторові відомості та статус об’єкта. Перші два складають істотні компоненти даних спостережень. Пов’язані з властивістю датчика, інші знання (включаючи останнє) можуть бути отримані далі.

За різними умовами пошуку запити можна розділити на дві категорії. Перший - це пошук значень властивостей датчика або даних спостереження на основі конкретних ідентифікаторів. Наприклад, інженер може захотіти отримати інформацію про стан інструмента (кількість обертів, вживаний струм, тощо) на основі ідентифікатора інструмента. Другий тип - це пошук відповідних значень серії заданих обмежень у властивостях датчика або даних спостереження. Запити можуть бути введені у формі "вивести значення якості отриманих виробів для для вказаної моделі верстата" Або "показати усі процеси із заданим типом аномалій". Для пошуку другого типу запиту слід виконати два етапи. залучений. Спочатку слід знайти елементи запису, які задовольняють всі обмеження, а потім отримати конкретне значення у записах. Таким чином,

перша категорія є підмножиною другої, тому запити або використовують ідентифікатори для пошуку даних, або використовують обмеження діапазону для пошуку ідентифікаторів. Як наслідок, ми пропонуємо чотири основні моделі запитів відповідно до різних об'єктів пошуку та обмежень у запитах.

Базовий запит 1 (БЗ1): Пошук значення властивості датчика на основі ідентифікатора датчика, пов’язаного з позначеним іменем властивості.

$$[SID]\&\&[qp_1, qp_2, \dots qp_n]\ ) \Rightarrow [pv_1, pv_2, \dots pv_n]. \tag{2.28}$$

Базовий запит 2 (БЗ2): Пошук даних спостережень на основі ідентифікатора датчика або ідентифікатора спостереження.

$$[SID]|||[OID_1, OID_2 \dots OID_n]\ ) \Rightarrow [o_1, o_2, \dots o_n]. \tag{2.29}$$

Базовий запит 3 (БЗ3): Пошук датчиків, які задовольняють обмеженням властивостей у діапазонному запиті.

$$[qp_1 : (qpv_{1l}, qpv_{1h}), qp_2 : (qpv_{2l}, qpv_{2h}), \dots, qp_n : (qpv_{nl}, qpv_{nh})]\ ) \Rightarrow [SID_1; SID_2, \dots, SID_m]. \tag{2.30}$$

Базовий запит 4 (БЗ4): Пошук спостережень, які відповідають запитуваному інтервалу за місцем розташування, часом або значенням даних (i-й інтервал представлений як qol_i, qot_i, qov_i).

$$[qol_1, qol_2, \dots, qol_m] \,||\, [qot_1, qot_2, \dots, qot_n] \,||\, [qov_1; qov_2, \dots, qov_p]\ ) \Rightarrow [OID_1; OID_2, \dots, OID_q]. \tag{2.31}$$

3 РЕАЛІЗАЦІЯ МЕТОДІВ ІНДЕКСАЦІЇ ТА ПОШУКУ У БАГАТОВИМІРНИХ ЧАСОВИХ ПОСЛІДОВНОСТЯХ

Даний розділ містить докладний опис даних, технологій, особливостей реалізації алгоритмів і експериментів в розробленому рішенні.

Розв'язок написано переважно мовою Python, а критичні для швидкості виконання функції реалізовані мовою C. Також було використано бібліотеки Numpy, Pandas, Sklearn, Scipy і Matplotlib. Розробка і розрахунки були проведені в середовищі дистрибутиву Debin, базованому на ядрі Linux. Обчислення відбувалися на спеціалізованій платформі лабораторії L3S та Google Cloud Platform.

З міркувань збереження балансу між витратами часу на обчислення, якості результатів було прийнято рішення зосередити зусилля на підмножиною даних доступних від замовників. По-перше, для експериментів було обрано дані лише десяти експериментів. По-друге, для прикладу багатовимірних даних достатньо даних знятих з одного з тривимірних сенсорів прискорення. Його було обрано, тому що частота вибірки на ньому рівна 1000 вимірів на секунду. В середньому за один експеримент в залежності від тривалості експерименту такий сенсор створює близько 300000 вимірів. Оскільки така послідовність містить дуже багато деталей про характер часового процесу, це робить її найкращим кандидатом для початкових даних. Зайві деталі по ходу обчислень можуть бути опущені або трансформовані. Три виміри, які були згадані, це x, y і z - значення прискорення переміщення інструменту. Попри те, що для обчислень експериментів було обрано спеціальні платформи для машинного навчання, для того, щоб здійснити обчислення за розумний проміжок часу, було також прийнято рішення скоротити вихідну кількість даних до 100000 на кожну координату. Підсумовуючи вищесказане, простими обчисленнями отримуємо кількість використаних вимірів: 1000000 значень для кожної з осей або 3000000 значень загалом.

3.1 Реалізоване рішення

Реалізоване рішення є модифікацією метода індексації STB. Необхідність модифікацій обумовила природа використовуваних даних.

Вся практична частина поділяється на дві великих етапи: створення індексу і пошук. Етап створення індексу складається з таких послідовних кроків:

1. Розбиття вихідних послідовностей на підпослідовностей фіксованої довжини;

2. Визначення корзини, до якої належить підпослідовність.

Для першого пункту варто зазначити довжину підпослідовності обрану для нашого рішення рівну 2000 елементів. Така довжина була обрана з огляду на баланс точності і швидкості обчислень. Дана інформація зберігається у вигляді словника, в якому ключем є унікальний ідентифікатор підпослідовності, а значеннями є пара координат даної підпослідовності у часі.

Другий пункт потребує більш детального коментаря. За оригінальним алгоритмом для обчислення бінарної мітки корзини обирається сегмент підпослідовності, довжиною меншою за довжину підпослідовності. Але через певну причину, в даній роботі розмір сегмента для обчислення мітки дорівнює довжині підпослідовності. Ця причина - мітка в нашому рішенні також виступає як дуже наближений ідентифікатор форми, і під однією міткою мають бути зібрані підпослідовності, що схожі між собою на всій довжині, а не тільки на початку. Сама мітка представляє собою сітку, що складається з n вікон. В кожному з вікон окремо обчислюється сума спадаючих і зростаючих сегментів. Це обумовлено тим, що в оригінальній послідовності, рисунок 3.2, горизонтальних і вертикальних відрізків немає, тому кожен із них робить вклад. Коли суми обчислені, визначається більше абсолютне значення. Якщо додатня сума більша за абсолютом, то біт рівний 1, якщо навпаки - рівний 0. Останнє питання, яке потрібно прояснити, це кількість біт, що використовуються для

мітки. Детально це питання було освітлено у оригінальній роботі, для наших експериментів було обрано значення рівне 5, що створить 32 унікальні ідентифікатори, або іншими словами - розіб'є множину всіх підпослідовностей на 32 класи. Це достатньо для того, щоб розмістити 1500 підпослідовностей максимально ефективно.

