ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічного матеріалу

ІАЛЦ.467100.006 Д1. Витяг даних. Схема структурна

ІАЛЦ.467100.007 Д2. Модулі програмного додатку. Схема структурна

ІАЛЦ.467100.008 Д3. Система кластеризації. Схема алгоритма

ІАЛЦ.467100.009 Д4. Робота програми. Схема алгоритма

GPS - Система глобального позиціонування (англ. Global Positioning System) — сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері.

JSON - JavaScript Object Notation. Текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, і може бути з легкістю прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

XML - Extensible Markup Language. стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет.

ОС - операційна система.

AJAX - Asynchronous JavaScript And XML. Підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані.

GUI - графічний інтерфейс кори́стувача (ГІК, англ. GUI, Graphical user interface) — тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

SQL (англ. Structured query language — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

API - прикладний програмний інтерфейс (інтерфейс програмування застосунків, інтерфейс прикладного програмування) (англ. Application Programming Interface, API) — набір визначень взаємодії різнотипного програмного забезпечення. API — це зазвичай (але не обов'язково) метод абстракції між низькорівневим та високорівневим програмним забезпеченням.

ВСТУП

На сьогодні, багато популярних мобільних додатків використовують точне місце розташування для спільного використання. Крім того, смартфони з безліччю датчиків дозволяють ідентифікувати координати користувача за допомогою GPS та Wi-Fi з високим ступенем точності.

Значиме місце є регіоном, де користувач зупиняється або уповільнюється для того, щоб виконати важливі види діяльності, такі як дім, місця роботи, ресторани, магазини. GPS може записувати локації користувача як послідовності мітки координати широти і довготи та часу. Таким чином, значимі місця формуються з точок GPS.

Проблема вивчення закономірностей людської поведінки за допомогою сенсорних датчиків виникає в багатьох випадках, в системах інтелектуальних середовищ, у взаємодії людини та робота, а також допоміжних технологіях для інвалідів.

Є багато способів, щоб виявити значимі місця, що базуються на точках перебування або щільності точок GPS. Точки перебування виявляються за допомогою вилучення точок із виконанням умов часу перебування і порогу відстані з траєкторій послідовних GPS точок. Ці точки перебування потім групуються в Region of Interest (ROI) з використанням кластеризації на основі щільності. Центр уваги цих методів на виявленні місць, де користувач залишається протягом деякого періоду часу.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Опис існуючих сервісів, що базуються на локації

Проаналізуємо системи, що визначають значимі місця і є популярними на сьогодні.

Google Maps

Google Maps - набір додатків, побудованих на основі безкоштовного картографічного сервісу і технологій, які надає компанія Google Inc.

Картографічний сервіс Google має функцію під назвою «Ваша хронологія». Він дозволяє переглядати історію відвідування тих чи інших місць на карті, маршрути і багато іншого. Google впевнена, що історія місцезнаходжень дозволяє розширити можливості сервісу, надавши користувачам автоматичний підбір маршруту і розширений пошук завдяки збереженій карті попередніх переміщень. Якщо користувач обере певний відрізок часу або

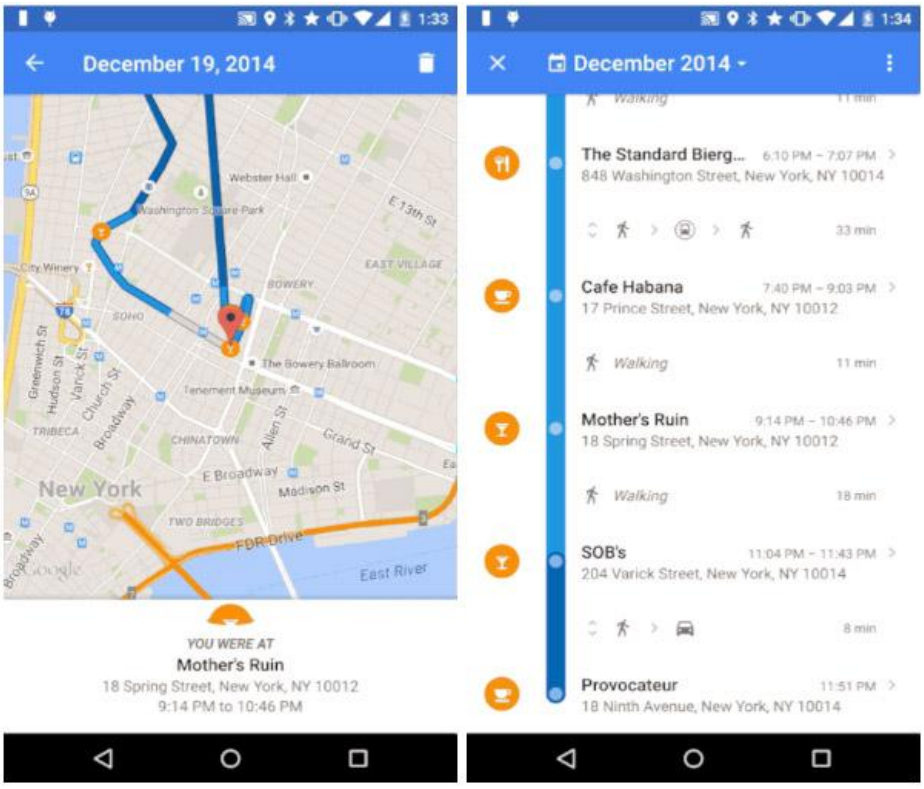

Рис. 1.1 Інтерфейс програми Google Maps

точну дату, то сервіс покаже маршрути переміщень, а також відвідувані місця.

iOS

iOS це власницька мобільна операційна система від Apple. Розроблена спочатку для iPhone, вона стала операційною системою також для iPod Touch, iPad і Apple TV. IOS починаючи з 7 версії має вбудовану функцію «Часто відвідувані місця». Відповідно до опису функції, з її допомогою Apple виконує збір координат GPS і прив'язує їх до поштовою адресою, асоційованого з Apple ID. Смартфон веде історію пересувань і демонструє географічні координати і час, протягом якого людина перебувала в даному місці. Наприклад, 26 раз відвідував роботу з 1 травня 2013 року або 6 серпня перебував у парку ЦПКіВ з 20:15 до 22:54.

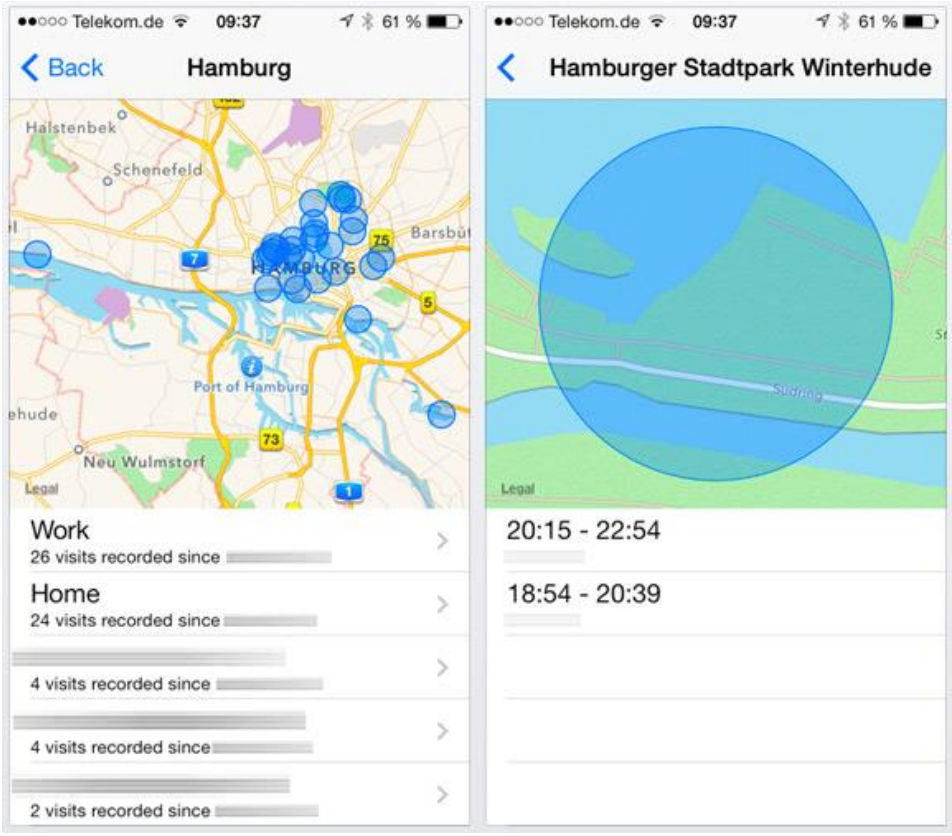

Рис 1.2 Інтерфейс програми iOS Maps

Foursquare

Foursquare - соціальна мережа з функцією геопозиціонування, призначена для роботи з мобільних пристроїв. Сервіс доступний користувачам

не тільки з пристроями, що обладнані GPS-навігацією, наприклад користувачам смартфонів.

Коли користувач запускає програму на своєму мобільному пристрої, отримані з його пристрою геолокаційні дані (довгота і широта) дозволяють надати інформацію, пов'язану з місцем розташування користувача (тобто показати список закладів, що знаходяться поблизу, друзів і підказок)

