Alma Mater Studiorum · Università di Bologna

Scuola di Ingegneria e Architettura
Campus di Cesena
Corso di Laurea in Ingegneria e Scienze Informatiche

Mining di pattern rilevati da dati di traiettoria

Relatore:
Chiar.mo Prof.
Matteo Golfarelli

Presentata da:
Nicola Narducci

Correlatore:
Dott. Matteo Francia

II Sessione
Anno Accademico 2016/2017

Keywords:

Data Mining

Trajectory

Clustering

A Vladi e Raffa perché questo traguardo è anche vostro.

A Maddi che sei lontana, ma ti pensiamo sempre.

A Michi che stai ancora tracciando la tua strada.

Agli amici di sempre e a quelli incontrati lungo il cammino.

A tutto il resto della famiglia.

No! Provare no! Fare, o non fare! Non c'è provare!

M. Y.

Introduzione

Contesto

Negli ultimi anni si è potuto assistere ad una fortissima diffusione di smartphone dotati di sensori sempre più precisi ed accurati. Tra questi il sensore GPS (*Global Positioning System*) ha permesso di generare una quantità sempre maggiore di informazioni legate alla posizione degli smartphone (e quindi delle persone). Sono tantissime, ad oggi, le applicazioni che sfruttano questi dati, mettendo a disposizione degli utenti che le utilizzano i servizi più diversi. Per nominare un esempio che sicuramente ha accesso ad un pool di utenti enorme, *Google Timeline* [1]. Questa applicazione recupera le posizioni degli utenti dai propri smartphone per poi elaborarli e ottenere i luoghi visitati. Dai luoghi visitati pone poi domande per migliorare le ricerche e affinare i risultati su Google Search[2].

La grande disponibilità di queste informazioni ha permesso di spostare l'interesse dei ricercatori dall'analisi dei dati grezzi alla creazione di elaborate tecniche orientate alla risoluzione di specifiche problematiche. Da qui sono nate nuove possibilità di studiare gli spostamenti individuali e di gruppo per scopi diversi, ad esempio per rendere più efficienti i trasporti cittadini, la caratterizzazione di luoghi e regioni, la scoperta di eventi sociali, la predizione di rotte per evitare il traffico o la caratterizzazione di oggetti in movimento.

Grazie all'evoluzione tecnologica oggi è possibile costruire degli algoritmi che ci permettono di elaborare questa enorme mole di informazioni e ottenere dei risultati

[1] `https://www.google.com/maps/timeline?pb`

[2] `https://www.google.com/`

in tempi utili a risolvere le problematiche sopradescritte, addirittura si può arrivare ad ottenere risultati in rempo reale per alcuni algoritmi con complessità lineare. In questo contesto il contributo si concentra sull'estrapolazione di pattern per la costruzione del **personal gazetteer** (il dizionario dei luoghi importati) a partire da traccie di movimento grezze formate da un elenco di coppie ⟨ posizione, istante ⟩.

Contributo

L'obiettivo principale di questo progetto è quello di sviluppare uno strumento che permetta in primo luogo di elaborare le tracce di movimento grezze raccolte e aggiungere in seguito informazioni utili ai fini delle elaborazioni (*enrichment*). Sui dati così ottenuti si andranno ad applicare gli algoritmi di scoperta (*discovery*) degli *staypoint*, concentrandosi sull'estrapolazione di *punti di fermo* (**staypoint**) che evidenziano *luoghi importanti* per l'utente (o gli utenti) su cui si sta facendo mining, da cui inferire luoghi significativi, con particolare attenzione ai pattern **casa** e **lavoro**. Al fine di presentare i risultati ottenuti sarà inoltre sviluppato un portale web che faciliti l'analisi qualitativa e la visualizzazione dei dati grezzi in modo da poter verificare meglio i risultati che vengono presentati.

Questo contributo trova spazio all'interno del progetto di ricerca "Modelling social behaviours from trajectory" presso il Business Intelligence Group. Si dispone attualmente di due dataset differenti che provengono in parte da dati anonimi e in parte da dati raccolti all'interno del Busines Intelligence Group per poter verificare l'efficacia di quanto sviluppato.

Struttura

L'esposizione del lavoro svolto in questa tesi è suddiviso in quattro capitoli. Il primo capitolo approfondisce la tematica del Trajectory Mining in dettaglio, spiegando i termini che si sono adottati e i lavori precedenti su questo argomento, si spiega il funzionamento del GPS e tratta brevemente le tecnologie esistenti per la memorizzazione dei dati geospaziali. Nel secondo si spiegano gli algoritmi che

sono stati utilizzati per l'elaborazione dei dati, descrivendone il funzionamento. Nel terzo si espone una panoramica sulle tecnologie utilizzate per la realizzazione dell'applicazione, i data set utilizzati e i test effettuati per calcolare gli indici che permettano di capire la bontà degli algoritmi sviluppati. Il quarto capitolo trae conclusioni sul lavoro svolto, proponendo qualche spunto per lavori futuri e discutendo in breve su alcune problematiche che potrebbero emergere.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Trajectory Mining

1.1 I dati e le traiettorie

Per spiegare meglio l'obiettivo della tesi in questo capitolo introduciamo alcuni concetti e termini che sono stati coniati nell'ambito dello studio delle informazioni generate dagli spostamenti.

La **traiettoria** descrive gli spostamenti di un oggetto, può essere quindi definita come il *movimento di un oggetto in uno spazio geografico nell'arco di un periodo di tempo* [22]. La cattura del movimento di un oggetto può essere fatta sfruttando diversi sensori, ad esempio GPS (Global positioning system), RFID (Radio Frequency Identification), WiFi, o GSM (Global System for Mobile Communications). Ognuno di questi presenta caratteristiche differenti tra cui risoluzione e scala. Nella tabella 1.1 sono riportate le caratteristiche e i rispettivi valori per alcune di queste tecnologie [18].

La scelta del tipo di sensore è determinata dalle specifiche necessità del campo di applicazione degli studi, ma negli ultimi tempi la sempre crescente disponibilità di sensori GPS nei dispositivi indossabili e smartphone e al tempo stesso le elevate caratteristiche lo hanno reso la tecnologia più utilizzata nel campo dello studio delle traiettorie.

Tecnologia	**GSM**	**WiFi**	**GPS**
Risoluzione	km	m	m
Scala	Area metropolitana	Campus o complesso di uffici	Globale
Velocità	No	No	Si
Tempo di fermo	Approssimativo	Approssimativo	Esatto
Informazioni di percorso	Approssimative	Approssimative	Esatte

Tabella 1.1: Caratteristiche principali delle tecnologie considerate, anche se GPS e WiFi hanno la stessa risoluzione (m) il WiFi non permette di calcolare la velocità e il tempo di fermo perché il posizionamento ottenuto indica un'area (ad esempio l'ufficio) e non una posizione esatta.

1.1.1 GPS

La tecnologia

Il **GPS** (Sistema di posizionamento globale) è una tecnologia del governo statunitense che permette agli utenti che ne fanno uso di sfruttare servizi di posizionamento, navigazione e *timing* (sincronizzazione). Il sistema è costituito da 3 componenti principali [1]:

- La componente nello *spazio* (Space segment) consiste in una costellazione di satelliti in orbita media intorno alla terra. L'orbita che descrivono è una *medium Earth orbit* (orbita terrestre media) cioè ad un'altezza approssimativa di 20.000 km, orbitando 2 volte al giorno. Gli stati uniti si sono impegnati a mantenere operativi almeno 24 satelliti il 95% del tempo. La costellazione è organizzata su 6 piani orbitali da 4 satelliti ognuno, in modo da assicurare una copertura di almeno 4 satelliti in ogni punto sul pianeta. L'Air Force sta mantenendo operativi 31 satelliti dal 2011[2], questo permette di aumentare

[1] `http://www.gps.gov/systems/gps`
[2] `http://www.gps.gov/systems/gps/space/`

la precisione e la tolleranza ai guasti, ma i satelliti supplementari non sono considerati parte della costellazione primaria.

- La componente di *controllo* (Control segment) è composta da una rete globale di strutture terrestri che tracciano i satelliti, monitorano le trasmissioni, effettuano analisi e inviano comandi alla costellazione. Al momento ci sono 11 stazioni di comando, 16 di monitoraggio e una che coordina le varie stazioni chiamata *Master Control Station* (MCS, stazione di controllo principale) situata in Colorado.

- La componente *utente* (User segment) è aperta ad ogni applicazione che ne voglia fare uso, a partire dai cellulari fino ad arrivare al tracciamento del traffico aereo.

La U.S. Air Force sviluppa e mantiene operativi i segmenti nello spazio e quello di controllo.

Il funzionamento

Il funzionamento del GPS è basato sulla posizione dei satelliti nella costellazione. Ogni satellite è equipaggiato con un orologio atomico molto preciso sincronizzato giornalmente con le stazioni terresti. La posizione dei satelliti è nota con grande precisione e costantemente ognuno trasmette in broadcast le informazioni della propria posizione e dell'orario. I ricevitori GPS possono risalire, se ricevono il segnale di almeno 4 satelliti, tramite formule matematiche alla propria posizione[3]. Per calcolare la distanza da un satellite è sufficiente applicare la formula inversa:

$$\text{spazio} = \text{velocità x tempo}$$

dove *velocità* equivale a c (la velocità della luce), cioè la velocità di propagazione di un segnale radio nel vuoto, mentre il *tempo* è la differenza tra quello indicato nel segnale ricevuto e il momento della ricezione. Una volta conosciuta la distanza rispetto ad almeno 4 satelliti tramite la trigonometria si può determinare la propria posizione sulla Terra.