В результаті ініціалізації було отримано по 32 корзини для кожної вісі. Приклад реально утвореної корзини схематично зображено на рисунку 3.1:

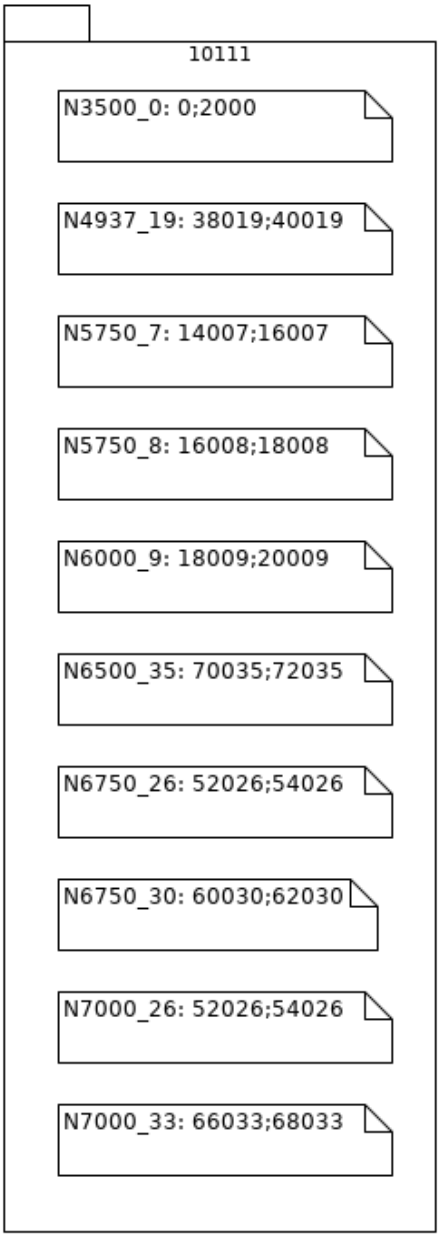


Рисунок 3.1 - Корзина після ініціації

Наступний крок - розрахунок відстаней між підпослідовностями. Для того, аби перейти до цих обчислень, потрібно зробити відступлення та розглянути часові послідовності візуально.

Оригінальна часова послідовність виглядає наступним чином, рисунок 3.2:

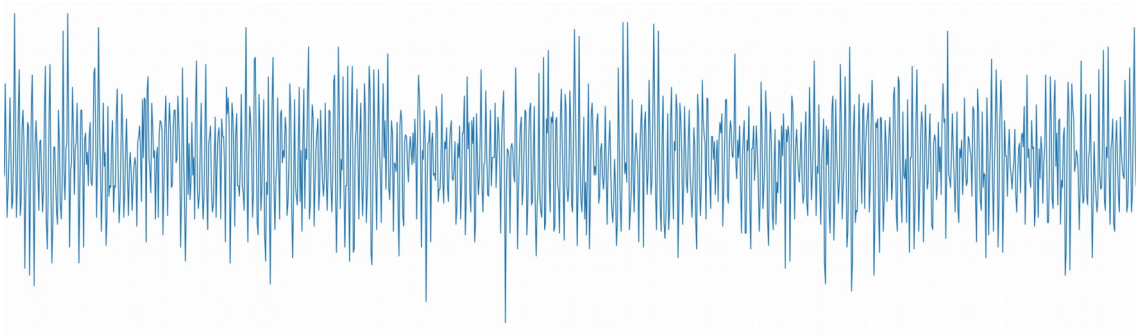


Рисунок 3.2 - Характер часової послідовності на прикладі фрагменту N3500_0

На рисунку 3.2 гарно видно, що лінія послідовності дуже ламана. Подібна властивість перешкоджає індексуванню. Головною задачею є вилучення форми послідовності. Щоб отримати її, було використано обвідну сигналу. Рисунок 3.3 показує, як вона була побудована.

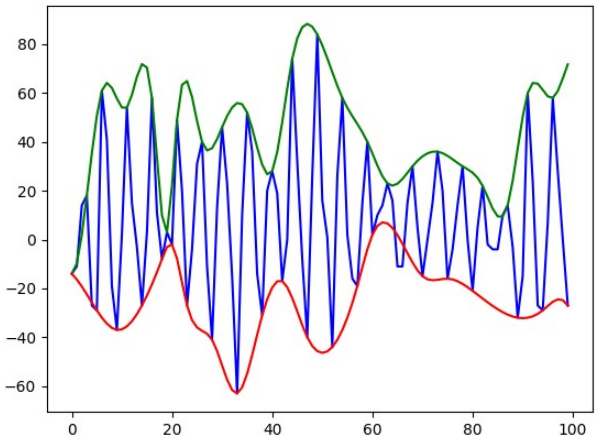


Рисунок 3.3 - Обвідна фрагменту підпослідовності N3500_0

Обвідна складається з двох кривих, верхньої і нижньої обмежувальних границь. Для того, аби передати форму однією кривою, для кожного моменту часу ми рахуємо середнє арифметичне значення двох кривих. Третя крива містить інформацію одразу про коливання обох кривих і одночасно є значно згладженою у порівнянні з вихідною послідовністю, рисунок 3.4.

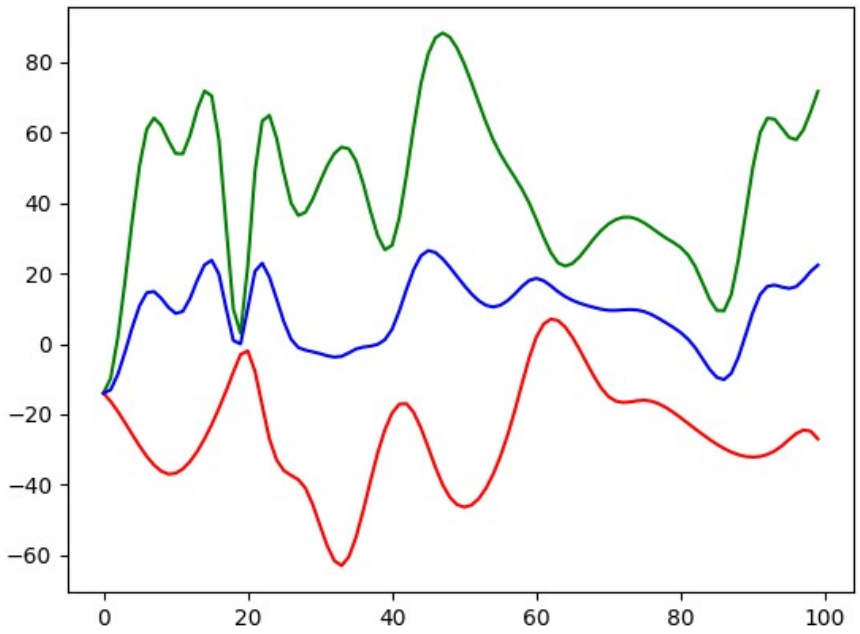


Рисунок 3.4 - Результат згладження

Підготовлену послідовність можна перетворити на текстову строку методом SDA(5), як це було описано у підрозділі 2.3.1 Після перетворення послідовність являє собою послідовність символів, а не чисел, тобто текст. Для порівняння текстових послідовностей можна застосувати відомий алгоритм Левенштайна. Результат алгоритму при порівнянні двох строк - ціле невід’ємне число, яке позначає кількість перетворень, які потрібно внести в першу строку, щоб отримати другу строку. Тобто, ми можемо стверджувати, що чим менше дане число, тим більш схожими за формою дві послідовності. Описаний метод має дві переваги: він дозволяє порівнювати послідовності різної довжини і він є

метрикою, бо гарантує виконання трикутної нерівності. Дана властивість грає важливу роль у подальшому і буде обов’язково описана далі.