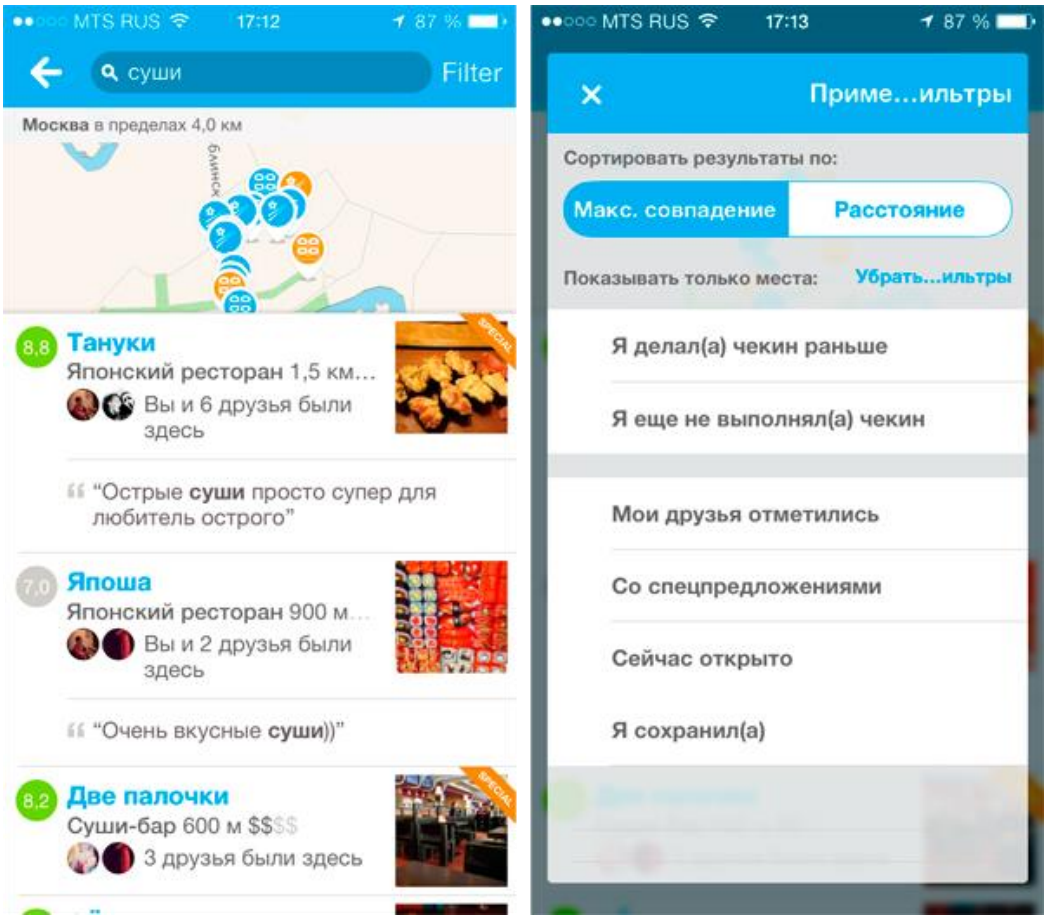

Рис 1.3 Інтерфейс програми Foursquare

1.2 GPS дані

GPS була введена в дію США в 1994 році. Складається вона з 24 супутників і наземних приймальних комплексів, яким може бути і твій GPS-

навігатор або GPS-модуль. Для точного визначення координат твій навігатор повинен бачити мінімум 4 супутника.

Навігатор отримує інформацію з кожного з видимих супутників, які є для нього маяками. GPS-приймач обчислює власне місцезнаходження, вимірюючи час проходження сигналу від GPS-супутників. Кожен супутник постійно надсилає повідомлення, в якому міститься інформація про час, точку орбіти супутника, з якої було надіслано повідомлення, та загальний стан системи й приблизні дані орбіт усіх інших супутників системи GPS. Ці сигнали розповсюджуються зі швидкістю світла в космосі (і з трохи меншою швидкістю — в атмосфері). Приймач визначає час затримки в надходженні сигналу та обчислює відстань до супутників, виходячи з якої, застосувавши метод трилатерації, визначає своє місце[1]. Отримані координати перетворюються в наочну форму (широта та довгота чи положення на карті) та відображаються користувачеві.

Теоретично для визначення власних координати достатньо визначити відстань до трьох супутників. Однак для обчислення положення необхідно знати час із високою точністю. Щоб усунути потребу в високоточному годиннику, отримують інформацію з 4-х чи більше супутників, тобто, GPS-приймач використовує чотири параметри для обчислення чотирьох невідомих: x, y, z та t.

У деяких окремих випадках можна обійтися меншою кількістю супутників. Якщо заздалегідь відома одна змінна (наприклад, висота над рівнем моря човна в океані дорівнює 0), приймач може обчислити положення, використовуючи дані з трьох супутників. Також на практиці приймачі використовують різну допоміжну інформацію для обчислення положення з меншою точністю в умовах відсутності одразу чотирьох супутників.

В даний час у продажу є величезна кількість GPS-навігаторів і GPS-модулів для КПК і ноутбуків. GPS-навігатор це GPS-приймач з екраном, на

якому відображається інформація про твоє місцезнаходження, а GPS-модуль - це GPS-приймач, який підключається до комп'ютера і передає всю навігаційну інформацію програмі, яка з ним працює.

1.3 Методи кластеризації

Кластеризація - це задача розбиття множини об'єктів на групи, які називаються кластерами. У кожній групі повинні виявитися «схожі» об'єкти, а об'єкти різних груп мають бути якомога більш відмінні.
Застосування кластерного аналізу в загальному вигляді зводиться до наступних етапів:

- Відбір вибірки об'єктів для кластеризації.
- Визначення безлічі змінних, за якими будуть оцінюватися об'єкти у вибірці. При необхідності - нормалізація значень змінних.
- Обчислення значень міри схожості між об'єктами.
- Застосування методу кластерного аналізу для створення груп схожих об'єктів (кластерів).
- Представлення результатів аналізу.

Після отримання та аналізу результатів можливе корегування обраної метрики і методу кластеризації до отримання оптимального результату.

Міри відстані

Щоб визначати «схожість» об'єктів, для початку потрібно скласти вектор характеристик для кожного об'єкта - як правило, це набір числових значень. Однак існують також алгоритми, що працюють з якісними характеристиками.[2]

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

Для кожної пари об'єктів вимірюється «відстань» між ними - ступінь схожості. Існує безліч метрик, ось лише основні з них:

Евклідова відстань

Найбільш поширена функція відстані. Являє собою геометричним відстанню в багатовимірному просторі. Обчислюється за формулою 1.1.

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

де x_i – і-та координата першого об’єкта, x_i' – і-та координата другого об’єкта, n – розмірність об’єктів.

Квадрат евклідової відстані

Застосовується для додання більшої ваги більш віддалений друг від друга об'єктів. Це відстань обчислюється за формулою 1.2.

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2 \quad (1.2)$$

де x_i – і-та координата першого об’єкта, x_i' – і-та координата другого об’єкта, n – розмірність об’єктів.

Відстань міських кварталів (Манхеттенська відстань)

Це відстань є середнім різниць по координатах. У більшості випадків ця міра відстані приводить до таких же результатів, як і для звичайного

відстані Евкліда. Однак для цього заходу вплив окремих великих різниць (викидів) зменшується (тому що вони не зводяться в квадрат). Формула для розрахунку манхеттенського відстані:

$$\rho(x,x') = \sum_{i}^{n} | x_i - x_i' |, \quad (1.3)$$

де x_i – і-та координата першого об’єкта, x_i' – і-та координата другого об’єкта, n – розмірність об’єктів

Відстань Чебишева

Це відстань може виявитися корисним, коли потрібно визначити два об'єкти як «різні», якщо вони розрізняються за якоюсь однією координаті. Відстань Чебишева обчислюється за формулою 1.4.

$$\rho(x,x') = \max(| x_i - x_i' |), \quad (1.4)$$

де x_i – і-та координата першого об’єкта, x_i' – і-та координата другого об’єкта, n – розмірність об’єктів

Показникова відстань

Застосовується в разі, коли необхідно збільшити або зменшити вагу, що відноситься до розмірності, для якої відповідні об'єкти сильно відрізняються. Показникова відстань за обчислюється формулою 1.5

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

, де x_i – і-та координата першого об’єкта, x_i' – і-та координата другого об’єкта, n – розмірність об’єктів а r і p - параметри, що визначаються користувачем. Параметр p відповідальний за поступове зважування різниць за окремими координатами, параметр r відповідальний за прогресивне зважування великих відстаней між об'єктами. Якщо обидва параметри - r і p - дорівнюють двом, то це відстань збігається з відстанню Евкліда.

Вибір метрики повністю належить досліднику, оскільки результати кластеризації можуть істотно відрізнятися при використанні різних метрик. класифікація алгоритмів

Цілі кластеризації

- Розуміння даних за допомогою виявлення кластерної структури.
- Розбиття вибірки на групи схожих об'єктів дає можливість спростити обробку даних в подальшому і прийняття рішень, до кожного кластеру застосовуючи власний метод аналізу (стратегія «розділяй і володарюй»).
- Стиснення даних. Коли вихідна вибірка сильно велика, то можна її скоротити, залишивши від кожного кластера по одному типовому представникові.
- Виявлення новизни (англ. Novelty detection). Виділяють нетипові об'єкти, які не виходить ні до одного з кластерів приєднати.

Число кластерів в першому випадку намагаються робити менше. У другому випадку важливішим буде забезпечити більшу ступінь подібності об'єктів в кожному кластері, а кластерів може бути скільки завгодно. Найбільший інтерес в третьому випадку представляють окремі об'єкти, які не вписуються ні в один з кластерів. У всіх цих ситуаціях може використовуватися ієрархічна кластеризація, коли великі кластери дроблять на більш дрібні, ті дробляться в свою чергу ще дрібніші, і так далі. Такі завдання називають завданнями таксономії. Підсумок таксономії - ієрархічна деревоподібна структура. Кожен об'єкт при цьому характеризується перерахуванням кластерів, яким він належить, від великого до дрібного.

Класифікації алгоритмів кластеризації

Алгоритми кластеризації поділяють на:

- Ті, що будують “знизу-догори” та “згори-донизу”

- Монотетичні та політетичні
- Чіткі та нечіткі
- Детерміновані та стохастичні
- Потокові та непотокові
- Залежні від початкового розбиття та не залежні
- Залежні від порядку перегляду об’єктів чи ні
- Ієрархічні та неієрархічні алгоритми

Ієрархічні алгоритми

Принцип роботи даних методів полягає в послідовному об'єднанні маленьких кластерів в великі або навпаки поділі великих кластерів на маленькі.

Відповідно, або на початку роботи алгоритму всі об'єкти є окремими кластерами і на наступних кроках найбільш схожі об'єкти об'єднуються в кластери до тих пір, поки всі об'єкти не об'єднаються в один. Або спочатку всі об'єкти належать одному кластеру, який на наступних кроках розділяється на менші кластери, в результаті чого утворюється послідовність поділяючихся підмножин.

Перевага цієї групи методів - їх наочність і можливість отримання детального уявлення про структуру даних. Недоліки: негнучкість отриманих класифікацій, обмеження обсягу аналізованих даних. Велика складність даних алгоритмів робить їх непридатними при значній кількості досліджуваних даних.

Неієрархічні алгоритми

Неієрархічні методи засновані на поділі набору даних на певну кількість кластерів і виконанні ітеративного процесу оптимізації деякої цільової функції, яка визначає оптимальність (обумовлену особливостями алгоритму) даного розбиття множини об'єктів на кластери. На ітеративний процес накладається умова зупинки, яка в більшості випадків є параметром алгоритму.

Переваги цього типу методів в більш високій стійкості по відношенню до шумів, вибору метрики, додаванню груп незначущих об'єктів у вихідні дані, які беруть участь в кластеризації. Ціну, яку доводиться платити за вибір подібних методів, є знання, якими спочатку повинен володіти дослідник. Необхідно заздалегідь визначити параметри кластеризації, в тому числі кількість кластерів, а також кількість ітерацій або правило зупинки і деякі інші.