[3] `https://en.wikipedia.org/wiki/GNSS_positioning_calculation`

L'accuratezza

I satelliti GPS inviano in broadcast il loro segnale con una certa accuratezza URE (User Range Error) sottoposta ad un controllo periodico affinché rimanga entro certi limiti definiti dal GPS Standard Positioning Service Performance Standard[4]. Lato utente l'accuratezza del segnale ricevuto dipende anche dalle condizioni atmosferiche, dagli elementi di disturbo del segnale e dalla qualità costruttiva dell'apparecchio ricevente. Per esempio i moderni sensori inseriti negli smartphone hanno un'accuratezza di circa 5 metri a cielo aperto, ma peggiora velocemente vicino ad edifici, ponti, alberi. Tuttavia tramite l'utilizzo di strumentazioni migliori si può arrivare ad ottenere la posizione con un'accuratezza di pochi millimetri.

1.1.2 Le traiettorie

Indipendentemente dal sensore utilizzato ogni oggetto in movimento genera un flusso di **dati grezzi** (raw data). L'insieme dei dati provenienti da un sensore è una **traccia di movimento**: una sequenza temporale di coppie di ⟨**instante**, **punto spaziale**⟩ corrispondente al campionamento del movimento. Dalla traccia di movimento possiamo estrapolare solo il segmento di movimento a cui siamo interessati. Il segmento identificato da un punto di **inizio** e da uno di **fine** è la traiettoria [24]. Possiamo estrarre più traiettorie rilevanti da una singola traccia di movimento. Una **traiettoria grezza** è una traiettoria estratta da una **traccia di movimento grezza** contenente solo raw data nel suo intervallo.

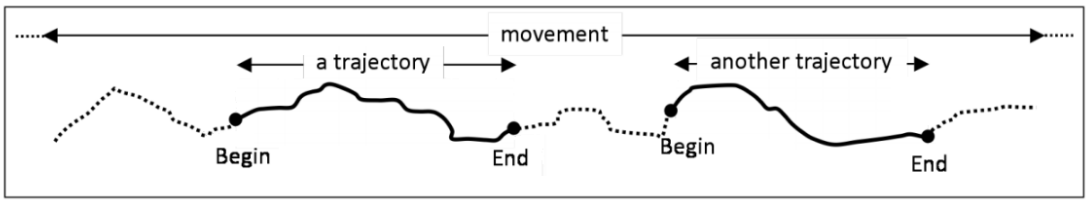


Figura 1.1: Estrazione di più traiettorie da una singola traccia di movimento. Immagine da [22]

[4] http://www.gps.gov/technical/ps/#spsps

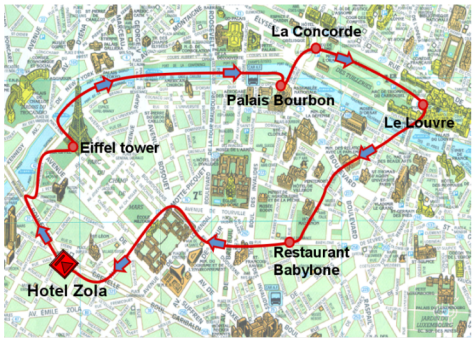


Figura 1.2: Rappresentazione di una traiettoria di movimento su una mappa bidimensionale, le frecce indicano la direzione di movimento. Immagine da [22]

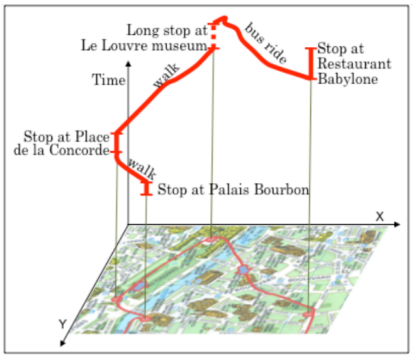


Figura 1.3: La traiettoria della figura precedente rappresentata su 3 assi, l'asse verticale indica il trascorrere del tempo. Immagine da [22]

La traccia di movimento contiene solo punti **campionati**, mentre il movimento è **continuo**. Questo può essere considerata un' *approssimazione* del movimento reale. Possono essere usate funzioni di interpolazione per dedurre la posizione dell'oggetto in movimento tra due punti campionati consecutivamente. Qualche volta può succedere che si abbiano dei **gap** maggiori del solito intervallo temporale che intercorre tra due punti. Questi gap si dividono in due categorie [26]:

- Un **hole** rappresenta la mancanza *accidentale* di informazioni, per esempio in seguito a un malfunzionamento o a una perdita di segnale dovuta a ostacoli come una galleria o l'ingresso in un palazzo.
- Un **semantic gap** rappresenta la mancanza *intenzionale* di informazioni, per esempio in seguito allo spegnimento volontario del sensore GPS da parte del proprietario del dispositivo.

Gli *hole* possono essere eventualmente riempiti con un'interpolazione lineare per computare le posizioni mancanti, i *semantic gaps* invece non vengono riempiti perchè sono deliberatamente mancanti e devono essere registrati come tali.

Le traccie di movimento possono essere estese con altri dati quali la *velocità istantanea* o l'informazione se il punto è fermo o meno (*stillness*), e ancora

l'*accelerazione* o la *direzione*, o altre informazioni utili a semplificarne lo studio. Questo processo è chiamato **arricchimento** (**enrichment**). Aggiungendo dei *repository di dati contestuali* a una **raw trajectory** possiamo trasformarla in una traiettoria semantica (**semantic trajectory**), questo processo è chiamato arricchimento semantico (**semantic enrichment**). La sorgente dei dati può essere ad esempio un database con informazioni geografiche come OpenStreetMap[5], i dati presi da ISTAT[6] o alcune pagine web. I dati che possono essere aggiunti possono essere il mezzo con cui ci si sta spostando (come *bicicletta* o *autobus linea 13*), o il motivo per cui quella traiettoria è stata descritta dall'utente (*viaggio d'affari* o *vacanza*). Le informazioni così aggiunte prendono il nome di **annotazioni** e possono essera aggiunte sia all'intera traiettoria che a sottoparti di essa.

Per dare una definizione formale di traiettoria si può quindi dire che: *Una traiettoria può essere formalmente rappresentata come* $T = \langle p_1 ... p_n \rangle$ *dove* $p_k = (id_k, loc_k, t_k, A_k)$ *è la k-esima posizione,* id_k *è l'identificativo di posizione,* loc_k *è la posizione spaziale,* t_k *è il momento in cui la posizione è stata registrata e* A_k *è una lista, potenzialmente vuota, di annotazioni descrittive* [20].

Un esempio di come questi dati provenienti dalle traiettorie possano essere sfruttati da una ditta di spedizioni che può farci sapere dove si trova un determinato pacco in ogni momento. Ma più frequentemente abbiamo la necessità di esaminare in modo analitico i dati prodotti per scoprire dei **trend**. Ad esempio la traiettoria di un turista che visita una città può essere analizzata per diversi scopi: creazione di profili turistici e suggerimento di tour e servizi personalizzati, regolare il flusso di turisti nelle varie attrazioni, mettere a punto le strutture. Per poter fare delle analisi più profonde sui dati di traiettorie è necessario cominciare ad analizarre le **similitudini** e le **dissimilitudini** tra le traiettorie, raggruppandole e classificandole, identificando i valori anomali ed estraendo le caratteristiche comuni. Un set di caratteristiche determina la creazione di un gruppo di traiettorie.

Un **trajectory behaviour** (comportamento di traiettoria) è un set di caratteristiche che identifica un particolare oggetto in movimento o un gruppo di oggetti.

[5] `http://www.openstreetmap.org`
[6] `http://www.istat.it/it/`

Nell'esempio precedente potremmo dire che la traiettoria presenta un **tourist behaviour** (comportamento da turista). Questo comportamento è definito dai luoghi che visita la persona, mentre invece un esempio di behaviour più generico e **application independent** (indipendente dall'applicazione e quindi dal contesto) è quello del **meet behaviour** (comportamento di "incontro") [10]. Un set di traiettorie mostra questo comportamento se ogni traiettoria del set finisce nello stesso punto approssimativamente nello stesso momento. Il predicato che definisce un behaviour può fare affidamento sulle caratteristiche spazio temporali delle traiettorie: lo **speeding behaviour** è il comportamento che caratterizza gli oggetti che si muovono al di sopra del limite di velocità. Ma può anche dipendere dalle informazioni semantiche che vengono aggiunte alla traiettoria, come per il *tourist behaviour*. Dal momento in cui ogni comportamento è definito da caratteristiche diverse una traiettoria può fare parte di più behaviour contemporaneamente. La stessa traiettoria potrebbe essere parte del set *Speeding* e *Tourist* e allo stesso tempo fare parte del gruppo di traiettorie che mostra il comportamento *Meet*.