В кожній корзині знаходяться набори найбільш подібних підпослідовностей. Між кожною парою підпослідовностей розраховується дистанція Левенштайна. Для оптимізації обчислень, оскільки метрика симетрична, значення розраховується тільки для послідовностей над головною діагоналлю. Приклад розрахованої матриці на рисунку 3.5:

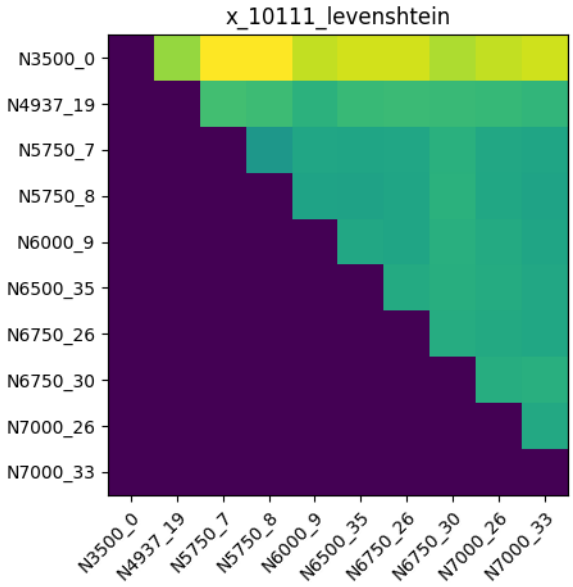


Рисунок 3.5 - Дистанції між підпослідовностями

Після того, як два найголовніших кроки по створенню індексу завершені, можна приготувати запити до пошуку. Для експериментів було обрано дві підпослідовності, наведені на рисунках 3.6 - 3.11.

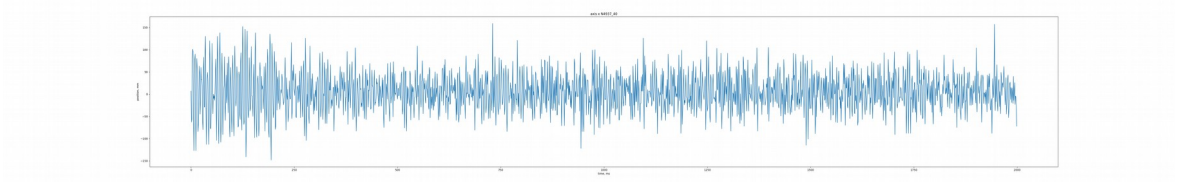


Рисунок 3.6 - Запит 1, послідовність N4937_40, вісь x

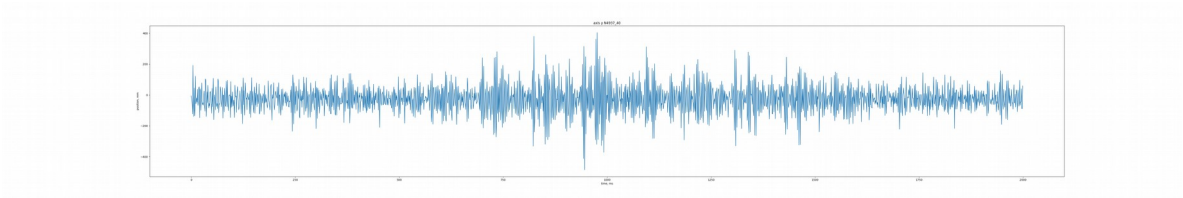


Рисунок 3.7 - Запит 1, послідовність N4937_40, вісь y

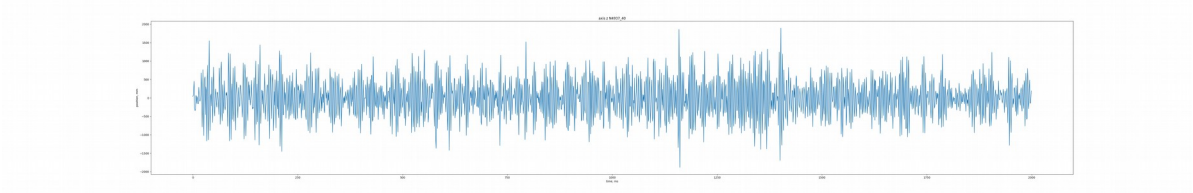


Рисунок 3.8 - Запит 1, послідовність N4937_40, вісь z

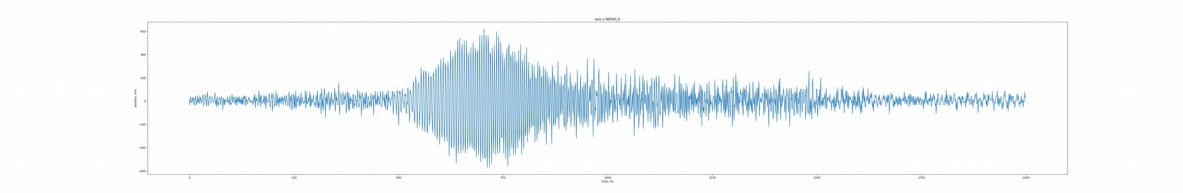


Рисунок 3.9 - Запит 2, послідовність N6000_0, вісь x

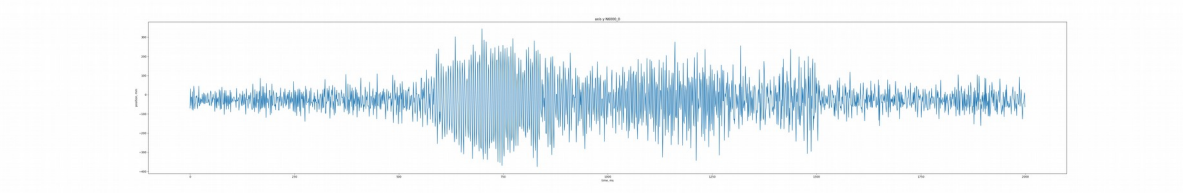


Рисунок 3.10 - Запит 2, послідовність N6000_0, вісь y

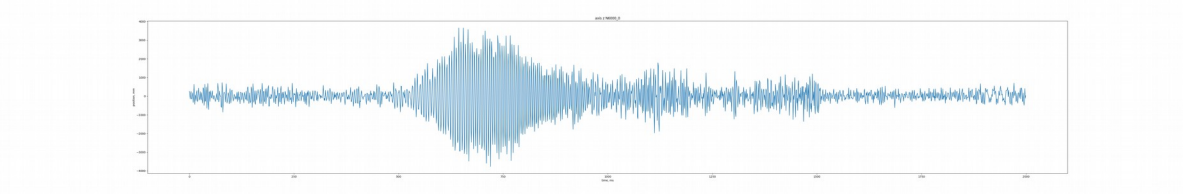


Рисунок 3.11 - Запит 2, послідовність N6000_0, вісь z

Для запитів було обрано два сегменти, один з яких представляє собою приклад без аномалій, а другий - приклад з аномаліями.

На етапі пошуку відбувається два кроки:

- впорядкування корзин у порядку від корзини з послідовностями, що мають найвищу подібність за формою до запиту, до найменшої подібності;

- пошук всередині корзини за допомогою тих самих перетворень, які були використані при розрахунках дистанцій.

На першому кроці, для того щоб впорядкувати корзини, необхідно знайти кількісну міру подібності всієї корзини загалом. Для цього використовуються мітки корзин, а також деяка нова міра подібності. Спершу на запит накладається та сама сітка, що була використана для розрахунку мітки корзини. Цього разу, єдина відміна полягає у тому, що значення утворені у вікнах сітки не перетворюються на значення бітів, а зберігаються у вектор, який для нашого випадку, має розмірність [5,1]. Для отримання оцінки корзини, здійснюємо множення двох векторів, в результаті чого отримуємо скалярне ціле значення. За даною оцінкою впорядковуємо корзини і переходимо до другого кроку.

На другому кроці, перед інспекцією корзини, ми проводимо над запитом ті самі перетворення що й над індексованими підпослідовностями: згладження і конвертацію в текстову строку. Далі переходимо до порівнянь із підпослідовностями з матриці відстаней корзини. Критерієм прийняття підпослідовності є те, що різниця між запитом та підпослідовністю менша за деяку порогову величину. Нехай вона називається best_so_far. Перше порівняння відбувається з підстрокою, що знаходиться першою відповідній строці матриці. Значення отримане при порівнянні алгоритмом Левенштайна стає опорним значенням. Раніше було згадано, що дистанція Левенштайна гарантує правило трикутника, яке стає у нагоді саме зараз. Справа в тому, що різниця відстані від запиту до опорної підпослідовності та відстані між опорною послідовністю та будь-якою іншою підпослідовностю у поточному рядку матриці менша за порогове значення, це означає, що інша підпослідовність подібна до запиту. Оскільки інші відстані вже розраховані, то необхідності обчислювати вдруге немає, а пошук зводиться до порівняння

значень під час ітерації по матриці відстаней, нагадуємо, по частині над головною діагоналлю.