- Чіткі і нечіткі

Існує безліч методів кластеризації, які можна класифікувати на чіткі і нечіткі. Чіткі методи кластеризації розбивають вихідну множину об'єктів X на кілька непересічних підмножин. При цьому будь-який об'єкт з X належить тільки одному кластеру. Нечіткі методи кластеризації дозволяють одному і тому ж об'єкту належати одночасно кільком (або навіть всім) кластерам, але з різним ступенем. Нечітка кластеризація в багатьох ситуаціях більш "природна", ніж чітка, наприклад, для об'єктів, розташованих на кордоні кластерів.

Нечітке розбиття дозволяє просто вирішити проблему об'єктів, розташованих на кордоні двох кластерів - їм призначають ступені належності рівні 0.5. Недолік нечіткого розбиття проявляється при роботі з об'єктами, віддаленими від центрів всіх кластерів. Віддалені об'єкти мають мало спільного з будь-яким з кластерів, тому інтуїтивно хочеться призначити для них малі ступені належності. Однак, сума їх ступенів належності така ж, як і для об`єктів, близьких до центрів кластерів, тобто дорівнює одиниці. Для усунення цього недоліку можна використовувати ймовірносне розбиття, яке вимагає, тільки щоб довільний об'єкт з X належав хоча б одного кластеру з деяким ступенем.

Чіткі алгоритми кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одному кластеру. Нечіткі алгоритми кожному об'єкту ставлять у відповідність набір реальних

значень, що показують ступінь відношення об'єкта до кластерів. Тобто кожен об'єкт відноситься до кожного кластеру з певною ймовірністю.

Методи кластеризації також класифікуються за тим, чи визначено кількість кластерів заздалегідь чи ні. В останньому випадку кількість кластерів визначається в ході виконання алгоритму на основі розподілу вихідних даних.

1.4 Python додатки

Python є мовою, що інтерпретується, об'єктно-орієнтована мова високого рівня з динамічною семантикою. Вона містить в собі структури даних, в поєднанні з динамічною типізацією і динамічним зв'язуванням, що робить її дуже привабливою для швидкої розробки додатків. Python має простий, легкий для вивчення синтаксис, має добру читаність і, отже, знижує витрати на обслуговування програми. Python підтримує модулі та пакети, які створюють модульність програми і повторне використання коду. Інтерпретатор Python і велика бібліотека стандартних функцій доступні в вихідному коді або бінарній формі для всіх основних платформ, і може поширюватися вільно.

Переваги

Python забезпечує підвищення продуктивності. Оскільки кроку компіляції немає, редагування тестів та відладження відбуваються неймовірно швидко. Відладження програм на мові Python є легким: помилка або неправильний вхід ніколи не призведе до помилки сегментації. Замість цього, коли інтерпретатор виявляє помилку, він викликає виключення. Коли програма не перехоплює виняток, інтерпретатор друкує трасування стека. Відладчик вихідного рівня дозволяє перевіряти значення локальних і глобальних змінних, проводити оцінку довільних виразів, встановлювати контрольні точки, в той час

крокуючи по коду рядка. Відладчик написаний на самому Python, це свідчить інтроспективну владу Python.

Python стає популярним в Інтернеті речей, так як на ньому засновані нові платформи, такі як Raspberry Pi. Документація Raspberry Pi говорить про мову як «чудова і потужна мова програмування, яку легко використовувати (легко читати і писати) і яка дозволяє підключити ваш проект до реального світу разом з Raspberry Pi.». Ця мова є портативною, маштабуємою і не залежить від системи, і, отже, її підтримують багато одноплатних комп'ютерів на ринку, незалежно від архітектури і операційної системи. Також має значення, що Python має величезне співтовариство, яке забезпечує підтримку бібліотек для мови.

Недоліки

Проблемою може бути швидкість. Інтерпретована мова, часто працює повільніше ніж мови, що транслюються.

Недоліком є обмеження дизайну. Python ентузіасти навели кілька проблем з конструкцією мови. Оскільки мова є динамічно типізованою, це вимагає додаткового тестування. Python додатки мають помилки, які з'являються тільки під час виконання.

Структура програм Python повинна бути послідовною, в інших мовах користувачеві давали більше свободи дужки або інші ідентифікатори, відступи - це те, що має значення, коли мова йде про Python.

2. АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ

2.1. IPython

IPython це командна оболонка для інтерактивних обчислень, розроблена для мови програмування Python. Вона пропонує самоаналіз, мультимедійний синтаксис оболонки, автодоповнення і історію.

Коли ви викликаєте IPython, ви отримаєте інтерактивний інтерпретатор Python, але з деякими особливостями, які роблять його набагато простіше і приємніше у використанні. Кілька з них:

- Вкладка автодоповнення (імена класів, функції, методи, змінні)
- Чіткіші повідомлення про помилки і їх виділення кольором
- Краще управління історією
- оболонки інтеграції Basic UNIX (можна виконувати прості команди оболонки, такі як cp, ls, rm, cp і т.д. безпосередньо з командного рядка Jupyter)
- Гарна інтеграція з багатьма GUI модулями (PyQt, PyGTK і Tkinter)

Причиною величезного успіху IPython є те, що він є формою програмування під назвою «грамотне програмування». Грамотне програмування є стилем розробки програмного забезпечення, який вперше винайшов Стенфордський вчений Дональд Кнут. Цей тип програмування робить важливі експозиції з легким для читання текстом, що перемежовується з кодовими блоками. Він використовується для демонстрації, науково-дослідних і навчальних цілей, особливо для науки. Грамотне програмування дозволяє сформулювати і описати свої думки, доповнені математичними рівняннями, до того, як писати блоки коду.

2.2. Jupyter Notebook

Jupyter Notebook це веб-додаток з відкритим вихідним кодом, який дозволяє створювати і обмінюватися документами, які містять код, рівняння, візуалізації і пояснювальний текст. Області застосування включають в себе: очищення даних і перетворення, чисельне моделювання, статистичне моделювання, машинне навчання і багато іншого.

Ноутбуки можуть бути збережені у вигляді файлів, у системі управління версіями, як і код, і ними можна вільно ділитися. Вони працюють в будь-якому місці, тому що в їх основі. Хоча найвпливовіший ноутбук, Jupyter, має свої витоки на мові програмування Python, тепер підтримує безліч інших мов програмування, в тому числі R, Scala.

Є дві ключових області, які пояснюють чому зберігається популярність ноутбуків.

Співробітництво

Попередні рішення для спільної роботи в галузі науки про дані були вельми обмежені, часто обмежувалися лише обміном SQL запитів. Сучасна наука про дані відбувається набагато швидше, аналіз може відбуватися за участю безлічі інструментів, таких як Python, Hadoop і Spark. Вона також розвивається рука об руку з інжинірингом, де відбувся великий стрибок в системі контролю версій, системі управління проблемами, відстеженнях проекту, і багато іншого. Ноутбуки, будучи по суті файлами, можуть вільно використовуватися в мережах управління версіями, такі як Github, їми легко обмінюватися у організації. Більш того, ноутбук дозволяє легко додавати текст і діаграми, передивлятися сам код, що поясняє чому ноутбуки використовуються в науці про дані.

Легкий доступ до ресурсів

Ноутбуки дають вченим легкий доступ до даних і обчислювальної потужності. Так як ноутбуки розміщуються в браузері, немає необхідності відправляти дані на комп'ютер, щоб оброблювати дані. Це означає, що вчені

можуть працювати безпосередньо на кластері, який зберігає всі необхідні їм дані, що доводить, що ноутбуки ідеально підходять для проведення аналізів великих даних. Ноутбуки дають вченим доступ до обчислювальних кластерів , що масштабуються, не вимагаючи від них, щоб вони мали знання про розподілені системи або тримали у себе в DevOps спеціаліста.

Ноутбуки змінюються, як і роботи науковців про дані, завдяки поєднанню веб-браузера призначеного для інтерфейсу, з відкритим вихідним кодом, і масштабованої хмари рішень для великих даних. Тепер науковці витрачають менше часу для доступу, відбору проб і транспортування даних.

2.3. Google Maps API

Google Maps API є однією з найпопулярніших JavaScript бібліотек в Інтернеті, яка використовуються більш ніж на 350000 сайтів. І з ясною причиною – ця бібліотека потужна і протягом шести років була абсолютно вільною.

Google надає розширені можливості: потужної маршрутизації (в тому числі і для прогулянок, їзди на велосипеді, і транзит), Street View, 3D будівлі, погоду та інформацію про дорожній рух. Деякі з цих особливостей є унікальними для Google, так що розробник не може мати ніякого вибору, окрім як використовувати його API. Google Maps поставляється з трьома базовими картами: вулицями, супутником і місцевістю.

Використовуючи Google Play services location API, додаток може отримати останнє відоме місце розташування пристрою користувача. У більшості випадків, потрібно знати поточне місцезнаходження користувача, яке, як правило, еквівалентне останньому відомому розташуванні пристрою. Зокрема, використовується об’єднання різних геолокаційних провайдерів для визначення місцезнаходження користувача. Об’єднаний геолокаційний сервіс є одним із location APIs в Google Play services, який також оптимізує використання заряду батареї пристроєм.

Після того,як додаток підключений до Google Play services і locations services API, він може отримати останнє відоме місце розташування пристрою користувача. Коли додаток підключений до цих служб ,він може використовувати доступний провайдер місцезнаходження getLastLocation(), метод який видає розташування пристрою. Точність розташування, яка надається цим провайдером, визначається параметром, який записаний в маніфесті програми.

Якщо додаток може безперервно відстежувати місце розташування, то можна надавати користувачеві більш точну та актуальну інформацію. Наприклад, якщо додаток показує маршрут шляху користувачеві під час руху, або ж, якщо додаток відстежує місце розташування активних об’єктів на карті, він має отримувати місце розташування пристрою з регулярними інтервалами. Так само як і географічне розташування (широта і довгота), програма може надати користувачеві додаткову інформацію, таку як напрям руху, висоту над рівнем моря та швидкість пристрою. Цю та іншу інформацію можна знайти в об'єкті Location, який може бути отриманий додатком з включеного провайдера місцезнаходження.

У той час як програма можете отримати місце розташування пристрою з об’єкта getLastLocation(), потрібно використати більш прямий підхід, щоб зробити запит періодичних оновлень від доступного провайдера місцезнаходження. У відповідь на цей запит, API періодично відправляє оновлені дані, ґрунтуючись на наявних в даний час провайдерах визначення місця розташування, таких як Wi-Fi і GPS (Global Positioning System). Точність розташування визначається провайдерами, дозволами по місцеположенням, що вказані, і встановленими параметрам в запиті місцезнаходження.

Сервіси місцезнаходження для додатків надаються службами Google Playі доступними провайдерами місцезнаходження. Для того, щоб використовувати ці послуги, програма підключається до клієнта Google API, а потім відправляє запити на оновлення місце розташування.