Una caratteristica importante nella classificazione delle traiettorie tramite behaviour riguarda la divisione tra i behaviour applicabili ad un singolo oggetto in movimento (**individual behaviour**) oppure applicabili ad un gruppo (**collective behaviour**). Quindi possiamo dire che *Speeding* e *Tourist* sono *individual behaviour* mentre *Meet* è un *collective behaviour*. Un altro collective behaviour conosciuto è il **Flock behaviour** (comportamento di gregge), che indica un gruppo di traiettorie che che non si allontanano l'un l'altra durante una durata minima [4]. Un set di traiettorie mostra il *Flock behaviour* durante un dato intervallo se: ad ogni istante t dell'intervallo c'è un cerchio con un raggio minore di una certa soglia che contiene le posizioni di tutte le traiettorie all'istante t. I comportamenti collettivi possono essere definiti sia per gruppi preesistenti, plotoni di soldati o una mandria di pecore mostrano entrambi un *Flock behaviour*, sia per i gruppi che vengono identificati dalla valutazione del predicato del behaviour.

In mezzo ai comportamenti di gruppo e individuali si posizionano quelli che mostrano peculiarità di entrambe le definizioni, come il **Leader behaviour** (comportamento da leader), l'esempio tipico è il lupo capobranco [1]. Questo comporta-

mento può essere definito come: dato un set S di traiettorie che mostrano un *Flock behaviour*, una traiettoria T parte di S mostra un *Leadership* behaviour se durante un intervallo di tempo ogni volta che il gregge S si muove T è davanti alle altre traiettorie S.

1.2 Trajectory data warehouse

I **Geographic Information System** (Sistema di informazioni geografiche o geospaziali) sono sempre più usati in molti domini applicativi, e di conseguenza sempre più sofisticati **GIS-based Decision Support System** (DSS) integrano funzionalità relative al **SOLAP** (Spatial OLAP). Nei GIS le informazioni spaziali sono memorizzate in *thematic layers* (strati informativi). Le informazioni sono memorizzate attraverso l'utilizzo di strutture dati specifiche. In generale nei GIS le informazioni di ogni layer sono una combinazione di dati puramente spaziali con dati alfanumerici classici. I tipi di modelli attualmente utilizzati sono categorizzabili in due tipologie principali:

- *vector model* (modello vettoriale) è il modello attualmente più usato in GIS [21]. In questo modello un set infinito di punti nello spazio sono rappresentati come una **geometria finita**, ad esempio **punti, linee e poligoni**, quindi i dati di un layer sono rappresentati come un numero finito di tuple ⟨ **geometria, attributi** ⟩.

- *raster model* (modello a trama) in cui lo spazio è campionato in pixel o celle, ognuna delle quali è associata ad un attributo o ad un set di attributi, queste celle formano una griglia organizzata in *zone* in cui ogni cella di una determinata zona ha gli stessi valori per determinati attributi.

Solitamente le informazioni dei vari layer sono sovrapposte o unite. Le query che necessitano della sovrapposizione sono più difficili da computare nei modelli vettoriali che in quelli raster [15]. D'altra parte i modelli vettoriali offrono una rappresentazione concisa delle informazioni, mentre i modelli raster possono essere

rappresentati come un caso particolare dei modelli vettoriali considerando ogni cella come un poligono.

L'importanza della *data analysis* (analisi dei dati) da parte delle aziende e organizzazioni di tutti i settori è aumentata significativamente negli ultimi anni per permettere loro di mantenere un vantaggio competitivo nel loro processo decisionale. **OLAP**(On Line Analytical Processing) [16] comprende un set di strumenti e algoritmi per permettere l'analisi di enormi quantità di dati in modo efficiente. Questi database sono solitamente di sola lettura (l'aggiornamento viene effettuato off-line), e sono chiamati *data warehouse*. I sistemi OLAP sono basati su *modelli multidimensionali* chiamati *data cube* [17]. In questo cubo multidimensionale ogni cella contiene valori o set di valori aggregati degli attributi di interesse.

La necessità di analizzare i dati attraverso i *GIS-based Decision Support System* con OLAP ha permesso di sviluppare strumenti molto sofisticati, e query aggregate su dati spaziali sono centrali su GIS DSS. Una query OLAP classica potrebbe essere "Vendite totali di macchine in Italia" e l'aggregazione con dati spaziali deve essere efficientemente supportata trasformando la query precedente in una query SOLAP del tipo "Vendite totali nelle città attraversate dal fiume Po ed entro un raggio di 100km da Torino". I risultati provenienti dalle query SOLAP possono essere ulteriormente rappresentati attraverso le operazioni tipiche per la navigazione dei risultati che implicano l'aggregazione (*roll up*) o la de-aggregazione (*drill down*). Il concetto di *SOLAP* è stato introdotto con lo scopo di esplorare i dati spaziali filtrati tramite mappe [23] in modo analogo a quanto avviene normalmente in OLAP con tabelle e grafici.

1.3 I problemi applicativi ed Estrazione di feature da dati grezzi

Nel tempo sono diversi i problemi applicativi a cui si è trovata una soluzione attraverso il trajectory mining. Possiamo suddividerli in sette categorie principali [20], ognuna delle quali si pone un obiettivo la caratterizzazione di oggetti e luoghi o la scoperta di legami:

1.3.1 Caratterizzazione di oggetti in movimento

L'obiettivo è creare una struttura di informazioni in grado di descrivere il movimento e quindi di inferire le **attività** degli umani o degli animali determinando un **profilo di mobilità**. C'è grande interesse infatti nel comprendere il comportamento degli individui attraverso le attività che intraprendono oppure il *mezzo di trasporto* utilizzato per muoversi. Un approccio comune in questo tipo di problemi è la scoperta di **luoghi frequentemente visitati** attraverso il **sequential pattern mining**. Un approccio simile viene utilizzato anche per determinare i *pattern di movimento animali*, ad esempio cercando di individuare i flussi migratori degli stormi di cicogne.

Obiettivo	Tecniche utilizzate
Inferenza di attività e mezzi di trasporto	Estrazione di feature caratteristiche e categorizzazione
Profiling di mobilità umana, scoperta di movimenti di animali	Estrazione di sequenze di luoghi frequentemente visitati

Tabella 1.2: Obiettivi di Caratterizzazione di oggetti in movimento

1.3.2 Scoperta di relazioni sociali

L'obiettivo di questo problema applicativo è quello di scoprire l'esistenza di *relazioni* tra diversi oggetti in movimento, ad esempio i **legami** tra due individui o l'**interazione** tra animali diversi come un predatore e una preda. Il grande successo dei social media ha incoraggiato le persone nel condividere luoghi e posizioni. Alcune ricerche basano la scoperta di relazioni sociali attraverso la compresenza (l'essere nello stesso posto simultaneamente) [7], oppure sulle visite allo stesso luogo [12]. Infatti attraverso modelli probabilistici si può capire che se due persone sono spesso insieme nello stesso luogo quasi certamente c'è un legame di qualche tipo tra di loro.

Obiettivo	Tecniche utilizzate
Scoperta di legami sociali tra due individui, scoperta della comunità	Raggruppamento sulla base di stay location individuali Estrazione di sequenze temporalmente ordinate
Scoperta di interazione tra gli animali	Ricerca di gruppi di animali in movimento e cambiamento della struttura del gruppo nel tempo

Tabella 1.3: Obiettivi di Scoperta di relazioni sociali

1.3.3 Caratterizzazione di luoghi e regioni

Lo scopo di questo application problem è la classificazione di zone e l'assegnamento di **funzioni socio-economiche**, la scoperta di *hotspots* in città o la scoperta di habitat di particolari animali. Ad esempio applicando degli algoritmi di clustering per derivare il *tipo di utilizzo* che viene fatto di un terreno o attraverso lo studio delle traiettorie dei taxi capire quali sono i luoghi in cui avvengono i *pick-up* e i *drop-off* (carico e scarico dei passeggeri) e la loro *distribuzione temporale*. Un'altra istanza di questo problema sono le congestioni del traffico, che possono essere trovate tramite lo studio dei movimenti di gruppo (*flock patterns*).

1.3.4 Caratterizzazione di connessioni tra luoghi o regioni

Queste connessioni possono essere scoperte mediante lo studio degli oggetti che si muovono attraverso le regioni interessate, per scoprire la quantità delle connessioni e la loro intensità. Uno sviluppo ulteriore di questa categoria riguarda la **mobilità umana**. Ad esempio alcuni studi [19] si concentrano sull'analisi di traccie GPS prodotte dai taxi per scoprire le connessioni attraverso luoghi differenti della stessa città in giorni diversi.