3.2 Опис експерименту

Експеримент полягає у пошуку обраних сегментів з різним пороговим значенням, яке було згадане як best_so_far. Ця величина логічно розширює міру неспівпадіння кандидатської послідовності коли ми її збільшуємо, і навпаки, звужує - коли зменшуємо. Для експериментів, було обрано 2 діапазони варіювання цієї величини: спершу на інтервалі [100,800], а потім на діапазоні [10,90]. Результати розрахунків відображені на спільних графіках, рисунки 3.12 - 3.14.

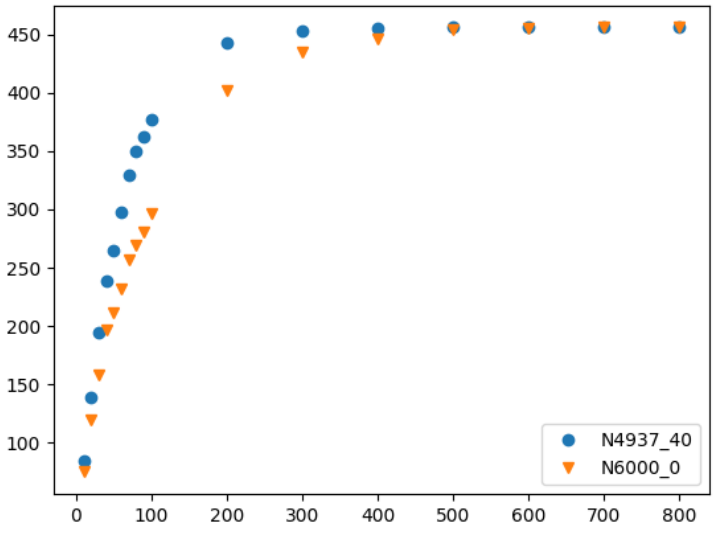


Рисунок 3.12 - Графік залежності кількості знайдених підпослідовностей від порогової величини на осі X

З графіку 3.12 видно, що найбільше зростання кількості запитів відбувається на інтервалі [0,200], після якого приріст мінімальний. Схожий

характер зростання відбувається і в інших осях. Ці графіки показують інформацію індивідуально для кожної вісі зовсім не даючи розуміння які з отриманих підпослідовностей у кожному вимірі можуть належати до імовірних результатів. Пояснимо на конкретному уявному прикладі. Якщо для запиту за віссю X було знайдено підпослідовність N6250_21, N6250_12 і N6250_41, за віссю Y було знайдено підпослідовності N6500_7, N6250_21 і N5750_21, і відповідно за віссю Z - N3500_1, N7000_2 і N6250_21, то лише N6250_21 може вважатися повноцінним результатом пошуку, оскільки тільки ця підпослідовність вказана у результатах для кожної осі.

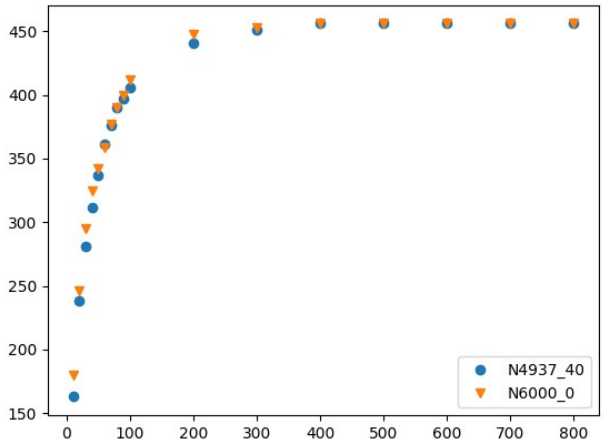


Рисунок 3.13 - Графік залежності кількості знайдених підпослідовностей від порогової величини на осі Y

З цих графіків можна лише зробити висновок, що тенденція зростання кількості результатів пошуку була підтверджена результатами експериментів. Але, для нас надзвичайно важливо дізнатися, як саме веде себе величина кількості пошукових результатів саме для багатовимірних результатів. Тому для наступних графіків, була зроблена інша проекція отриманих даних. Спочатку ми згрупували дані за пороговим значенням і за запитом, а потім взяли перетин всіх ідентифікаторів підпослідовностей в результатах по осям. Рисунки 3.15 і 3.16 показують ці результати на графіках.

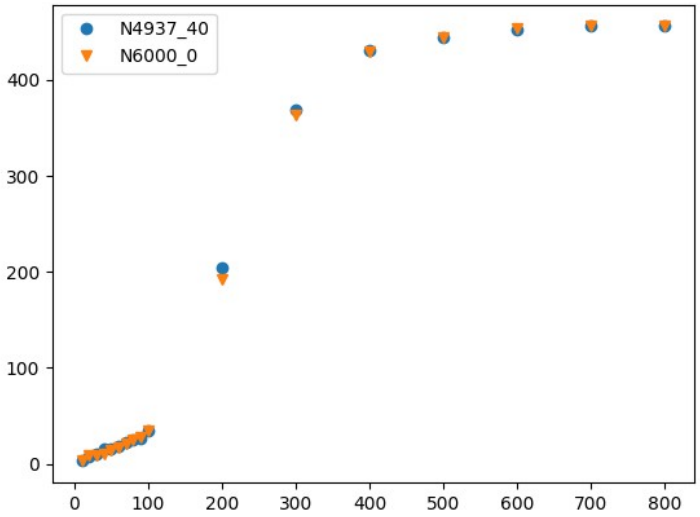


Рисунок 3.14 - Графік залежності кількості знайдених підпослідовностей від порогової величини на осі Z

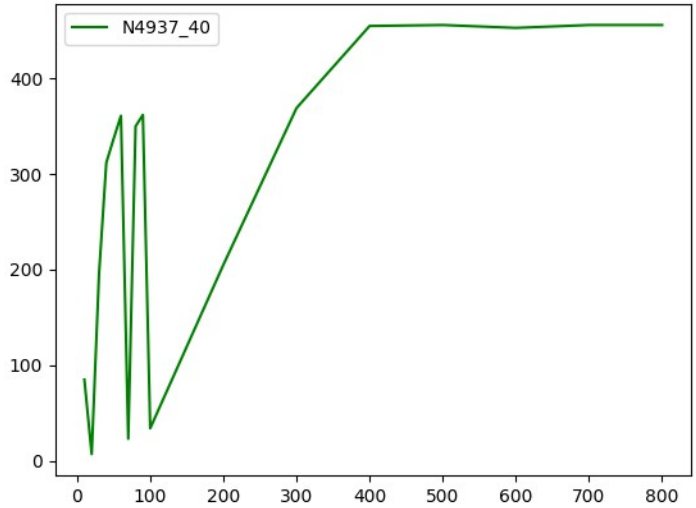


Рисунок 3.15 - Графік залежності кількості результатів при групуванні

Ці два рисунки знову демонструють тенденцію, але також показують, що пошукові результати виконані мають сенс. Якщо з отриманих для кожної вісі результатів підпослідовностей можна було би отримати невелику кількість часових сегментів, то такі результати не можна вважати надійними.