Останнє відоме розташування пристрою забезпечує зручну базу для початку, гарантуючи, що програма має відоме місце розташування перед початком періодичних оновлень місцезнаходження.

2.4. Види запитів: JSON та XML

JSON

JSON— це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, і може бути з легкістю прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON виступає як заміна XML під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-додатки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси). Формат JSON також добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

JSON будується на двох структурах:

1. Набір пар ім'я/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом.

2. Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі форми:

Об'єкт — це невпорядкована множина пар ім'я/значення. Об'єкт починається з символу { і закінчується символом }. Кожне значення слідує за : і пари ім'я/значення відділяються комами.

Масив — це впорядкована множина значень. Масив починається символом [і закінчується символом]. Значення відділяються комами. Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

Рядок — це впорядкована множина з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються із зворотної косої риски (backslash). Символи представляються простим рядком. Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

XML

Розширювана мова розмітки (Extensible Markup Language, скорочено XML) — запропонований консорціумом World Wide Web (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML документ складається із текстових знаків, і придатний до читання людиною.

Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої, шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (англ. Schema), такими як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, основаних на XML є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, і також XML Schema.

Коректність

Коректний документ (well-formed document) відповідає всім синтаксичним правилам XML. Документ, що не є коректним, не може називатись XML-документом. Сумісний синтаксичний аналізатор (англ. Conforming parser) не повинен обробляти такі документи.

Зокрема, коректний XML документ має:

1. Лише один елемент у корені.

2. Непорожні елементи розмічено початковим та кінцевим тегами (наприклад, <пункт>Пункт 1</пункт>). Порожні елементи можуть позначатися «закритим» тегом, наприклад <IAmEmpty />. Така пара еквівалентна <IAmEmpty></IAmEmpty>.
3. Один елемент не може мати декілька атрибутів з однаковою назвою. Значення атрибутів перебувають або в одинарних ('), або у подвійних (") лапках.
4. Теги можуть бути вкладені, але не можуть перекриватись. Кожен некореневий елемент мусить повністю перебувати в іншому елементі.
5. Документ має складатися тільки з правильно закодованих дозволених символів Юнікоду. Єдиними кодуваннями, які обов'язково має розуміти XML-процесор, є UTF-16 та UTF-8. Фактичне та задекларованe кодування (англ. character encoding) документа мають збігатись. Кодування може бути задекларовано ззовні, як у заголовку «Content-Type» при передачі по протоколу HTTP, або в самому документі використанням явної розмітки на самому початку документа. У разі відсутності інформації про кодування, документ має бути в кодуванні UTF-8 (або його підмножині ASCII).

Валідність

Документ називається валідним (valid), якщо він є коректним, містить посилання на граматичні правила та повністю відповідає обмеженням, вказаним у цих правилах (DTD або XML Schema або іншому подібному документі).

Синтаксичний аналізатор

Синтаксичним аналізатором (часто, парсер від англ. parser) називається

програма або компонент, що читає XML-документ, проводить синтаксичний аналіз, та відтворює його структуру. Якщо синтаксичний аналізатор перевіряє документ на валідність, то такий аналізатор називають валідатором (англ. validating).

Правильний вибір назв для XML елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних. Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів.

2.5 Алгоритм кластеризації DBSCAN

Алгоритм DBSCAN був запропонований Мартін Естер, Ганс-Пітером Кригель у 1996 рік як вирішення проблеми розбиття (спочатку просторових) даних на кластери довільної форми. Більшість алгоритмів, які проводять плоске розбиття, створюють кластери формою близькою до сферичної, так як мінімізує відстань документів до центру кластера. Автори DBSCAN експериментально показали, що їх алгоритм здатний розпізнати кластери різної форми[3].

Ідея, покладена в основу алгоритму, полягає в тому, що всередині кожного кластера щільність точок (об'єктів) помітно вище, ніж щільність зовні кластера, а також щільність в областях з шумом нижче щільності будь-якого з кластерів. Для кожної точки кластера її околиця в діапазоні заданого радіусу повинна містити не менше деякого числа точок, яке задається граничним значенням. У загальному випадку алгоритм DBSCAN має квадратичну обчислювальну складність через пошуку Eps-сусідства-O ($N2$). Однак автори алгоритму використовували для цієї мети спеціальну структуру

даних - R * -дерева, в результаті пошук Eps-сусідства для однієї точки - O (logn). Загальна обчислювальна складність DBSCAN - O (n * logn).

Формальний підхід

Введемо кілька визначень. Нехай задана деяка симетрична функція відстані ρ (x, y) і константи ε і т. тоді

1) Назвемо область E (x), для якої $\forall$ y: ρ (x, y) ≤ε, ε-околиця об'єкту x.
2) Кореневом об'єктом або ядерним об'єктом ступені m називається об'єкт, ε-околиця якого містить не менше м об'єктів: | E (x) | ≥ m.
3) Об'єкт p безпосередньо щільно-досяжний з об'єкта q, якщо p$\in$E (q) і q - кореневий об'єкт.
4) Об'єкт p щільно-досяжний з об'єкта q, якщо $\exists$ p_1, p_2 ... p_n, p_1 = q, p_n = p, такі що $\forall$ i$\in$1 ... n-1: p_{i+1} безпосередньо щільно-досяжний з p_i

Оберемо будь-який кореневий об'єкт p з датасета, помітимо його і помістимо всіх його безпосередньо щільно-досяжних сусідів в список обходу. Тепер для кожного q зі списку: помітимо цю точку, і, якщо вона теж коренева, додамо всіх її сусідів в список обходу. Тривіально доводиться, що кластери позначених точок, сформовані в ході цього алгоритму максимальні (тобто їх не можна розширити ще однією точкою, щоб задовольнялося умова) і зв'язні в сенсі щільно-досяжності. Звідси випливає, що якщо ми обійшли не всі точки, можна перезапустити обхід з будь-якого іншого кореневого об'єкта, і новий кластер не буде частиною попереднього.

Нюанси застосування

В ідеальному випадку DBSCAN може досягти складності O (N), але не варто на це розраховувати. Якщо не перераховувати кожного разу E (x) від точок, то очікувана складність - O (NlogN). Найгірший випадок (погані дані або брутфорс-реалізація) - O (N^2). Наївні реалізації DBSCAN використовують O (N^2) пам'яті під матрицю відстаней - це надлишково. Багато версій алгоритму вміють працювати і з більш щадними структурами даних: sklearn і R реалізацією можна оптимізувати за допомогою KD-дерева.

DBSCAN з невипадковим правилом обробки крайових точок є детермінованим. Однак більшість реалізацій для прискорення роботи і зменшення кількості параметрів віддають крайові точки першим кластерам, які до них дотяглися. Наприклад, центральна жовта точка на зображенні в різних запусках може належати як нижнього, так і верхнього кластеру. Як правило, це не впливає на якість роботи алгоритму, адже через граничні точки кластер все одно не поширюється далі - ситуація, коли точка перескакує з кластера в кластер і «відкриває дорогу» до інших точок, неможлива.

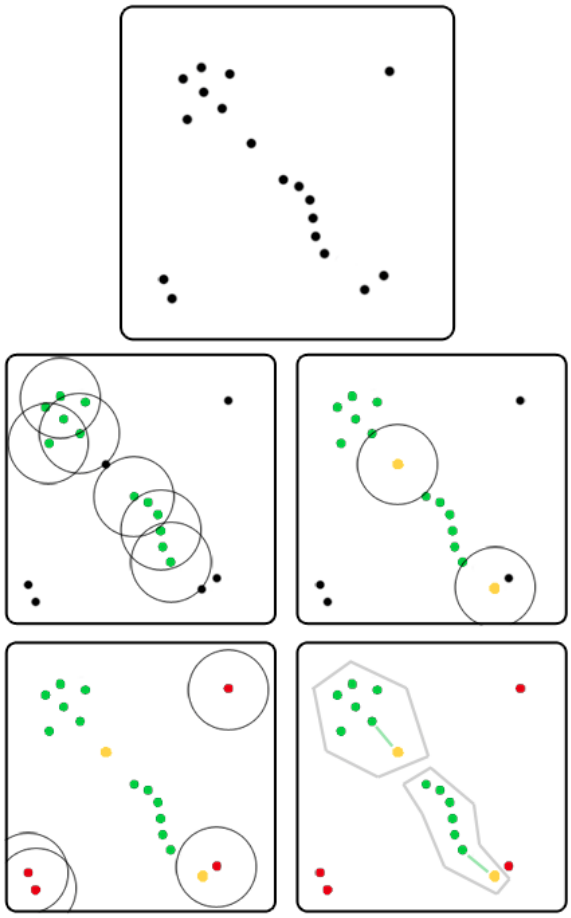

Рис 2.1 Приклад роботи алгоритму DBSCAN

DBSCAN не вираховує самостійно центри кластерів, однак це проблема, особливо з огляду на довільну форму кластерів. Зате DBSCAN автоматично визначає викиди, що досить корисно.

Співвідношення m/ε^n, де n - розмірність простору, можна інтуїтивно розглядати як порогову щільність точок даних в області простору. Очікувано, що при однаковому співвідношенні m/ε^n, і результати будуть приблизно однакові. Іноді це дійсно так, але є причина, чому алгоритму потрібно задати два параметра, а не один. По-перше типова відстань між точками в різних датасетах різний - явно задавати радіус доводиться завжди. По-друге, відіграють роль неоднорідності датасета. Чим більше m і ε, тим більше алгоритм схильний «пробачати» варіації щільності в кластерах. Це може бути

корисно: неприємно побачити в кластері «дірки», де просто не вистачило даних. З іншого боку, це шкідливо, коли між кластерами немає чіткої межі або шум створює «міст» між скупченнями. Тоді DBSCAN просто з'єднає дві різні групи. У балансі цих параметрів і криється складність застосування DBSCAN: реальні набори даних містять кластери різної щільності з межами різного ступеня розмитості. В умовах, коли щільність деякого кордону між кластерами більше або дорівнює щільності якихось відокремлених кластерів, чимось доводиться жертвувати.

Існують варіанти DBSCAN, здатні пом'якшити цю проблему. Ідея полягає в підстроювання ε в різних областях по ходу роботи алгоритму. На жаль, зростає кількість параметрів алгоритму.

Існує евристика для вибору m і ε.

1) Оберіть m. Зазвичай використовуються значення від 3 до 9, чим більш неоднорідний очікується датасет, тим більший рівень шуму, тим більшим слід взяти m.
2) Обчисліть середню відстань по m найближчих сусідів для кожної точки. Тобто якщо m = 3, потрібно вибрати трьох найближчих сусідів, скласти відстані до них і поділити на три.
3) Сортуємо отримані значення по зростанню і виводимо на екран.
4) Бачимо щось на кшталт такого різко зростаючого графіка. Слід взяти ε в смузі, де відбувається найсильніший перегин. Чим більше ε, тим більше вийдуть кластери, і тим менше їх буде.