Obiettivo	Tecniche utilizzate
Scoperta del tipo di utilizzo di terra o funzioni economico-sociali	Estrazione di regioni Estrazione di feature caratteristiche e caratterizzazione
Scoperta di hotspots	Raggruppamento basato su caratteristiche spaziali o temporali dei dati
Rilevamento di ingorghi di traffico	Ricerca di gruppi di macchine in movimento, filtro basato sulla velocità dei gruppi

Tabella 1.4: Obiettivi di Caratterizzazione di luoghi e regioni

Obiettivo	Tecniche utilizzate
Scoperta di connessioni, stima dell'intensità della connessione	Mapping origine-destinazione, statistiche di derivazione dei collegamenti, raggruppamento Separazione dei collegamenti normali da quelli anomali

Tabella 1.5: Obiettivi di Caratterizzazione di connessioni tra luoghi o regioni

1.3.5 Caratterizzazione di scoperta e riconoscimento di eventi sociali

Analizzando le traiettorie di oggetti in movimento è anche possibile riconoscere l'esistenza di **luoghi di raccolta**. Questi metodi sono utilizzati per scoprire luoghi con un'alta concentrazione di oggetti (e quindi *persone*) in uno specifico intervallo di tempo, tramite il riconoscimento dei dispositivi stazionari che si stavano inizialmente muovendo. Una volta che l'evento è stato rilevato analizzando la distribuzione dei partecipanti, la forma, il luogo e la densità di persone ad un determinato evento è possibile capire il **tipo di evento** che sta le raccogliendo. Uno studio che si muove in questa direzione è quello di Calabrese [6] che analizza l'origine dei partecipanti

ad un evento dai dati di mobilità ricavati dai cellulari per riconoscere il tipo di evento.

Obiettivo	Tecniche utilizzate
Scoperta di eventi sociali	Raggruppamento basato su proprietà spazio temporali
Riconoscimento degli eventi sociali scoperti	Estrazione di feature cararistiche e categorizzazione

Tabella 1.6: Obiettivi di Caratterizzazione di scoperta e riconoscimento di eventi sociali

1.3.6 Predizione di traiettorie

Questo problema mira ad analizzare le traiettorie di oggetti in movimento per poter fare predizioni sulle prossime posizione degli oggetti stessi, la loro destinazione, il percorso intrapreso. I metodi più comuni utilizzano i modelli di *Markov* [2] o il *movement pattern mining*. Alcuni lavori mirano a predirre le congestioni dovute al traffico predicendo dove il traffico stesso si sposterà utilizzato una struttura ad albero di T-Patterns [13].

Obiettivo	Tecniche utilizzate
Predizione di luogo e di percorso	Costruzione di un modello probabilistico comparandolo con la traiettoria attuale Estrazione di sequenze ordinate di luoghi visitati dalla cronologia comparandoli con quelli della traiettoria attuale
Predizione di eventi (ingorghi di traffico)	Estrazione di densità delle traiettorie e verifica della variazioni nel tempo

Tabella 1.7: Obiettivi di Predizione di traiettorie

1.3.7 Raccomandazioni basate sulle traiettorie

Questo tipo di raccomandazioni viene spesso effettuato tramite il confronto di cronologie di mobilità tra utenti con profili simili, l'assunzione dietro questo meccanismo è che persone con un comportamento di mobilità (*mobility behaviour*) simile tenderanno ad avere interessi e preferenze in comune, quindi potrebbero essere potenzialmente amici o fare visita alla stessa tipologia di luoghi. Diventa quindi possibile sia da un lato *consigliare* amici ad un utente, sia consigliare luoghi da visitare.

Obiettivo	Tecniche utilizzate
Raccomandazione di luoghi	Estrazione di sequenze di luoghi visitati, ordinati per popolarità e selezionando i primi
Raccomandazione di percorso	Estrazione di sequenze di luoghi visitati, ordinati per similitudine a quelli che visita l'utente in questione e selezionando i primi luoghi
Raccomandazione di amici	Estrazione di sequenze di luoghi visitati, ordinati per similitudine a quelli che visita l'utente in questione e selezionando gli utenti che li hanno visitati
Raccomandazione di viaggio	Estrazione di sequenze di luoghi visitati, estraendo le preferenze di viaggio in termini di tempo trascorso nei luoghi

Tabella 1.8: Obiettivi di Raccomandazioni basate sulle traiettorie

1.4 Gli ambiti applicativi

La conoscenza generata attraverso il trajectory data mining è utile in diverse aree applicative, infatti la soluzione ad un problema applicativo può essere usato

in ambiti differenti. I principali ambiti in cui ad oggi vengono sfruttate le soluzioni ai problemi illustrati precedentemente sono:

1.4.1 Trasporti

Nel settore dei trasporti è particolarmente importante scoprire le caratteristiche di città e regioni, in modo da capire dove si concentrano partenze e arrivi e le caratteristiche degli oggetti, per sapere dove si muovono in grande numero e dove si fermano. Queste problematiche sono strettamente legate alla *caratterizzazione di luoghi e regioni* (sezione 1.3.3) e *caratterizzazione di oggetti in movimento* (sezione 1.3.1).

1.4.2 Pianificazione territoriale

La disciplina che regola l'utilizzo del territorio e delle attività umane che vengono svolte su di esso ha la necissità di conoscere le interazioni tra le varie zone e regioni e le varie carattaristiche per accelerarne lo sviluppo. Questo in passato poteva venire svolto attraverso delle indagini laboriose, mentre ora è possibile identificare le necessità attraverso il trajectory mining. In particolare i problemi che vengono affrontati in questo settore sono la *caratterizzazione di luoghi e regioni* (sezione 1.3.3) e la *caratterizzazione di connessioni tra luoghi o regioni* (sezione 1.3.4).

1.4.3 Ambiente

Nella lotta all'inquinamento ambientale è importante che i livelli di inquinamento siano quantificati in posizioni differenti e i dati da varie fonti integrati per ottenere risultati precisi. Questo si può tradurre come problema di *caratterizzare luoghi e regioni* (sezione 1.3.3) facendo mining su traiettorie associate agli aspetti di inquinamento dell'aria e inquinamento acustico.

1.4.4 Energia

Il settore energetico spesso si occupa di determinare i pattern di consumo di una specifica zona o degli individui. Questi interessi diventano la *caratterizzazione di luoghi e regioni* (sezione 1.3.3) e *caratterizzazione di oggetti in movimento* (sezione 1.3.1).

1.4.5 Applicazioni sociali

Il trajectory mining in questo caso può essere utilizzato per la scoperta del comportamento degli individui e comparare i profili così creati per diversi tipi di raccomandazioni. Ad esempio dopo aver scoperto le similitudini tra profili differenti è possibile raccomandare le amicizie o i luoghi da visitare oppure ancora i percorsi da seguire. Altrimenti si possono sfruttare i profili per comprendere lo stile di vita dei soggetti che sono caso di studio. Tutto questo si traduce come *caratterizzazione di oggetti in movimento* (sezione 1.3.1), *scoperta di relazioni sociali* (sezione 1.3.2) e *raccomandazioni basate sulle traiettorie* (sezione 1.3.7).

1.4.6 Business

Nel settore business spesso si ha la necessità di calcolare il numero poteziale di visite ottinibili in uno specifico luogo, e i pattern di mobilità degli utenti in questi luoghi. Il trajectory data mining in questo caso viene utilizzato per *caratterizzazione di oggetti in movimento* (sezione 1.3.1), *caratterizzazione di luoghi e regioni* (sezione 1.3.3) e *raccomandazioni basate sulle traiettorie* (sezione 1.3.7). Esempi tipici di questo settore sono trovare il luogo migliore per aprire un'attività, il luogo migliore in cui mettere una pubblicità oppure il layout migliore per un negozio.

1.4.7 Sicurezza e ordine pubblico

Un importantissimo campo in cui viene utilizzato il trajectory mining è il settore della sicurezza e ordine pubblico. Infatti identificando e monitorando oggetti in movimento con un alto potenziale di essere causa o obiettivo di una

minaccia è possibile scoprire per tempo o addirittura prevenire queste minaccie. Gli application problem coinvolti sono *caratterizzazione di oggetti in movimento* (sezione 1.3.1), *caratterizzazione di luoghi e regioni* (sezione 1.3.3) e *predizioni di traiettorie* (sezione 1.3.6). Ad esempio alcuni lavori in questo campo sono stati la rilevazione di comportamenti sbagliati [8], monitorare grossi eventi e i comportamenti della folla [27], verificare il movimento degli uragani [9], e predirre le loro future formazioni [5].

1.4.8 Ecologia

Le ricerche che riguardano gli animali sono spesso orientate al determinare il comportamento degli animali individuali, le loro interazioni e l'uso dell'habitat. Questo è risolto tramite gli application problem di *caratterizzazione di oggetti in movimento* (sezione 1.3.1) e la *scoperta di relazioni sociali* (sezione 1.3.2).

1.4.9 Sport

Nello sport (con molto movimento) spesso ci si concentra sulla dinamica dei movimenti e quindi sui *mobility pattern* dei giocatori e sulle loro interazioni. Il trajectory data mining in questo caso viene utilizzato per *caratterizzazione di oggetti in movimento* (sezione 1.3.1) e la *scoperta delle loro interazioni* (sezione 1.3.2).

Capitolo 2

Estrazione di StayPoints

Per l'elaborazione degli staypoints si è deciso di implementare due algoritmi differenti paragonando i risultati ottenuti quando possibile. Prima di spiegare i due algoritmi è doveroso fare una puntualizzazione riguardo ad alcuni termini che nel linguaggio comune sono interpretati in modo errato, **precisione** ed **accuratezza**:

- la **precisione** è legata a un fattore di **ripetitibilità** della misura, cioè se ad uno stesso valore reale ottengo sempre la stessa misurazione significa che ho una precisione elevata, quindi la precisione indica la **varianza** delle misurazioni di uno stesso valore. Una misura precisa oscilla poco intorno ad un valore che si discosta dal valore reale di una quantità chiamata "**errore sistematico**"
- l'accuratezza è lo scostamento della misura dal valore reale. Una misura accurata oscilla intorno al valore reale.