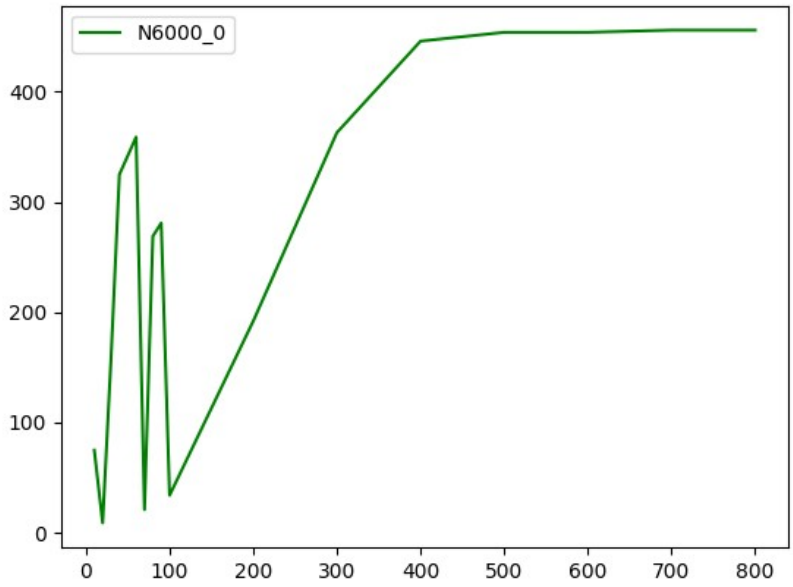


Рисунок 3.16 - Графік залежності кількості результатів при групуванні

Для нашого випадку відношення реальних результатів до загальної кількості результатів знаходиться в околиці 85%, що є де-факто взірцем в обробці даних.

Наостанок варто розглянути реальні результати. Для цього візьмемо наприклад три зі знайдених підпослідовності, рисунки 3.17-3.19.

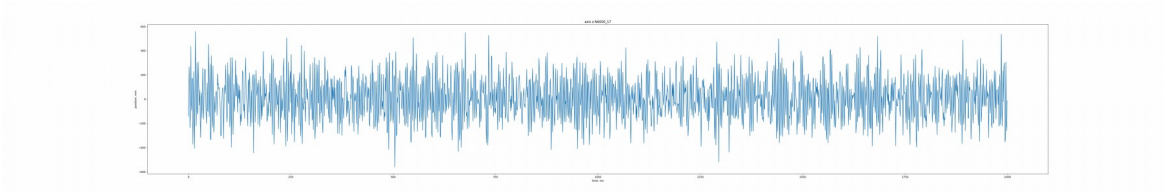


Рисунок 3.17 - Підпослідовність N6000_17

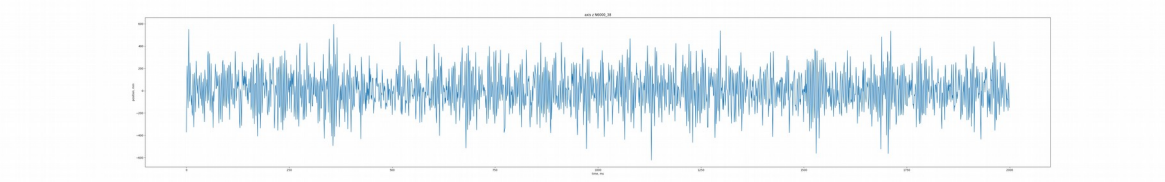


Рисунок 3.18 - Підпослідовність N6000_38

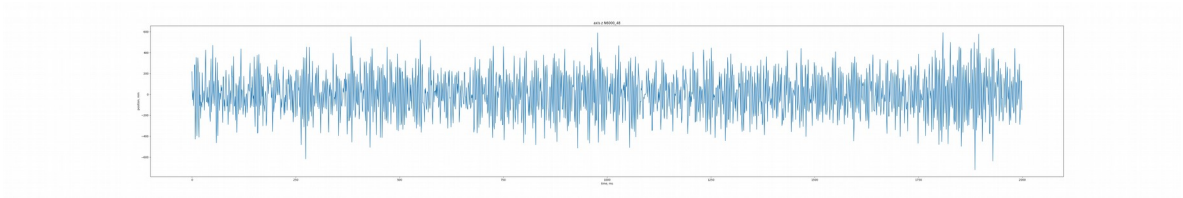


Рисунок 3.19 - Підпослідовність N6000_48

Ці результати розчаровують, бо форма описана на оригінальній послідовності надто не схожа на отримані результати. Не виключаю, що з математичної точки зору ці послідовності дійсно схожі між собою, але був очікуваний дещо інший результат.

Для того, аби переконатися в цьому, скористаємося мірою подібності DTW створеної для порівняння часових послідовностей. Візьмемо для порівняння N6000_0 і N6000_17, рисунок 3.20.

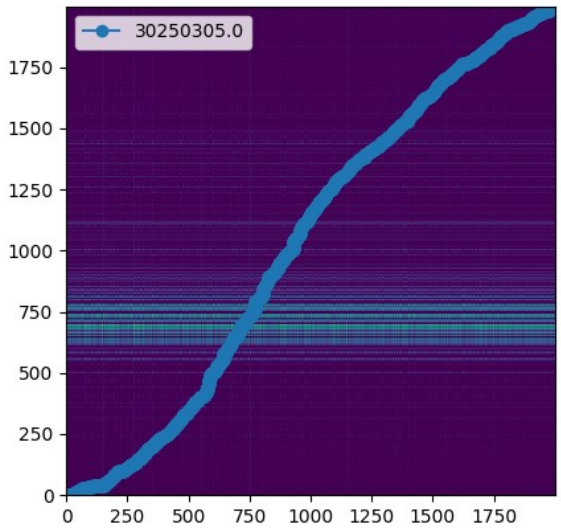


Рисунок 3.20 - Матриця викривлення для N6000_0 і N6000_17

Відстань, яка була порахована алгоритмом DTW між підпослідовностями N6000_0 і N6000_17 рівна 30250305, для N6000_0 і N4937_0 рівна 8748019. Вона є значно меншою, але була порахована для “сирих” даних.

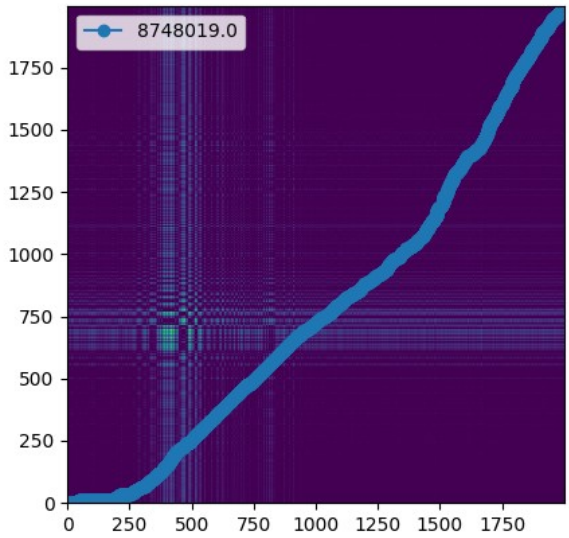


Рисунок 3.21 - Матриця викривлення для N6000_0 і N4937_0

Дистанцію Левенштайна було пораховано для перетворених підпослідовностей. Тому для чистоти експериментів було пораховано відстані для оброблених підпослідовностей.