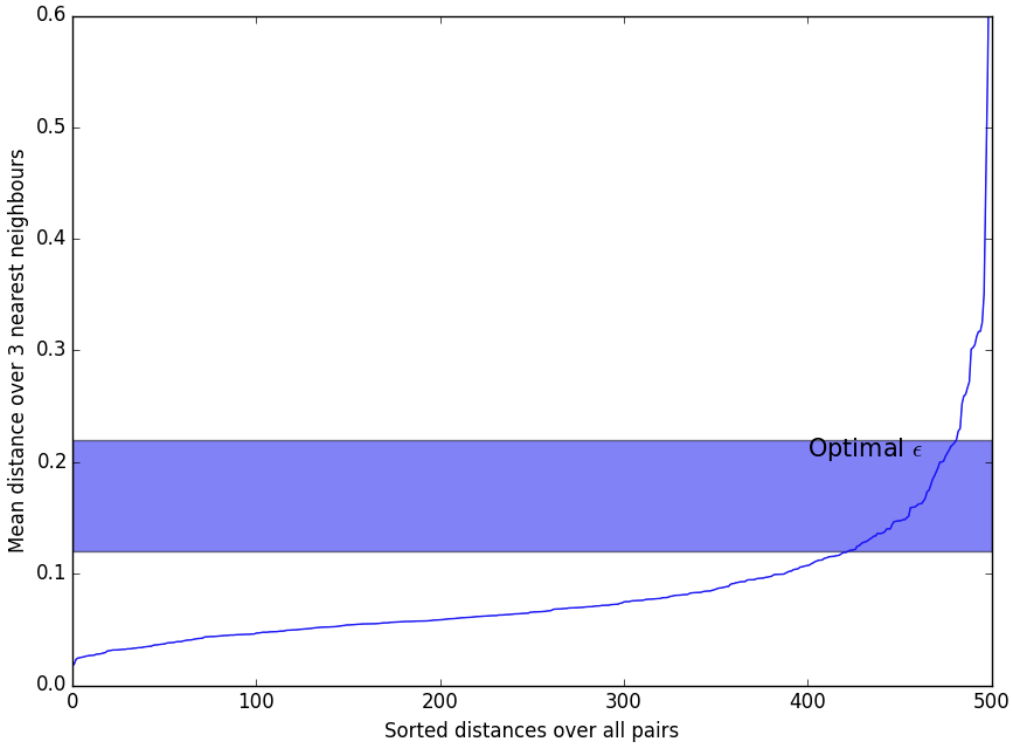

Рис 2.2 Залежність вибору оптимального ε

У будь-якому випадку, головні недоліки DBSCAN - нездатність з'єднувати кластери через отвори, і, навпаки, здатність пов'язувати явно різні кластери через щільно населені перемички. Саме тому при збільшенні розмірності даних n стає проблема розмірності: чим більше n, тим більше місць, де можуть випадково виникнути отвори або мости. Адекватна кількість точок даних N зростає експоненціально зі збільшенням n.[3]

DBSCAN використовується, коли:

- В міру великий ($N \approx 10^6$) датасет. Навіть $N \approx 10^7$-10^8 якщо використовується оптимізована і розпаралленна реалізація.
- Заздалегідь відома функція близькості, симетрична, бажано, не дуже складна. KD-дерево оптимізація часто працює тільки з евклідовою відстанню.

- Ви очікуєте побачити згустки даних екзотичної форми: вкладені і аномальні кластери, складки малої розмірності.
- Щільність кордонів між згустками менше щільності найменш щільного кластера. Краще якщо кластери зовсім відокремлені один від одного.
- Складність елементів датасета значення не має. Однак їх повинно бути досить, щоб не виникало сильних розривів в щільності.
- Кількість елементів в кластері може варіюватися як завгодно.
- Кількість викидів значення не має, якщо вони розсіяні по великому обсягу.
- Кількість кластерів значення не має.

2.6 scikit-learn та pandas

Scikit-learn є вільним програмним забезпеченням для машинного навчання для мови програмування Python. Він має різні ралізовані алгоритми класифікації, регресію і алгоритми кластеризації, включаючи підтримку векторних машин, випадкового лісу, градієнт підвищення, K-means і DBSCAN, і призначений для взаємодії з Python та чисельними й науковими бібліотеками NumPy і SciPy.[4]

Проект scikit learn був створений як scikits.learn, на проекті Google Summer Code 2007 David Cournapeau. Станом на 2017 рік, scikit learn знаходиться в стадії активного розвитку.

Scikit learn в основному написаний на Python, з деякими алгоритмами, написаними на Cython для збільшення продуктивності. Підтримка векторної машини реалізується обгорткою Cython навколо LIBSVM; логістична регресія і лінійні опорні вектори за аналогічною обгортці навколо LIBLINEAR.

Переваги

- Прагнення до зрозумілої документації та зручності використання

Розробники, що пишуть в scikit learn зобов'язані включати приклади з описами поряд з типовими сценаріями, які працюють на невеликих наборах даних. Крім гарної документації є й інші основні принципи, якими керується загальна спільнота для якості і практичності: гарантується глобальний API, всі публічні API, добре задокументовані, і при наявності відповідних вкладів до проекту рекомендується розширити охоплення модульних тестів.

- Моделі обрані і реалізовані спеціальною групою експертів

Постійні розробники scikit learn включають в собі фахівців в області машинного навчання і розробки програмного забезпечення.

- Охоплює більшість завдань машинного навчання

Scikit learn включає в себе інструменти для багатьох стандартних задач машинного навчання (наприклад, кластеризація, класифікація, регресія і т.д.). А оскільки scikit leatn розробляється великим співтовариством розробників і експертів по машинному навчанню, створені нові технології, як правило, включаються в досить короткі терміни.

Користувачі не повинні обирати з декількох конкуруючих реалізацій одного і того ж алгоритму (проблема, з якою часто стикаються користувачі R). Існує проста схема для користувачів, що допоможе обрати необхідний варіант.

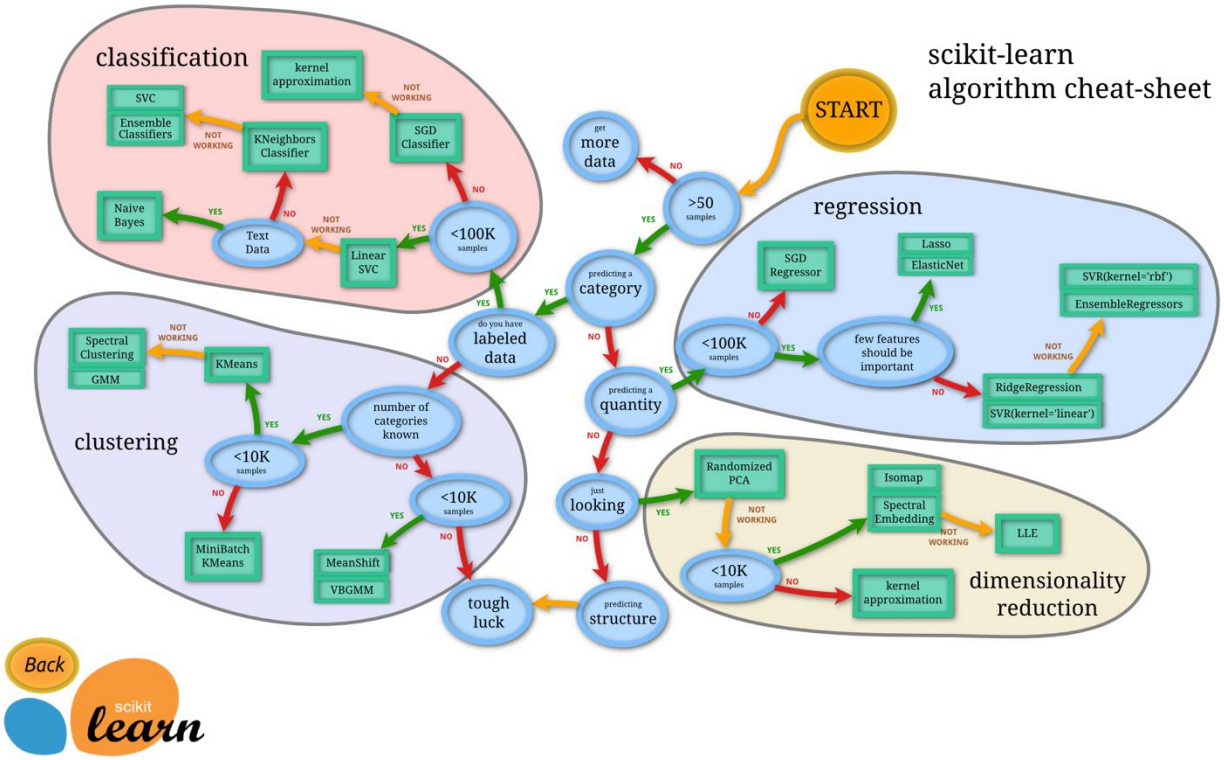

Рис 2.3 Схема для вибору оптимального алгоритму

- Python і Pydata

Інтерпретатор Python дозволяє користувачам взаємодіяти і грати з наборами даних, і з самого початку це зробило мову привабливим для аналітиків даних. Що ще більш важливо значний набір інструментів даних Python (pydata) з'явився протягом останніх кількох років

Багато вчених даних регулярно працюють з багатьма інструментами pydata включаючи scikit learn, IPython і Matplotlib. Звичайна практика при використанні scikit learn - створювати Matplotlib графіки для оцінки якості даних або налагодження моделі

Ще одна ознака того, що Python виник як пріоритетна мова для вчених про дані: нові аналітичні інструменти, такі як Spark (PySpark), GraphLab (GraphLab ноутбук), і Adatao всі підтримують Python.

- Фокус

Scikit learn - бібліотека машинного навчання. Її мета полягає в тому, щоб забезпечити набір загальних алгоритмів для користувачів Python через послідовний інтерфейс. Співтовариство нещодавно вирішило, що Deep Learning має досить спеціалізовані вимоги (велике число гіпер-параметрів, обчислення на GPU вводить нову складну програмну залежність), що краще включити його в новому проекті. Розробники scikit learn вирішили реалізувати базові нейронні мережі в якості будівельних блоків (багатошарового персептрона і обмежувальної больцманівської машини).

- Scikit learn маштабується для більшості проблем

Проблемами в Python є швидкість і масштаб. Виявляється, що в той час як маштабуємість може бути проблемою, дійсно це відбувається не так часто. Багато проблем можуть бути вирішені за допомогою одного сервера (з великою пам'яттю), і добре розробленого програмного забезпечення, яке працює на одній машині може замінити розподілені системи. Інші методи, такі як відбір проб також можуть бути використані для підготовки моделей на масивних наборів даних.