La figura 2.1 mostra con quattro esempi le definizioni appena date di accuratezza e precisione, mentre la figura 2.2 mostra il l'errore sistematico comparato con il valore reale.

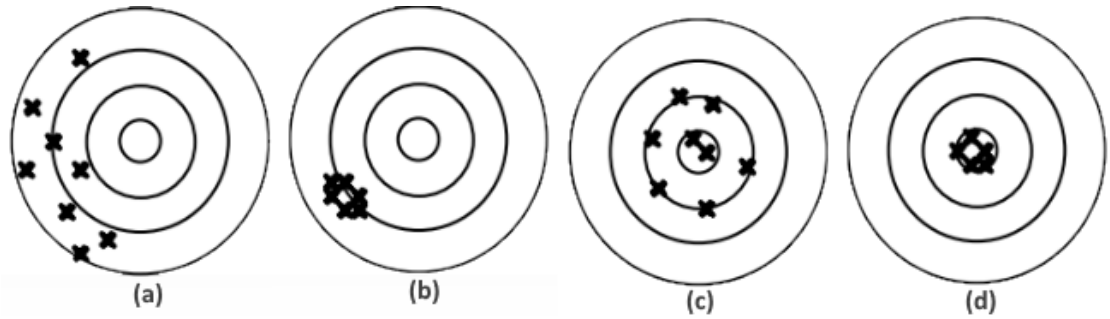


Figura 2.1: La figura mostra la differenza tra precisione e accuratezza: (a) Misura imprecisa e inaccurata, (b) Misura precisa e inaccurata, (c) Misura imprecisa e accurata, (d) Misura precisa e accurata

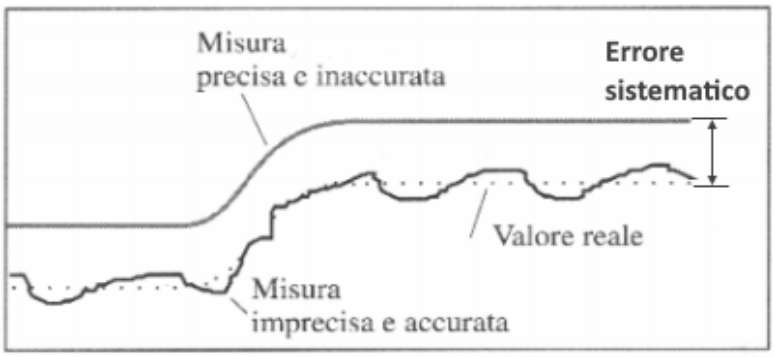


Figura 2.2: Differenza tra misura precisa e misura accurata, con particolare attenzione alla differenza con il valore reale

2.1 Curve Extrema

Il Problema

Il primo obiettivo è estrapolare gli staypoints da una traiettoria, la definizione formale del problema può essere posta in questo modo:

- **Obiettivo**: identificare gli staypoint di un utente data una determinata traiettoria
- **Input**: Traiettoria dell'utente rappresentata come set di posizioni
  S= $\{loc_i=(t_i, l_i)|i = 1...n\}$
  con l_i = (latitudine, longitudine) e t_i = timestamp
- **Output**: Sequenza di staypoints S_i che rappresentano le posizioni in cui si è trascorso un certo ammontare di tempo

Il primo algoritmo scelto è Curve Extrema [25], che si dimostra efficace nell'estrapolare sequenze di staypoints in tempo lineare. Dalla definizione di staypoint come *luogo in cui una persona si ferma per un certo ammontare di tempo* [25] si deduce che il funzionamento di questo algoritmo si basa sul **tempo trascorso** in una determinata posizione. Uno staypoint potrebbe quindi essere il luogo di lavoro o quello in cui si abita, oppure un ristorante in cui si è cenato.

Nella maggior parte dei lavori precedenti a quello del Curve Extrema [25] quando si procede alla ricerca degli staypoints la problematica principale era calcolare la distanza massima alla quale due posizioni potessero essere considerate sovrapposte e dopo quanto tempo si poteva considerare un luogo come staypoint. Con questo algoritmo si procede con la trasformazione della traiettoria di interesse in una **curva spaziale bidimensionale**. In questo modo si trasforma la problematica di trovare le soglie adeguate per i parametri di input all'individuare dei **massimi e dei minimi** nella curva spaziale (*Extrema extraction*).

Come menzionato precedentemente per cambiare il tipo di approccio al problema si procede con una trasformazione dell'input da una traiettoria ad una curva spaziale. La *curva spaziale* è una funzione monotona crescente composta di punti p_i = (x_i, y_i) dove x_i indica il tempo trascorso dall'inizio della traiettoria e y_i lo spazio

totale percorso. Questa funzione è rappresentabile con una curva bidimensionale come mostrato in figura 2.3. La trasformazione si ottiene attraverso una mappa

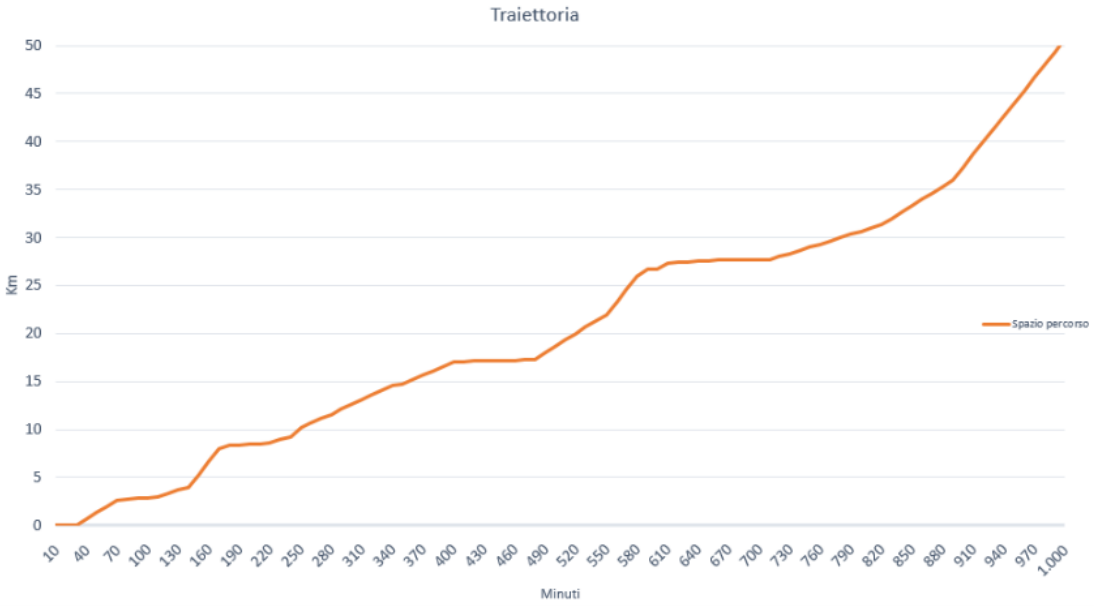


Figura 2.3: Esempio di trasformazione di una traiettoria in una curva spaziale

biunivoca M composta da due funzioni $f(t_i)$ e $g(l_i)$.

$$\mathbf{M}\text{: } (t_i, l_i\ ) \rightarrow (x_i, y_i)$$

$$\mathbf{f}\text{: } t_i \rightarrow x_i$$

$$\mathbf{f}(t_i) = t_i \text{ - timestampIniziale}$$

$$\mathbf{g}\text{: } l_i \rightarrow y_i$$

$$\mathbf{g}(l_i) = \sum_{i=2}^{n} distanza(l_i, l_{i\text{-}1}),\ g(l_1) = 0$$

Effettuata la trasformazione a curva spaziale si procede con l'identificazione dei minimi (**Extrema extraction**) infatti ricordando che la curva è monotona crescente è possibile ricondurre ogni staypoint a *un minimo locale della derivata prima della curva spaziale*, ovvero un momento nel quale lo spazio percorso non aumenta. La derivata prima della curva spaziale:

$$C'(x,y) = \Delta\ y\ /\ \Delta\ x$$

è anche la velocità di movimento. Quindi idealmente se un utente smette di muoversi allora C'(x,y) = 0. Graficando la derivata prima è possibili capire a colpo d'occhio dove l'utente era fermo (o prossimo a fermarsi). La figura 2.4 mostra la derivata prima della curva spaziale precedente. I segmenti orizzontali (o quasi orizzontali) indicano i punti in cui l'accelerazione è nulla, mentre i segmenti prossimi al valore zero indicano i punti di fermo.