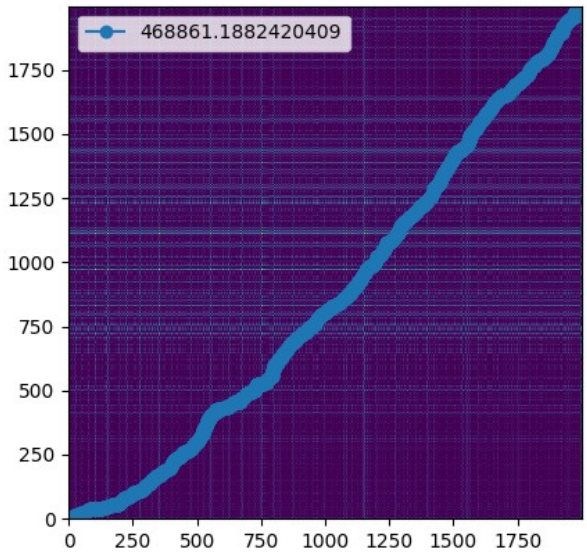


Рисунок 3.20 - Матриця викривлення для N6000_0 і N6000_17 для оброблених даних

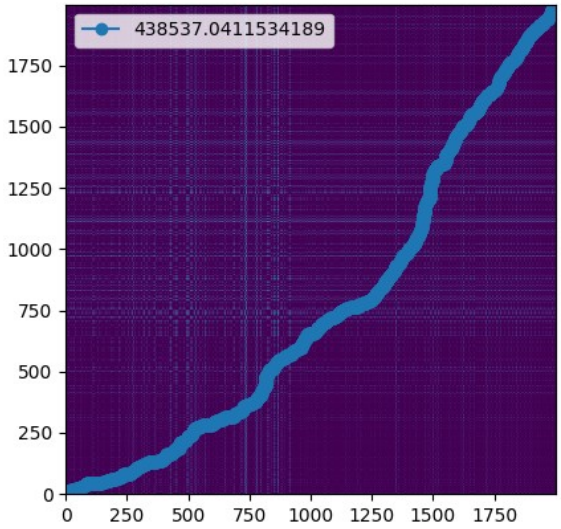


Рисунок 3.23 - Матриця викривлення для N6000_0 і N4937_0 для оброблених даних

Наостанок потрібно надати деякі відомості про загальний організації стан експерименту. Оскільки в даній роботі було зосереджено увагу на пошуку оптимальних алгоритмів, структур даних, то для втілення рішення було використано надзвичайно просту архітектуру. Клас, що представляє індекс, і клас, що представляє корзину. Логіка розрахунків розподілена між класом індексу і винесена в окремий клас, який в даній роботі не приведений. Діаграми класів приведені на рисунках: класу Bin - 3.24, класу STBIndex- 3.25, загальна структура експерименту схематично зображена - 3.26.

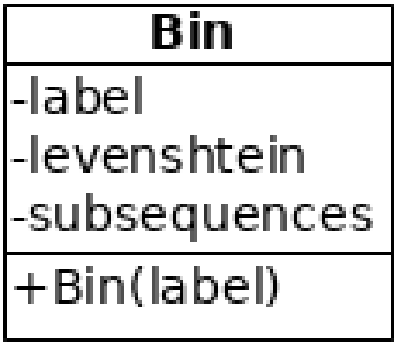


Рисунок 3.24 - Діаграма класу Bin

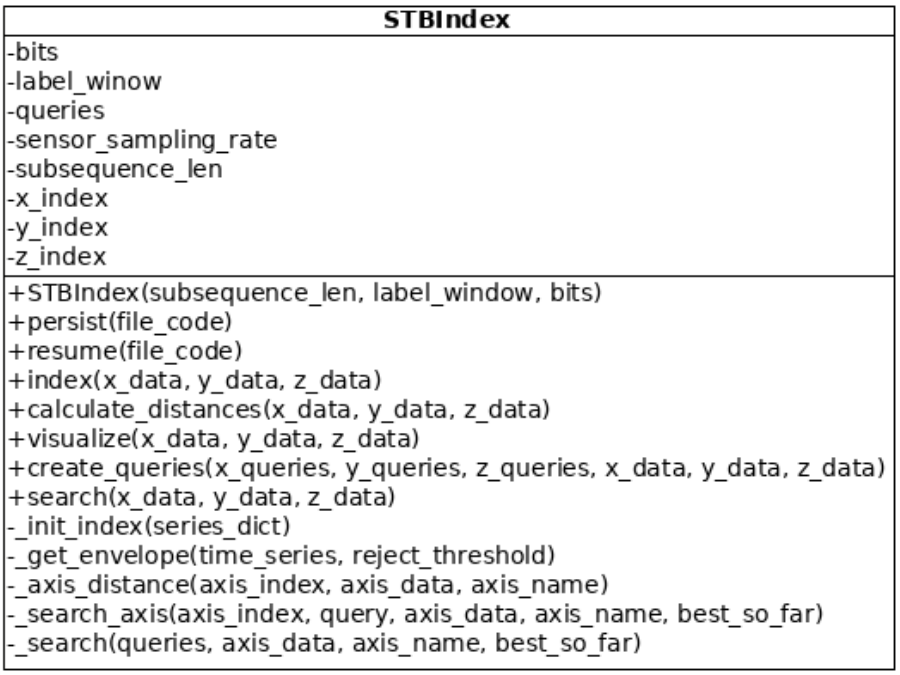


Рисунок 3.25 - Діаграма класу STBIndex

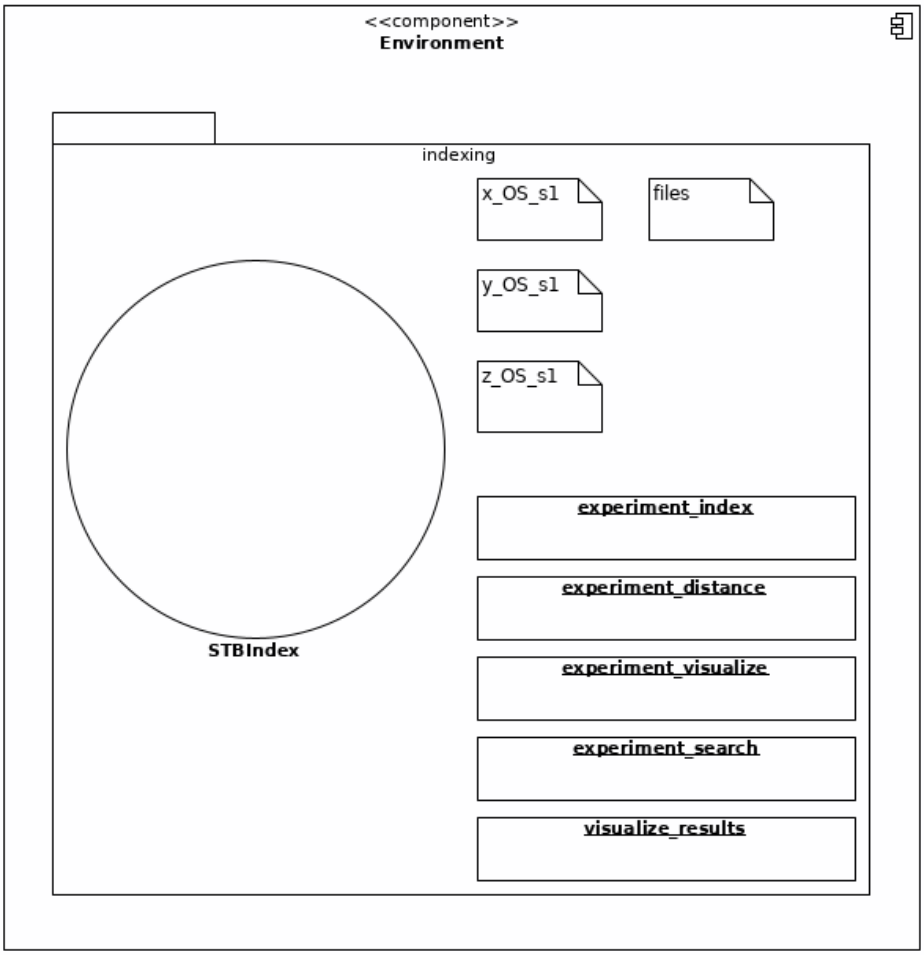


Рисунок 3.26 - Схема експериментів

4 ПЕРСПЕКТИВИ ДОСЛІДЖЕННЯ НЕСТРУКТУРОВАНИХ НАБОРІВ ДАНИХ

В даній роботі не було розглянуто питання 5 з огляду на великий об’єм роботи необхідний для дослідження перших п’яти питань. Загальна ідея, яка може бути використана для ранжування - чим більша довжина у часі і ближчий характер співпадіння підпослідовності, тим вище у списку результатів буде знаходитись ЧП, що її містить.