Але бувають випадки, коли комбінація великого розміру даних і робочого процесу диктує свій вибір інструментів. Іноді потрібно звертатися до ETL інструменту (наприклад, Spark, MapReduce, Scalding).

Pandas

Pandas бібліотека з відкритим вихідним кодом, BSD-ліцензована, що забезпечує високу продуктивність, легка у використанні структур даних і інструментів аналізу даних для мови програмування Python.

Акронім pandas походить від комбінації panel data, економетричного терміна, а також data analysis Python. Бібліотека націлена на п'ять типових кроків обробки і аналізу даних, незалежно від їх походження: завантаження, підготовка, маніпуляції, моделі і аналіз.[5]

Інструменти, що надані pandas заощаджують час при завантаженні даних. Бібліотека може читати записи в форматі CSV (значення, розділені комами), Excel, HDF, SQL, JSON, HTML і формати Stata; pandas приділяють велику увагу гнучкості, наприклад, при роботі з різнорідними сепараторів секцій. Крім того, вона читає об'єкти безпосередньо з кеша або завантажує серіалізовані у файл Python об'єкти за допомогою модуля pickle Python.

Підготовка завантажених даних, відбувається в такий спосіб. Записи видаляються, якщо виявляються помилкові дані або встановлюються значення за замовчуванням, потім нормалізується, згруповуваються, розсортовуються, трансформуються і таким чином стають пристосованими для подальшої обробки. Ця підготовча робота, як правило, включає в себе трудомісткі види діяльності, які варто стандартизувати, перш ніж почати інтерпретувати зміст.

Основні можливості бібліотеки pandas:

- Об'єкт DataFrame для маніпулювання індексованими масивами двовимірних даних
- Інструменти для обміну даних між структурами в пам'яті і файлами різних форматів
- Засоби поєднання даних і способи обробки відсутньої інформації
- Переформатування наборів даних, в тому числі створення зведених таблиць
- Зріз даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних
- Вставка і видалення стовпців даних
- Можливості групування дозволяють виконувати трьохетапну операції типу «поділ, зміна, об'єднання»
- Злиття і об'єднання наборів даних
- Ієрархічне індексування дозволяє працювати з даними високої розмірності в структурах меншої розмірності

- Робота з тимчасовими рядами: формування тимчасових періодів і зміна інтервалів і т. д.

Бібліотека оптимізована для високої продуктивності, найбільш важливі частини коду написані на Cython і C.

NumPy

NumPy - це розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою кількістю високорівневих математичних функцій для операцій з цими масивами.

Numberic був початковим пакетом, з умовою ефективних однорідних числових масивів для Python, але деякі розробники вважали, що йому не вистачає деяких істотних функцій, тому вони почали розробку незалежної реалізації під назвою Numarray. Наявність двох несумісних реалізацій масиву явно несли лихо, тому NumPy був розроблений, щоб бути покращенням для обох.

Математичні алгоритми, реалізовані на інтерпретованих мовах (наприклад, Python), часто працюють набагато повільніше ніж алгоритми, реалізовані на компільованих мовах (наприклад, Фортран, Сі, Java). Бібліотека NumPy надає реалізацію обчислювальних алгоритмів (у вигляді функцій і операторів), що оптимізовані для роботи з багатовимірними масивами.[8] В результаті будь-який алгоритм, який може бути виражений у вигляді послідовності операцій над масивами (матрицями) і реалізований з використанням NumPy, працює так само швидко, як еквівалентний код, що виконується в MATLAB.

Списки у Python є ефективними контейнерами загального призначення. Вони підтримують досить ефективну вставку, видалення, додавання і конкатенацію, їх легко побудувати і ними маніпулювати. Проте, у них є певні обмеження: вони не підтримують «векторизованних» операцій, такі як поелементного доповнення і множення, а також той факт, що вони можуть містити об'єкти різного типу означає, що Python має зберігати інформацію про тип для кожного елемента, і повинен виконатувати диспетчерський код при

роботі на кожному елементі. Це також означає, що дуже мало операції зі списками можуть здійснюватися за допомогою ефективних C циклів - кожна ітерація вимагає перевірки типу і інший Python API перевірки.

Масив NumPy є багатовимірним масивом об'єктів одного і того ж типу. На згадку, це об'єкт, який вказує на блок пам'яті та відстежує тип даних, що зберігається в цій пам'яті, відстежує розміри і наскільки вони великі , і головне зберігає відстань між елементами уздовж кожної осі.

Наприклад, ви можете мати масив NumPy, що представляє число від нуля до дев'яти, що зберігається в 32-розрядних цілих числах, один після іншого, в одному блоці пам'яті. (Для порівняння, кожне ціле число Python має мати деяку інформацію про тип, що зберігається разом з ним). Ви також можете мати безліч парних чисел від нуля до восьми, що зберігаються в одному блоці пам'яті, але з проміжком в чотири байти (одне 32-розрядне ціле число) між елементами. Це означає, що ви можете створити новий масив, який посилається на підмножини елементів в масиві без копіювання будь-яких даних. Такі підмножини називаються видами. Це очевидно посилює ефективність, але це також дозволяє змінювати обрані елементи масиву різними способами.

Важливе обмеження на NumPy масиви є те, що для даної осі, всі елементи повинні бути відокремлені один від одного таким же числом байтів в пам'яті. NumPy не може використовувати подвійну опосередкованість для доступу до елементів масиву. Це обмеження дозволяє все внутрішні цикли в коді Numpy записати в ефективному коді C.

3. РОЗРОБКА ПІДСИСТЕМИ КЛАСТЕРИЗАЦІЇ

3.1. Розробка концепції рішення

Вибір мови програмування

Для розробки програмного продукту було обрано мову програмування Python.

Вибір цієї мови зумовлений рядом причин:

- Python – інтерпритована об’єктно-орієнтовна мова програмування, що дозволяє зробити систему гнучкою до змін;
- має велику кількість модулів;
- має ефективні структури даних.

Вибір допоміжних компонентів програми

- Для пошуку значимих місць (кластерів) користувача використовується модуль scikit learn
- Для завантаження, обробки та зберігання вхідних даних використовується бібліотка pandas
- Для відображення кластерів на карті використовуються карти від Google.
- Взаємодія з ними відбувається через клієнт Google Maps API.
- Для обміну інформацією з підсистемою використовуються файли з даними у форматі JSON

Загальна схема роботи

Роботу підсистеми можно розділити на делілька етапів:

- Конвертація вхідних даних з JSON формату у внутрішній бінарний формат
- Представлення даних у виді, необхідному для кластеризації
- Кластеризація даних методом DBSCAN
- Конвертація отриманих результатів з векторного вигляду у зручне для сприйнятя людиною представлення на мапі

Для перевірки роботи алгоритму був використаний набір даних GeoLife GPS Trajectory Data.

Набір даних GeoLife містить датовані широти, довготи і висоти для 182 різних користувачів. Загальна кількість точок у датасеті - 17,621, що охоплює 2007-2012 роки.

Дані були зібрані з телефонів користувачів і інших реєстраторів GPS для моделювання діяльності, таких як шляхи переміщення, магазини, екскурсії тощо.

Деякі записи користувачів містять додаткову інформацію про режим транспортування.

Файл з даними має наступний вигляд

Поле 1: Широта в десяткових градусах.

Поле 2: довгота в десяткових градусах.

Поле 3: Все встановлено у 0 для цього набору даних.

Поле 4: Висота в футах (-777 якщо немає).

Поле 5: Дата - кількість днів (з дробовою частиною), що пройшла з 12/30/1899.

Поле 6: Дата у вигляді рядка.

Поле 7: Час у вигляді рядка.

Поле 5 і поля 6, 7 представляють ту ж дату / час в цьому наборі даних. Ми можемо використовувати будь-яку з них.[9]

GPS дані з цього датасету збираться кожні 1-5 секунд, цього забагато для кластеризації, для уникнення високого навантаження та збільшення швидкості алгоритму, я зменшила кількість даних до одного GPS сигналу у хвилину. Цього достатньо для валідної роботи алгоритму.

Сумарна кількість GPS точок після цієї модифікації для кожного користувача з датасету стає рівною приблизно 9-10 тисяч.

3.2. Опис програмного забезпечення

Після завантаження нових GPS даних для користувача відображаємо його траєкторії за допомогою Google Maps.

Для створення мапи з траєкторіями використовується бібіотека для Python – gmplot, що містить просту обгортку навколо служби геокодування Google, що дозволяє створювати карту за місцем вибору. Що має клас GoogleMapPlotter. Для першого користувача траєкторії мають вигляд, як на рис 3.1.

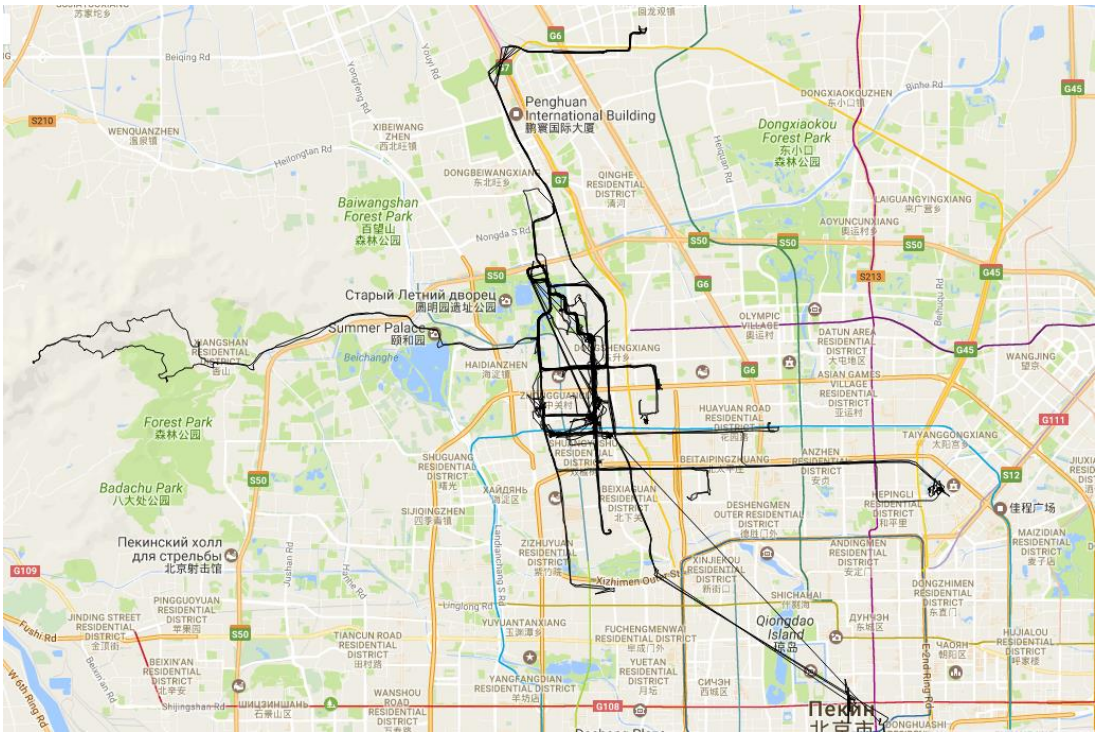

Рис 3.1 Відображення траєкторій користувача

Для роботи алгоритму DBSCAN потрібно обрати правильне eps. Це максимальна відстань, що може бути між GPS точками, що знаходяться у одному кластері. Для цього випадку у я призначаю eps значення прямо пропорційне 0.5 та обереннено пропорційне радіусу Землі у кілометрах.