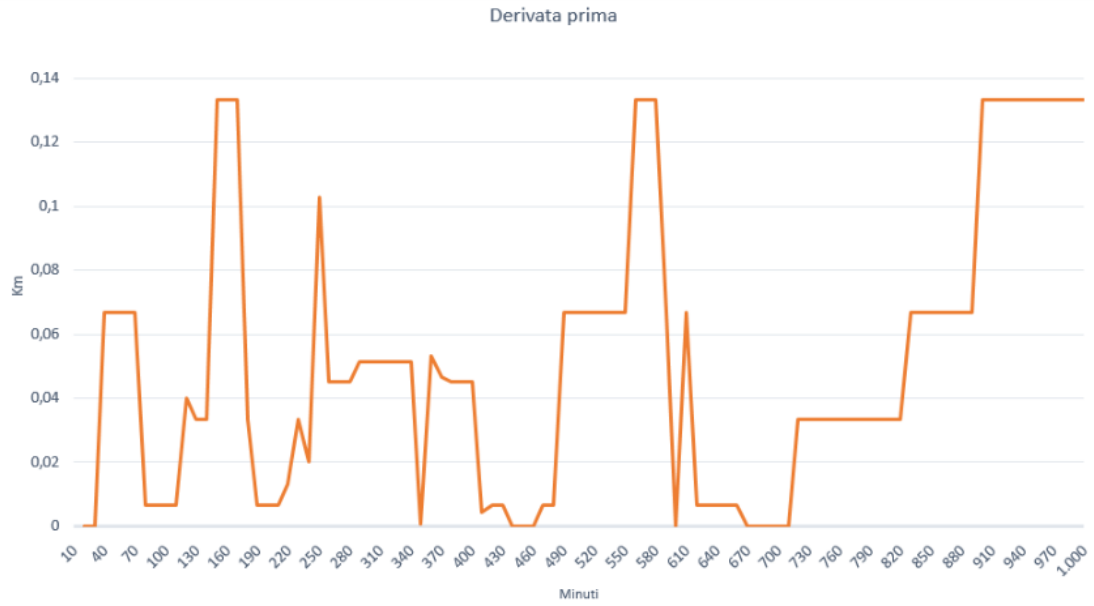


Figura 2.4: Derivata prima della curva spaziale 2.3

Dal punto di vista grafico è possibile ricavare questi punti di minimo locale derivando ancora una volta e ottenendo la derivata seconda,

$$C''(x,y) = \Delta^2\ y\ /\ \Delta^2\ x$$

i minimi locali saranno in corrispondenza di C"(x,y) = 0. La figura 2.5 mostra la derivata seconda della curva spaziale precedente. Nella realtà però raramente si otterranno dei dati così precisi, quindi per identificare i minimi si utilizza un approccio semplificato, cioè identificarli in corrispondenza di un cambio di segno della derivata seconda (o dove interseca lo zero). Dove la derivata seconda passa da un valore positivo a uno negativo si intuisce che l'accelerazione (C") sta diminuendo

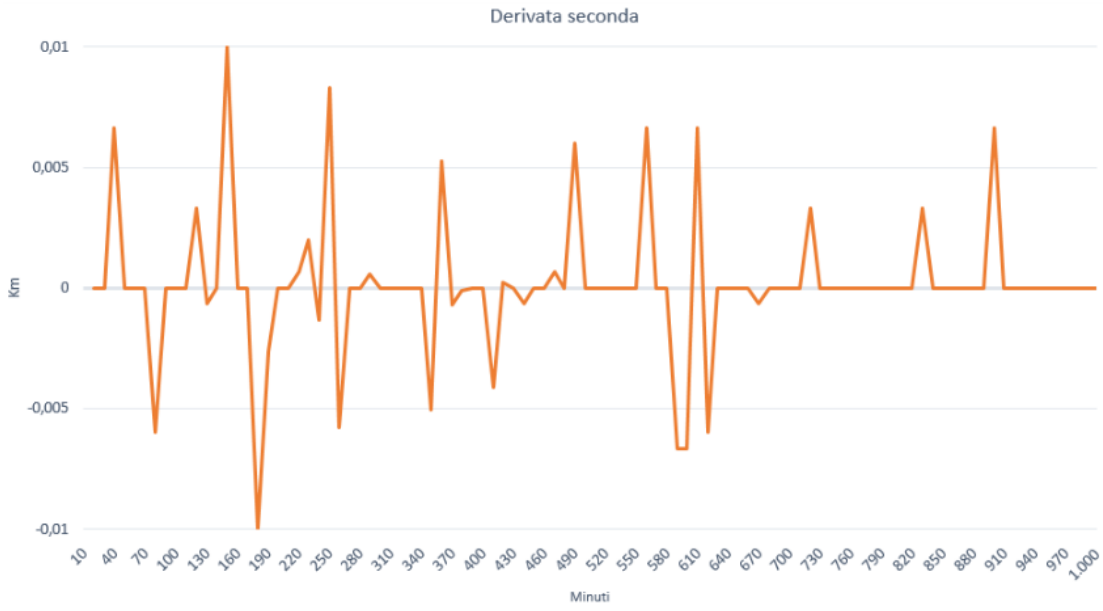


Figura 2.5: Derivata seconda della curva spaziale precedente 2.3

fino a fermasi, dove invece passa da un valore negativo a uno positivo vuol dire che sta aumentando e quindi riprende il movimento. Per fare in modo di estrarre da questi minimi locali solo quelli in cui l'utente è fermo basta applicare un filtro per cui la derivata prima sia sufficientemente vicina a zero $\mathbf{C'(x,y) \approx 0}$.

Al termine dell'algoritmo quello che si otterrà è una sequenza ordinata di luoghi visitati, con annotazioni temporali (**T-Patterns**). I T-Pattern sono *un set di traiettorie individuali che condividono la proprietà di visitare la stessa sequenza di luoghi con tempi di viaggio simili* [14]. I T-Pattern combinano le notazioni sequenziali spaziotemporali, che indicano in che sequenza i luoghi vengono visitati, con le sequenze temporalmente annotate (Temporary Annotated Sequence) che invece indicano il tempo di transizione tra le posizioni. Nella figura 2.6 sono mostrate le tre differenti notazioni.

La figura 2.7 mostra un esempio di T-Pattern di una tipica giornata feriale di un utente.

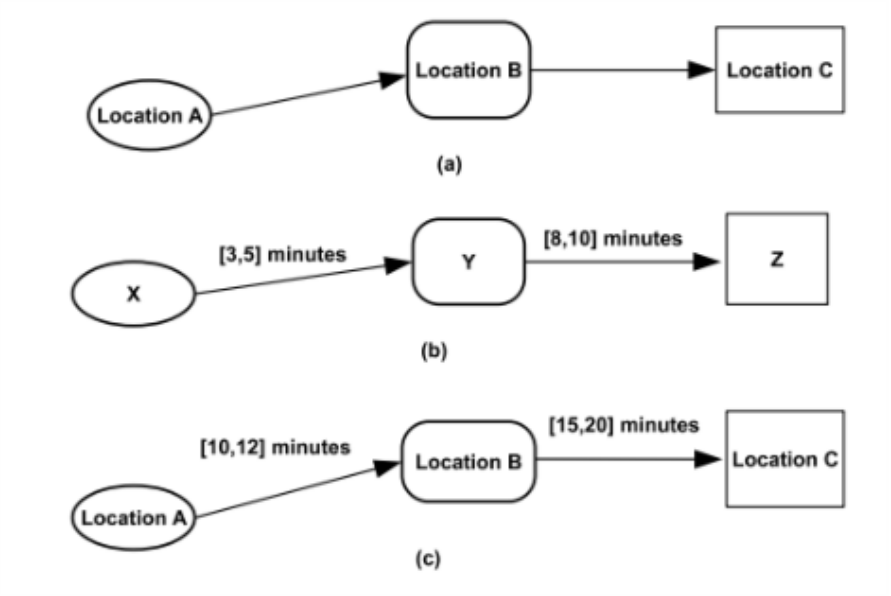


Figura 2.6: (a) Pattern di sequenza spazio temporale, indica i luoghi visitati e in che sequenza. (b) Sequenza temporalmente annotata (TAS), indica una sequenza generica di eventi, in questa notazione non ci sono informazioni spaziali. (c) T-Pattern in cui sono combinate le informazioni di (a) e (b), indica sia la sequenza dei luoghi visitati ma anche il tempo di transizione tra un luogo e l'altro. Immagine da [20].

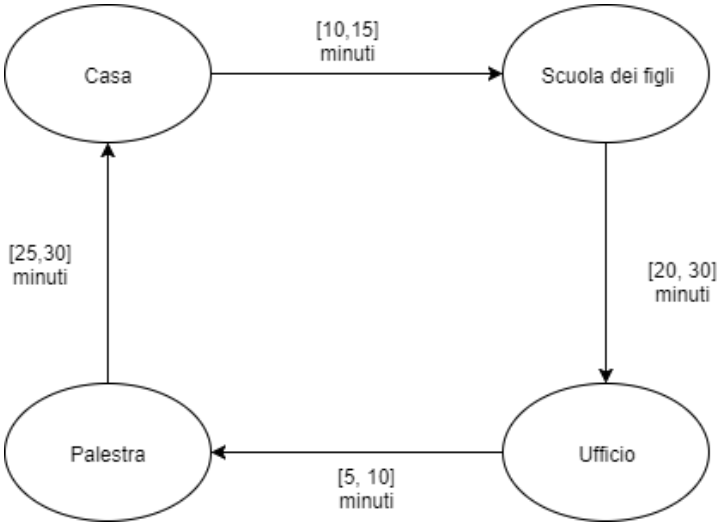


Figura 2.7: Esempio di 'Lunedì' di un utente, questo T-Pattern mostra la sequenza di luoghi che l'utente visita in un giorno e i tempi di transizione tra un luogo e l'altro.