В даній роботі було зосереджено увагу на створенні інструментів для обробки ЧП, без додатку, який би могли використовувати цільові користувачі. Попри цю обставину, робота над створенням додатку на момент написання цієї роботи вже була розпочата. Додаток проекту буде базуватися на існуючому проекті під назвою Workive що був створений із подібними цілями та завданнями. Проект Workive - це проект, орієнтований на забезпечення інтелектуального доступу для соціологів до цифрового архіву якісних даних у формі інтерв'ю працівників, що спостерігає за ситуацією в німецькій автомобільній та суднобудівній промисловості, зібраний за останні 50 років [4].

4.1 Архітектура видобутку даних

На рисунку 4.1 представлено приклад архітектури пошуку запропоновану у [11]. Для реалізації сервісу пошуку інформації базовану на індустріальних даних, інтерфейс запитів повинен мати два компоненти, підключені через потік даних, тобто збирання та зберігання інформації та пошук інформації, які спрямовані на збір. інформацію та отримання даних із сховища, відповідно. Звичайний запит є складним і вимагає повного відповіді на багато інформації. Для точного розуміння наміру пошуку та ефективного полегшення пошуку, запит повинен бути розбитий на кілька підзапитів, і кожен підзапит може

відповідати відповідному індексу в індексуючому компоненті для пошуку. Інформація зондування обладнання збирається зі звітів експериментів різних форматів. Вона поділяється на дані датчиків та дані, пов'язані з датчиком. Визначення цих двох типів даних наведено нижче. Для зручного подальшого дослідження вся зібрана інформація зберігається у сховищі індивідуально на основі запропонованих моделей представлення даних.

Визначення 4.1 (дані, про датчик): Додаткова інформація про датчики, яка допомагає записувати контекстну інформацію спостережуваних даних. У цій роботі вони також називаються властивостями датчиків.

Приклад: стан датчика, якість вимірювання, термін служби акумулятора тощо.

Визначення 4.2 (дані з датчика): значення, що спостерігається у оточуючому середовищі, генерується датчиком, також далі називається даними спостереження.

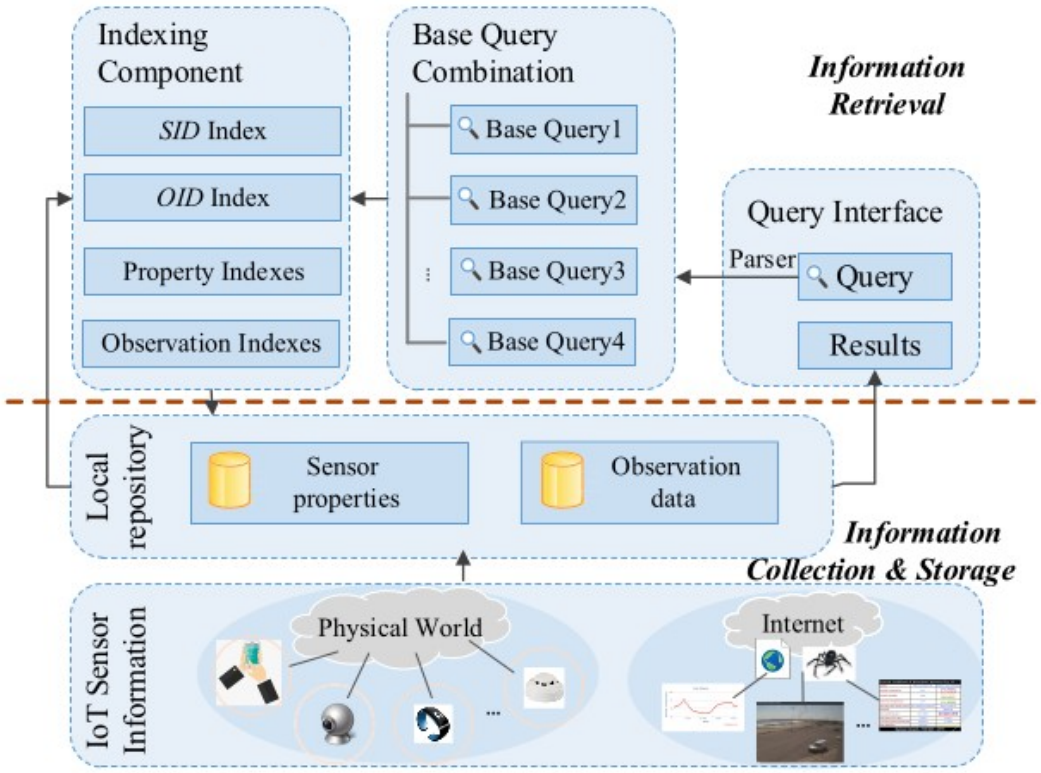


Рисунок 4.1 - Архітектура пошуку інформації для індустріальних даних

На основі різних обмежень пошуку, таких як обмеження на ідентифікатор датчика або діапазон простору, ми витягуємо чотири основні моделі запитів з різних запитів до індустріальних даних, за допомогою яких можна скласти всі можливі запити. Для швидкого пошуку кожної моделі запитів ми відповідно розробляємо чотири унікальні індекси. SID і OID являють собою ідентифікатор датчика і елемент даних спостереження, відповідно, а показники властивості та спостереження можуть підтримувати пошук властивостей датчика та даних спостереження відповідно.

ВИСНОВКИ

У першу чергу при обробці літератури основну увагу було приділено пошуку надійних мір подібності. Виявилося, що вже з кінця минулого століття існують алгоритми, які змогли стати класичними та породити сімейство алгоритмів. В даній роботі було зроблено спробу використати ці алгоритми з деякими модифікаціями, що стосуються використаного представлення даних. Одночасно з цим питання представлення даних також мало бути вирішено одночасно, оскільки має безпосередній вплив на всі алгоритми в подальшому. Серед низки різних методів опису форми було обрано передачу форми сигналу методом опису форми алфавітом. Головна причина полягала в тому, що характер вихідної часової послідовності є надлишково деталізований і для пошуку в певною мірою нечіткості обмежував би кількість співпадінь. Передача текстом поєднує звужує діапазон значень і полегшує навантаження на процесор при обчисленнях. На роль міри подібності була обрана дистанція Левенштайна.

У другу чергу було обрано схему індексування. Вона в великій мірі залежить від обраної міри подібності. Існує два основних підходи до пришвидшення пошуку за допомогою індексу. У першому варіанті обирається ефективний спосіб зменшити кількість даних і використання дуже ефективних алгоритмів пошуку реальному часі. Другий підхід на початку має фазу важких обчислень, для того, щоб створити деякі проміжні універсальні результати обробки даних і зберегти їх у структуру даних з деяким відношенням порядку визначеному на ньому, яка дозволяє зменшити кількість порівнянь. У даній роботі було описано обидва підходи відповідно. Головним фактором вибору на користь другої схеми є потреба у гарній архітектурі розподілених обчислень, яка дозволить використати повний потенціал підходу, створення якої виходить за межі даної роботи. В роботі над другим варіантом було модифіковано структуру даних для збереження результатів і принцип визначення параметрів індексування. Зокрема, при індексуванні для визначення мітки підпослідовності

було використано не її сегмент, а всю, оскільки ми зосереджені на формі послідовностей. Схема індексування також відповідає за алгоритм пошуку. Він в значній мірі базується на створеній структурі даних. В нашій роботі було використано алгоритм пошуку без жодних змін порівняно з оригіналом.