Радус Землі дорівнює 6371.0088 кілометрів, тому

eps = 0.5/6371.0088 = 0.000078

Для корректної роботи алгоритму також важливо обрати мінімальну кількість GPS точок у кластері, все останнє буде позначене, як шум.

Для цього прикладу мінімальна кількість буде становити 100, це значення було обрано з проміжку 10-1000, так, щоб результати кластеризації залишалися валідними.

Також має буди обраний алгоритм, за котрим буде обчислюватися відстані між точками та шукатися найближчий сусід.

Існує два можливих варіанти обчислення – KD дерево або ball tree.

KD дерева в цілому намагаються зменшити необхідну кількість обчислень відстаней за допомогою ефективного кодування сукупної інформації про відстані для точки. Основна ідея полягає в тому, що якщо точка A знаходиться дуже далеко від точки B, а точка B знаходиться дуже близько до точки C, то ми знаємо, що точки A і C знаходяться дуже далеко, без явного обчислення їх відстаней. Таким чином, розрахункова вартість найближчого пошуку сусідів з N точками з D вимірами може бути зведена до

O(D*N*LOG (N)] або краще. KD дерева погано працюють, коли приклади мають велику вимірність.

Щоб уникнути неефективності KD дерев при більш високих вимірах, була розроблена структура даних ball tree. Коли KD дерева розділяють дані уздовж декартових осей, ball tree розділяє дані у серії вкладенних гіпер-сфер. Це робить побудову дерева більш дорогим, ніж у дерева KD, але призводить до створення структури даних, що може бути дуже ефективною на високо структурованих даних, і навіть в дуже високих вимірах.

Ball tree рекурсивно ділить дані на вузли, визначені за допомогою центроїда C і радіусом r, таким чином, що кожна точка в вузлі лежить в межах гіпер-сфери, яка визначається r і C. Число кандидатів точок для пошуку сусідів знижується за рахунок використання нерівності трикутника. За допомогою цієї установки, одного розрахунку відстані між контрольною точкою і центром ваги досить, щоб визначити нижню і верхню межу на відстані до всіх точок в межах вузла.

У цьому випадку використовується алгоритм ball tree через те, що він швидко працює з даними, що мають велику розмірність.[7]

Використовуючи класс з scikit learn DBSCAN кластеризуємо попередньо отримані та відформатовані дані. DBSCAN сам обирає кількість необхідних кластерів на вхідних даних. Для користувача з датасету GeoLife під номером 1, з 9045 GPS точок було створено 4 кластери.

Відображаємо приблизні траекторії користувача та позначаємо знайдені кластери, використовуючи бібліотеку matplolib. Отримуємо зображення на рис 3.2.

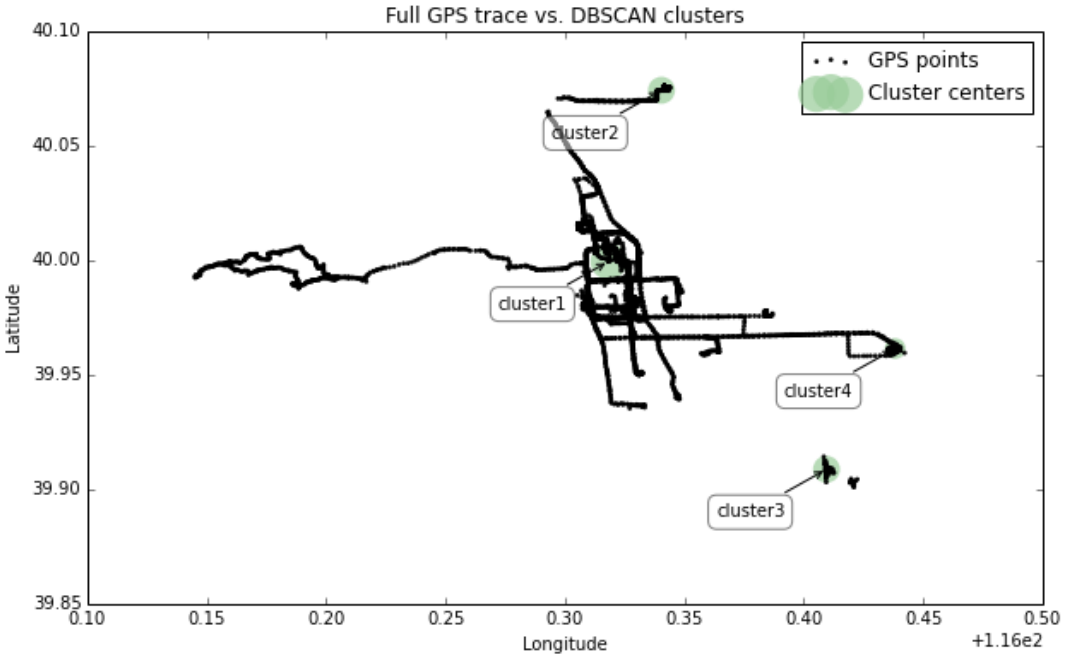

Рис 3.2 Відображення результатів кластеризації

Для наглядності кластерів побудуємо графік перебування користувача у межах кластеру в залежності від години. Отримуємо результат на рис 3.3.

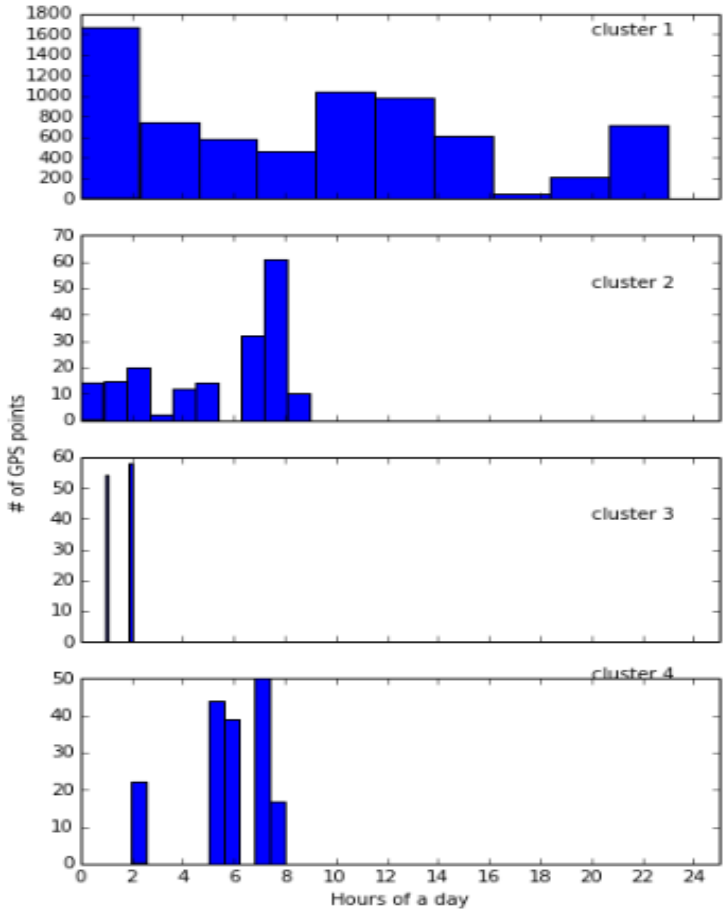

Рис 3.3 Графік залежності перебування у кластері від часу

Нехтуємо кластерами в яких користувач перебуває недостатньо часу, менше за встановлену величину. Отримуємо резутат на рис 3.4.

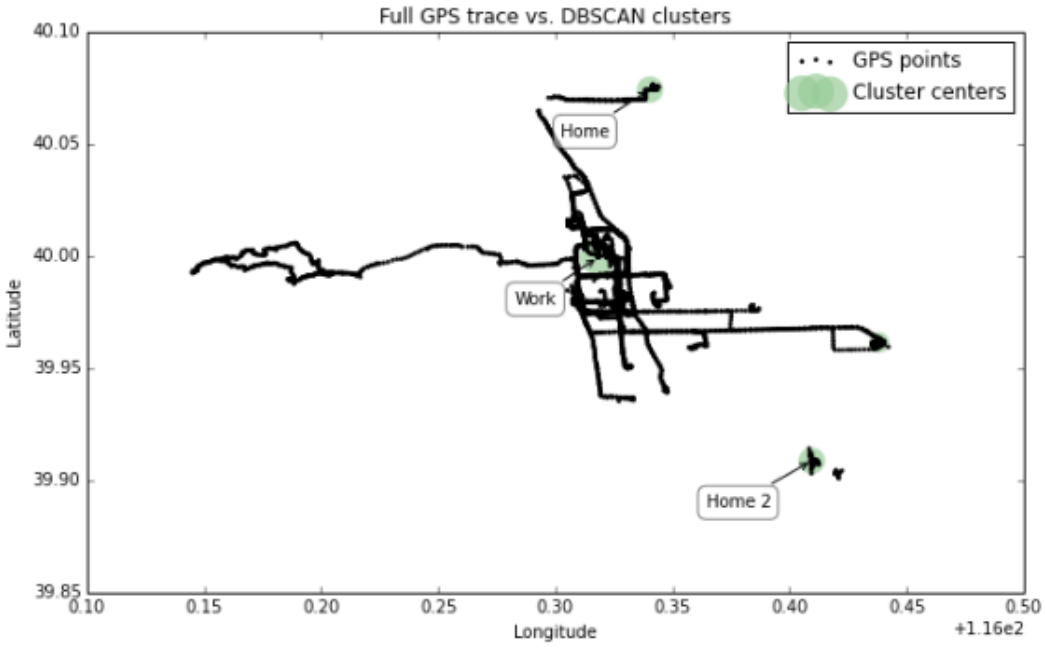

Рис 3.4 Фінальні результати кластеризації

3.3 Додатки, на основі підистеми кластеризації

Розглянемо можливості розширення та покращення підсистеми кластеризації.