2.1.1 Parametri di ingresso per l'elaborazione

I parametri di ingresso dell'algoritmo Curve Extrema sono due:

- speedthreshold $\longrightarrow$ velocità massima per considerare un punto come fermo (C'(x,y) $\approx$ 0)

- traiettoria $\longrightarrow$ su cui effettuare la ricerca degli staypoints

La traiettoria su cui effettuare l'elaborazione è estratta a partire da un dato elenco di posizioni di un utente ordinati per timestamp applicando un filtro temporale **f(min, max)** come indicato precedentemente (1.1), il filtro può essere costruito a partire da fascia oraria, giorno della settimana, etc.

2.1.2 Preprocessing

La fase di preprocessing è necessaria perché i dati sono solitamente sporcati da molto **rumore**, rendendo più difficile individuare gli staypoints. Quindi questa fase

avviene prima dell'effettiva elaborazione della traiettoria da parte dell'algoritmo. Il problema principale che si è andato ad affrontare è stato quello della **bassa precisione** dei segnali GPS. Per fare un esempio una traiettoria ricavata dagli orari notturni (quindi presumendo che l'utente dorma) anche rimanendo fermo in una determinata posizione, il valore rilevato oscilla in un intorno di diversi metri. La curva spaziale derivata da questi dati risulta essere una curva in cui lo spazio percorso è elevato (nell'ordine delle centinaia di metri), quando in realtà la posizione iniziale e quella finale non sono più lontane di qualche metro e tutte quelle intermedie possono essere contenute in un raggio relativamente piccolo (una decina di metri o poco più). La figura 2.8 (b) mostra 3 traiettorie notturne (rappresentate in colori diversi), sull'asse delle ascisse si trova il tempo (in secondi) e sulle ordinate lo spazio percorso calcolato con la formula $\mathbf{g}(l_i)$ (2.1) in metri.

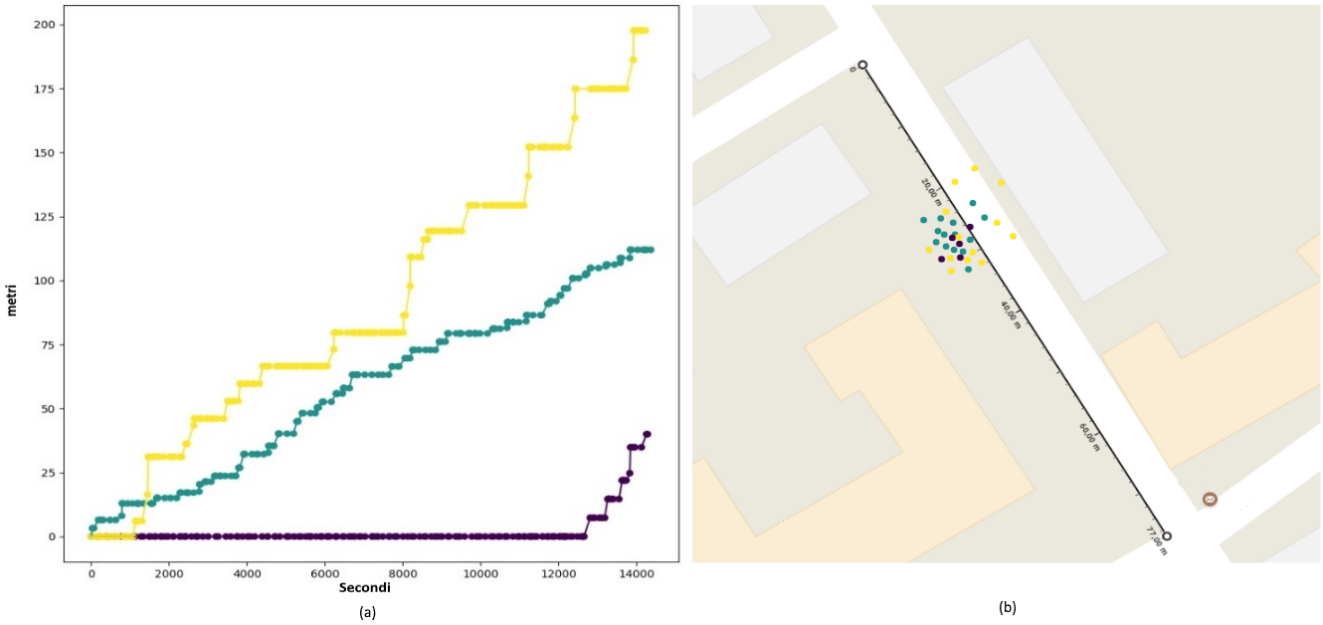


Figura 2.8: Curva spaziale di 3 traiettorie notturne (a) e relativa distribuzione geografica (b).

La figura 2.8 (a) mostra la distribuzione spaziale delle posizioni che compongono le tre traiettorie e si può notare che sono riconducibili a errori di precisione. Si nota subito che la traiettoria gialla in circa 4 ore percorre uno spazio totale di 200metri

sebbene i punti siano al massimo a 20 metri di distanza, le altre due traiettorie hanno una distribuzione geografica più compatta (hanno un valore con precisione maggiore) e quindi risulta percorrano uno spazio minore.

Per diminuire il rumore si è deciso di applicare una procedura di normalizzazione che permetta di diminuire il rumore e aumentare la precisione. La procedura ha due parametri di input:

- una soglia di distanza *distThr*
- una finestra di punti (in percentuale) che andrà a definire il numero minimo di punti di cui deve essere composta la traiettoria di output *windowSize*.

La traiettoria è elaborata in questo modo: si è deciso di campionare all'interno delle singole finestre con dimensione **windowSize**, se la distanza tra i punti è minore della soglia **distThr** si calcola il **centroide** della finestra. In questo modo si riesce a diminuire il rumore (aumentando la precisione). Il numero di punti totali diminuisce, sostituiti da alcuni punti fittizi (che corrispondono ai centroidi) che però sono una media dei punti originali e quindi ai fini dell'algoritmo hanno un significato analogo.

In figura 2.9 vengono mostrate le tre traiettorie analizzate in precedenza, in questo caso però è stato applicato il processo di "**normalizzazione**" appena discusso. È possibile osservare che lo spostamento massimo è ridotto di molto, infatti si passa da uno spostamento di 200m a uno di 30m, diminuendo in questo caso l'errore dovuto al tilting (oscillazione) del valore letto dal sensore dell'85% calcolato con la seguente formula:

$$\text{Diminuzione spostamento} = 100\% - \frac{S_{\text{iniziale}} - S_{\text{finale}}}{S_{\text{iniziale}}}\ \%$$

Questi risultati sono stati ottenuti dopo alcuni passaggi di tuning, alla fine dei quali si è deciso di adottare come valori di **distThr** e **windowSize** rispettivamente di 20m e 5%, decisi a partire dai valori di precisione riportati dai dati GPS di origine.

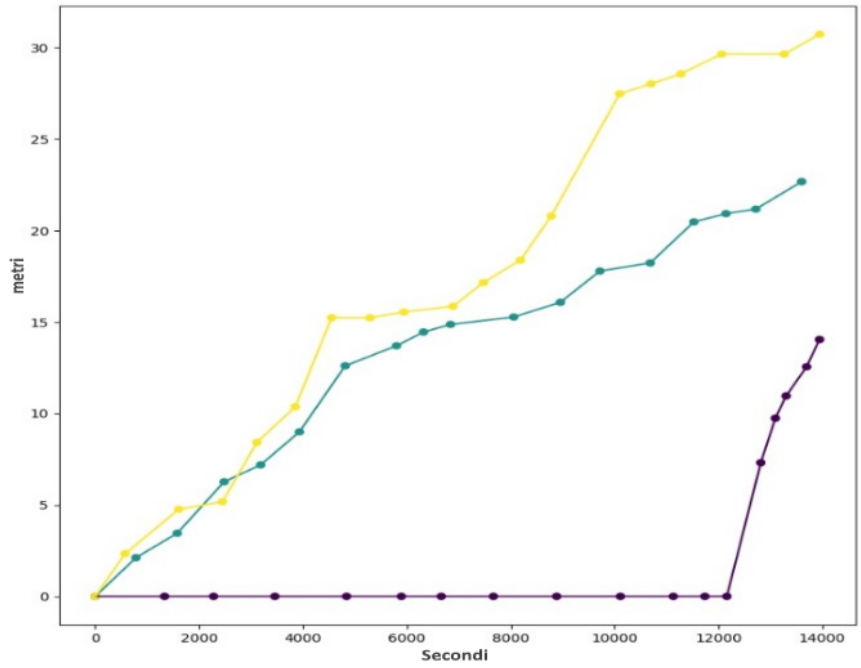


Figura 2.9: Curva spaziale delle traiettorie mostrate in figura 2.8 dopo che è stato applicata la funzione di normalizzazione.

2.1.3 Elaborazione

A questo punto sulle traiettorie normalizzate è possibile applicare normalmente l'algoritmo, di seguito nel pannello 1 viene presentato lo *pseudo codice* delle operazioni svolte per ottenere la sequenza di **staypoints**. Il risultato della procedura sarà un'elenco di **staypoints**, ogni staypoint sarà costituito dall'elenco dei punti che contiene.