Після вибору трьох ключових складових не менш важливим питанням стало вибір форми запиту. Запит в експериментальних умовах здійснюється як надання даних був використаний прямолінійний і полягав в тому, що пошуковому алгоритму повідомлювалася шукана послідовність. Описане рішення стосується в більшій степені реалізації додатку, яким будуть користуватися кінцеві користувачі. Оскільки багатовимірні часові запити не мають гарної реалізації, тому було вирішено сконцентруватися на одновимірних запитах в які був розкладений багатовимірний запит, пошук як для одновимірних випадків і наступне об’єднання результатів.

Після вибору та обґрунтування всіх складових рішення їх було втілено мовою програмування Python.

На етапі експериментів обчислення зайняли близько кілька днів, для того щоб переконатися у правильності створеного індексу. Після завершення пошукових випробувань, ми розпочали аналіз отриманих результатів.

Коли експериментальні результати були отримані і візуалізовані, то стало ясно, що отриманий результат не відповідає очікуванню. Попри гарні цифри, що підтверджують що потрібні вимоги були виконані, результати не відповідають очікуванням. Отримані результати були повторно порівняні з очікуваними результатами аналітично за допомогою алгоритму DTW. Він підтвердив, що попри математичну схожість отриманих результатів їх форма не відповідала формі запиту. Повторна перевірка породила гіпотезу, що усереднення значень часових послідовностей було зроблено занадто сильно, в результаті чого суттєві властивості часових послідовностей були втрачені. Це призвело до викривлення оцінок подібності послідовностей між собою на етапі

індексування, а запит до таких даних не може отримати релевантної "відповіді". В подальшій роботі це питання буде вирішено в першу чергу.

Загалом мною виконана робота оцінюється позитивно. Протягом її виконання мною було сформовано погляд на перспективну предметну область, яка буде розвиватися з приходом нових тенденцій у майбутньому. На момент написання цього розділу перелік питань, яким має бути приділена увага вже був сформульований. До нового етапу підхід буле враховувати аспекти, що були неочевидними на початку роботи над проектом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Baker, M. (2017). U.S. Patent No. 9,578,046. Washington, DC: U.S. Patent and Trademark Office.

2. Cervelli, D., Skiff, D., Tobin, D., & Cai, A. (2019). U.S. Patent Application No. 16/443,626.

3. Baum, M. J., Carasso, D., Das, R. K., Greene, R., Hall, B., Mealy, N., ... & Swan, E. M. (2012). U.S. Patent No. 8,112,425. Washington, DC: U.S. Patent and Trademark Office.

4. Baethge-Kinsky, V., & Zerr, S. (2015). Die Erschließung von Primärmaterial qualitativer Studien für die Sekundäranalyse als Herausforderung für Sozialwissenschaften und Informatik. Datenbank-Spektrum, 15(1), 33-39.

5. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., & Keogh, E. (2006). Indexing multidimensional time-series. The VLDB Journal, 15(1), 1-20.

6. André-Jönsson, H. (2002). Indexing Strategies for Time Series Data. Department of Computer and Information Science, Linköpings universitet.

7. Folgado, D., Barandas, M., Matias, R., Martins, R., Carvalho, M., & Gamboa, H. (2018). Time alignment measurement for time series. Pattern Recognition, 81, 268-279.

8. Laftchiev, E., Liu, Y. (2018) Finding Multidimensional Patterns in Multidimensional Time Series SIGKDD Workshop on Mining and Learning From Time Series

9. Senin, P. (2008). Dynamic time warping algorithm review. Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, 855(1-23), 40.

10. Ozsoyoglu, T. B. N. Y. M. (1997). Matching and indexing sequences of different lengths. In Proceedings of the Sixth International Conference on Information and Knowledge Management: CIKM'97, November 10-14, 1997, Las Vegas, Nevada (Vol. 6, p. 128). Association for Computing Machinery (ACM).

11. Liu, M., Li, D., Chen, Q., Zhou, J., Meng, K., & Zhang, S. (2018). Sensor Information Retrieval From Internet of Things: Representation and Indexing. IEEE Access, 6, 36509-36521.

12. Kahveci, T., & Singh, A. (2001, April). Variable length queries for time series data. In Proceedings 17th International Conference on Data Engineering (pp. 273-282). IEEE.

13. Keogh, E. J., & Pazzani, M. J. (1999, July). An indexing scheme for fast similarity search in large time series databases. In Proceedings. Eleventh International Conference on Scientific and Statistical Database Management (pp. 56-67). IEEE.

14. Salvador, S., & Chan, P. (2004). FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. 3rd Wkshp. on Mining Temporal and Sequential Data. *ACM KDD '04*.

15. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

16. Keogh, E. J., & Pazzani, M. J. (2000, August). Scaling up dynamic time warping for datamining applications. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 285-289). ACM.

17. Keogh, E. J., & Pazzani, M. J. (1999, September). Scaling up dynamic time warping to massive datasets. In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 1-11). Springer, Berlin, Heidelberg.

18. Vlachos, M., Kollios, G., & Gunopulos, D. (2002, February). Discovering similar multidimensional trajectories. In Proceedings 18th international conference on data engineering (pp. 673-684). IEEE.

19. Hetland, M. L. (2004). A survey of recent methods for efficient retrieval of similar time sequences. In Data mining in time series databases (pp. 23-42).

20. Keogh, E., Palpanas, T., Zordan, V. B., Gunopulos, D., & Cardle, M. (2004, August). Indexing large human-motion databases. In Proceedings of the

Thirtieth international conference on Very large data bases-Volume 30 (pp. 780-791). VLDB Endowment.

21. Gunopulos, D., & Das, G. (2000, August). Time series similarity measures (tutorial PM-2). In Tutorial notes of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 243-307). ACM.

22. Assent, I., Wichterich, M., Krieger, R., Kremer, H., & Seidl, T. (2009). Anticipatory DTW for efficient similarity search in time series databases. Proceedings of the VLDB Endowment, 2(1), 826-837.

23. Truong, C. D., & Anh, D. T. (2015). A fast method for motif discovery in large time series database under dynamic time warping. In Knowledge and Systems Engineering (pp. 155-167). Springer, Cham.

24. Kremer, H., Günnemann, S., Ivanescu, A. M., Assent, I., & Seidl, T. (2011, July). Efficient processing of multiple DTW queries in time series databases. In International Conference on Scientific and Statistical Database Management (pp. 150-167). Springer, Berlin, Heidelberg.

25. Niennattrakul, V., Wanichsan, D., & Ratanamahatana, C. A. (2009, April). Accurate subsequence matching on data stream under time warping distance. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 156-167). Springer, Berlin, Heidelberg.

26. Petitjean, F., Forestier, G., Webb, G. I., Nicholson, A. E., Chen, Y., & Keogh, E. (2014, December). Dynamic time warping averaging of time series allows faster and more accurate classification. In 2014 IEEE international conference on data mining (pp. 470-479). IEEE.

27. Petitjean, F., Forestier, G., Webb, G. I., Nicholson, A. E., Chen, Y., & Keogh, E. (2016). Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. Knowledge and Information Systems, 47(1), 1-26.