Створення мікросервісної архітектури та клієнтського відображення результатів у браузері.

Ця підсистема може буди розгорнута, як мікросервіс з REST API, до котрого можуть слати запити клієнти. Також може буди створений браузерний додаток, що дозволяє додавати нові дані та викликаючи REST API відображати результати у браузері.

Він може мати таку структуру:

1) Користувач обирає пункт в меню “Перевірити дані” і завантажує GPS дані у JSON або plt форматі (такі як широта, довгота, час перебування) в відповідному форматі.
2) Система обробляє запит користувача, на серверну частину відсилається JSON-файл з GPS даними.
3) Відбувається кластеризація отриманих даних на серверній стороні.
4) Результат кластеризації оброблюється системою і відсилається відповідь користувачу у вигляді JSON-файлу.
5) Список кластерів відображається на карті у браузері. Якщо користувач хоче побачити графіки з часом перебування у кластерах він обирає відповідний тип відображення результатів .
6) Для отримання більш детальної інформації, користувач обирає кластер, далі він може проглянути адресу та що знаходиться поряд.

Є можливість створення сімейних акаунтів у цьому сервісі, що допоможе знати, у яких місцях найчастіше знаходиться дитина. Для цього буде потрібна реєстрація через Google Mail, та відкритий доступ до GPS координат, що він збирає.

Алгоритм роботи сімейного акаунту системи знаходження значимих місць:

1) Користувачу необхідно увійти в свій акаунт в системі. Якщо акаунт відсутній, користувачу системи необхідно зарєструватися. Для того, щоб авторизуватися у системі, необхідно ввести ім'я і пароль в відповідну форму (дані введені в форму відправляються на серверну частину у форматі JSON).
2) Система перевіряє коректність введених користувачем даних через спеціально розроблений валідатор. Валідатор видасть помилку на екран пристрою, у разі, якщо користувач намагався ввести заборонені символи, чи перевищив дозволену довжину.

3) В іншому випадку, поля даних трансформуються в відповідну сутність, і виконується пошук в базі даних, який необхідний для того, щоб впевнитися, що акаунт існує у системі.
4) Відповідь з бази даних оброблюється, і повідомлення про результат пошуку виводиться на екран: в разі, якщо пошук видався невдалим (акаунт с такими даними не існує), користувач може побачити повідомлення, що рекомендує йому перевірити правильність введеня даних. В іншому випадку, після повідомлення про успішну авторизацію, користувач опиняється в головному меню додатка.
5) Користувач обирає відповідний пункт в меню, та вводить необхідні поля для того, щоб додати члена своєї сім'ї. Необхідно ввести логін користувача, которого потрібно додати до сімейного акаунту.
6) Всі введені дані відправляються на сервер JSON-файлом, де перевіряються спеціальним валідатором (перевіряються наявність такого користувача у базі даних). Якщо дані введені некоректно, то користувач сповіщується про це, і йому необхідно виправити логін.
7) Якщо дані повністю коректні, то користувач, котрого додали в сімейну групу має підтвердити участь у ній, після цього він відправляє відповідь на сервер JSON-файлом і в відповідній таблиці бази даних створюється новий запис (інформація про зв'язок користувачів зберігається в базі даних), користувач може побачити повідомлення про успішне виконання операції додавання нового члену до сімейної группи.

Ця система може знаходити факти перебування дитини у місцях, що аномальними для неї (для них не створенно окремого кластеру) та повідомляти про це батьків, тоді вони не матимуть обов'язково стежити за місцеперебуванням дитини періодично декілька разів на день, а лише тоді, коли система помітить підозрілі переміщення.

Ця система може реалізована з використанням клієнт-серверної архітектури (рис. 3.5).

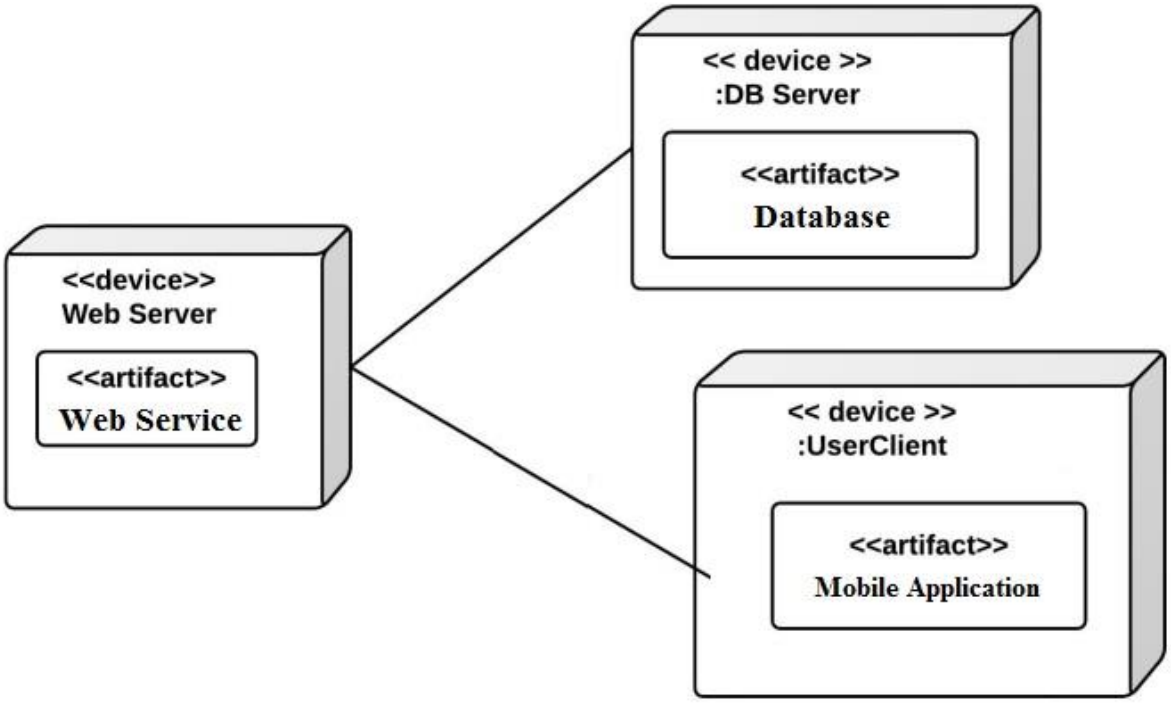

Рис 3.5 Архітектура системи

Клієнтом може бути мобільний пристрії з встановленим додатком. Фактично, додаток є інтерфейсом взаємодії між користувачем і логікою системи.

В системі присутні такі основні сутності:

User

Являє собою узагальненого користувача системою. Фактично, це будь-яка зареєстрована в системі людина. Містить у собі таку інформацію, як повне ім'я і дата народження користувача.

Connection

Зв’язок між користувачами. Зберігає основну інформацію про з’язок, його назву, дату створення.

Participant

Сутність учасника з’язку. Фактично, пов'язує таблиці Connection і User. Необхідна для того, щоб зберігати інформацію про учасників сімейної групи.

Status

Таблиця містить інформацію про можливі статуси користувачів (Safe, Commuting, Warning). Містить тільки назву статусу та ідентифікатор, для того щоб сутність Participant могла посилатися на статус. Один статус може бути у багатьох різних користувачів.

Place

Таблиця містить інформацію про значимі місця користувача. Для кожного користувача створюється нове місце, не дивлячись на те, що у різних користувачів може одні й ті самі значимі місця, в таблиці це будуть різні записи (оскільки координати можуть не суттєво відрізнятися).

Основні вимоги до устаткування:

- Мобільний пристрій на базі Qualcomm-сумісного процесору, починаючи з процесора Snapdragon 501, з тактовою частотою 2.1 гігагерц, рекомендовано — Snapdragon 801 з тактовою частотою 2.3 гігагерц та вище;
- об’єм оперативної пам’яті — 512 Мб, рекомендований об’єм — 2048 Мб та більше;
- мінімальний об’єм вільного дискового простору — 15 мегабайт,

рекомендований об’єм — 100 мегабайт та більш;

- монітор із підтримкою мінімальної роздільної здатності екрану 800 на 600 пікселів; для зручної роботи рекомендується роздільна здатність 1280 на 1024 пікселів та більше.
- Наявність Wi-fi модулю та GPS трекеру в мобільному пристрої

2) Вимоги до програмного забезпечення:

- Веб-сервер з встановленною серверною частиною та база даних
- активне з’єднання з мережею Інтернет.

ВИСНОВКИ

З розвитком технологій, мобільні пристрої зайняли важливе місце в житті сучасної людини. Вони є невід'ємним інструментом для спілкування і роботи. Крім того, вони обладнані множиною сенсорних датчиків, які можна використовувати для різноманітних цілей.

Виходячи з вимог для підсистеми кластеризації, був розроблений один з варіантів реалізації системи кластеризації GPS даних, що буде викладена в відкритий доступ. Розглянені необхідні методи для побудови даної підсистеми кластеризації, способи роботи з картами, способи визначення місцезнаходження. Для того, щоб користувач мав можливість швидкого доступу до кластерів місцеположення, алгоритм кластеризації був чітко налаштований та дані були очищені та видозмінні у зручному форматі.

Для цієї роботи була розроблена підсистема кластеризації та були проаналізовані дані з GPS сенсорів, вбудованих в телефон. Це дало можливість отримувати значні місця людини, без додаткових сенсорів та пристроїв.

Підсистема кластеризації була розроблена для операційних систем UNIX, написана на мові Python.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. GPS Basis [Електронний ресурс]. – 2015. – Режим доступу: https://learn.sparkfun.com/tutorials/gps-basics/all
2. Cluster analysis: Basic concepts and algorithms [Електронний ресурс]. – 2014. – Режим доступу: https://www-users.cs.umn.edu/~kumar/dmbook/ch8.html
3. DBSCAN [Електронний ресурс]. – 2017. – Режим доступу: https://docs.rapidminer.com/studio/operators/modeling/segmentation/dbscan.html
4. Documentation of scikit-learn [Електроний ресурс] – 2016. – Режим доступу: http://scikit-learn.org/stable/documentation.html
5. Intro to Pandas [Електроний ресурс] – 2015. – Режим доступу: https://pythonprogramming.net/python-pandas-data-analysis/
6. Numerical and Scientific Computing with Python [Електронний ресурс] – 2015. – Режим доступу: http://www.python-course.eu/numpy.php
7. Perfomance Evaluation: Ball tree and KD tree [Електронний ресурс] – 2015. – Режим доступу: https://arxiv.org/pdf/1210.6122
8. Programming Python 3rd edition Автор: Марк Луц Издательство: O'Reilly ISBN 978-0596009250; 2006 г.
9. GeoLife GPS Trajectory dataset. User Guide [Електронний ресурс] – 2011. – Режим доступу: https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/