Algorithm 1 Algoritmo Curve Extrema

```
1: stayPoints ← empty list
2: currentStayPoint ← empty list
3: t ← trajectory positions
4: loop:
5: currentPoint ← pop(t)
6: if currentPoint.speed <= speedthreshold then
7:     currentStayPoint.add(currentPoint)
8: else
9:     if currentStayPoint! = empty list then
10:        stayPoints.add(currentStayPoint)
11:        currentStayPoint ← empty list
12: if t! = empty list then
13:    goto loop.
14: if currentStayPoint! = empty list then
15:    stayPoints.add(currentStayPoint)
16: return stayPoints
```

In breve: si procede con l'elaborazione di ogni punto all'interno della traiettoria, se il punto soddisfa i requisiti (la velocità $\leqslant$ **soglia**, C'(x,y) ≈ 0) allora entra a far parte dello staypoint. Se invece il punto ha una velocità maggiore della soglia allora lo staypoint sarà formato da tutti i punti precedenti consecutivi che rispettano il requisito. Quando si incontra un nuovo punto con la velocità inferiore alla soglia si crea un nuovo staypoint. Il costo computazionale dell'algoritmo espresso in questo modo è **lineare**.

2.2 Density Join Cluster

Il Problema

Il secondo obiettivo è estrapolare i luoghi importanti di un utente a partire dalla sua triettoria, la definizione formale del problema può essere posta in questo modo:

- **Obiettivo**: identificare i punti di interesse di un utente dato un elenco di posizioni
- **Input**: Traiettoria dell'utente rappresentata come set di posizioni
  S= $\{loc_i=(t_i, l_i)|i = 1...n\}$
  con l_i = (latitudine, longitudine) e t_i = timestamp.
- **Output**: Luoghi importanti L_i che rappresentano le posizioni frequentate in modo abitudinario dall'utente.

Il secondo algoritmo scelto è Density Join Cluster [28], il cui scopo è la ricerca di **Punti di interesse** di un determinato utente, chiamati anche **Personal gazetteers** (*dizionario geografico dei luoghi di interesse di una persona*).

Questo algoritmo come suggerisce il nome si fonda sulla creazione di cluster in base alla densità dei punti. In questo senso è un'estensione dell'algoritmo DB-SCAN [11]. Per la creazione dei cluster si sfrutta la densità del **vicinato locale** di ogni punto definito sulla base della *distanza*. Preso un punto (dalla traiettoria utente) il suo vicinato (**neighborhood**) è l'insieme di posizioni che rientra all'interno del cerchio tracciato con raggio *d*. Nel caso in cui il vicinato sia formato da un numero sufficiente di punti allora il punto e il suo vicinato formano un cluster, altrimenti vengono semplicemente scartati. Il processo viene ripetuto per ogni posizione, andando ad unire i cluster eventualmente sovrapposti (formati dalle stesse posizioni). Al termine dell'elaborazione si possono ottenere in questo modo cluster di forma arbitraria. La figura 2.10 mostra un cluster creato con questo approccio.

Il calcolo del vicinato può essere definito in questo modo [28]: Il ***density-based neighborhood N*** di un punto p denotato come **N(p)** è definito da

$$N(p) = \{q \in S | dist(p, q) \leqslant Eps\}$$

Figura 2.10: Esempio di cluster creato attraverso approccio basato sulla densità, i punti verdi formano il cluster, mentre quelli rossi sono rumore (punti scartati).

in cui S è il set di tutti i punti, q è un punto qualunque del campione, *Eps* è il raggio del cerchio intorno a p che definisce la densità e *MinPts* è il numero minimo di punti richiesti all'interno del cerchio per poter soddisfare il requisito di densità. L'operazione di unione di due cluster è definita come **join**. Il cluster creato a partire dal vicinato N(p) è *densità-collegabile* (*density-joinable*) a N(q), denotato come **J(N(p), N(q))**, se rispettando i parametri *Eps* e *MinPts* esiste un punto o tale che sia *N(p)* che *N(q)* contengono o. Un esempio grafico è mostrato in figura 2.11.

A questo punto si può dare la definizione matematica dell'algoritmo *DJ Cluster*:

$$\forall p \in S, \forall q \in S, \exists N(p), N(q) \text{ tali che } \exists J(N(p), N(q))$$

Il vantaggio principale di questo tipo di approccio è che il risultato è deterministico, quindi indipendentemente dall'ordine in cui sono elaborate le posizioni il risultato è sempre lo stesso, a differenza di altri tipi di clustering come il K-Means, in cui i cluster finali dipendono dall'assegnamento casuale iniziale dei punti [3].

Dimostrazione del determinismo del DJ Cluster

Tesi: DJ-Cluster produce un unico risultato di *clustering*.

Supponiamo che la relazione *R(p,q)* sia vera se p e q sono punti nello stesso cluster. R è quindi riflessiva (un punto è nel suo stesso cluster), simmetrica (se

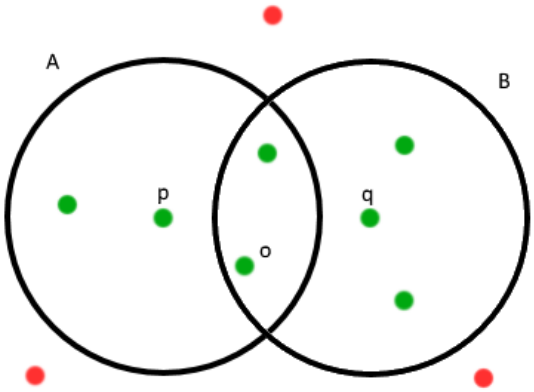


Figura 2.11: Il cluster A formato a partire dal vicinato di p e il cluster B formato a partire dal vicinato di q sono *density-joinable* poiché il punto *o* appartiene ad entrambi i cluster.

p è nel cluster di q allora q è nel cluster di p) e transitiva. Supponendo di avere R(p,q) e R(q,s), i punti saranno processati in un certo ordine. Se p fosse l'ultimo punto processato, sappiamo che q è nel cluster di s da R(q,s) ma anche che p è è nel cluster di q da R(p,q) quindi p verrà inserito (*merged*) nel cluster di q ed s, quindi R(p,s) sarà vera. Se invece l'ultimo punto processato fosse q allora sia p che s sarebbero nel suo vicinato e verrebbe creato un nuovo cluster a partire da N(q) che contiene tutti e tre i punti. Infine se l'ultimo punto processato fosse s allora sarebbe analogo al primo caso (ultimo punto processato p), dimostrando che la relazione R è anche transitiva. Dato che R è riflessiva, simmetrica e transitiva esiste un unico risultato di clustering possibile fornito da DJ Cluster.

2.2.1 Parametri di ingresso per l'elaborazione

I parametri di ingresso dell'algoritmo sono tre:

- traiettoria $\longrightarrow$ su cui effettuare la ricerca dei luoghi importanti.
- eps $\longrightarrow$ distanza massima alla quale calcolare il vicinato locale (neighborhood).
- minPts $\longrightarrow$ numero minimo di punti necessari per la creazione di un cluster.

La combinazione dei due parametri di ingresso determina la densità del neighborhood e quindi la dimensione e la forma dei cluster. Se i parametri decrescono allora il numero dei cluster aumenterà, diminuendo la dimensione.

2.2.2 Preprocessing

Eseguendo l'algoritmo su dati provenienti da utenti reali si formano una serie di problematiche, ad esempio se per andare al luogo di lavoro percorriamo sempre la stessa strada probabilmente molti dei punti della traiettoria saranno stati rilevati agli incroci di questa strada, la conseguenza sarebbe che le intersezioni stradali entrerebbero a far parte dei luoghi importanti falsificando i risultati. Per migliorare quindi le performance e i risultati si possono usare delle tecniche per filtrare dati non interessanti. Quella utilizzata da noi è stata eliminare le posizioni in cui la velocità era superiore ad una certa soglia, ottenendo solo i punti in cui l'utente era fermo (o quasi fermo). Questo ci ha permesso di rimuovere quasi tutti i punti collezionati durante la guida, posizioni a cui non siamo interessati.

Un'altra tecnica applicata riguarda il raggruppamento dei dati per posizione, infatti, dato che in questo algoritmo il tempo non è preso in considerazione, se una posizione è visitata due volte allora quelle due posizioni sono accorpate in una unica con ripetizione 2. Nei dati a nostra disposizione le posizioni si ripetono spesso (anche fino a migliaia di volte) questo permette di diminuire significativamente il numero totale di punti, incrementando ulteriormente le performance. Nella figura 2.12 è riportato il numero di punti di 10 utenti con e senza preprocessing, la differenza è significativa, passando da circa 30 mila punti per ognuno degli utenti a valori al di sotto di 3000 posizioni distinte, arrivando addirittura a 250 posizioni per alcune traiettorie.

2.2.3 Elaborazione

A questo punto sulle traiettorie a cui sono stati rimossi i punti non stazionari è possibile applicare normalmente l'algoritmo, di seguito nel pannello 2 viene presentato lo *pseudo codice* delle operazioni svolte per ottenere la sequenza di **stay-**

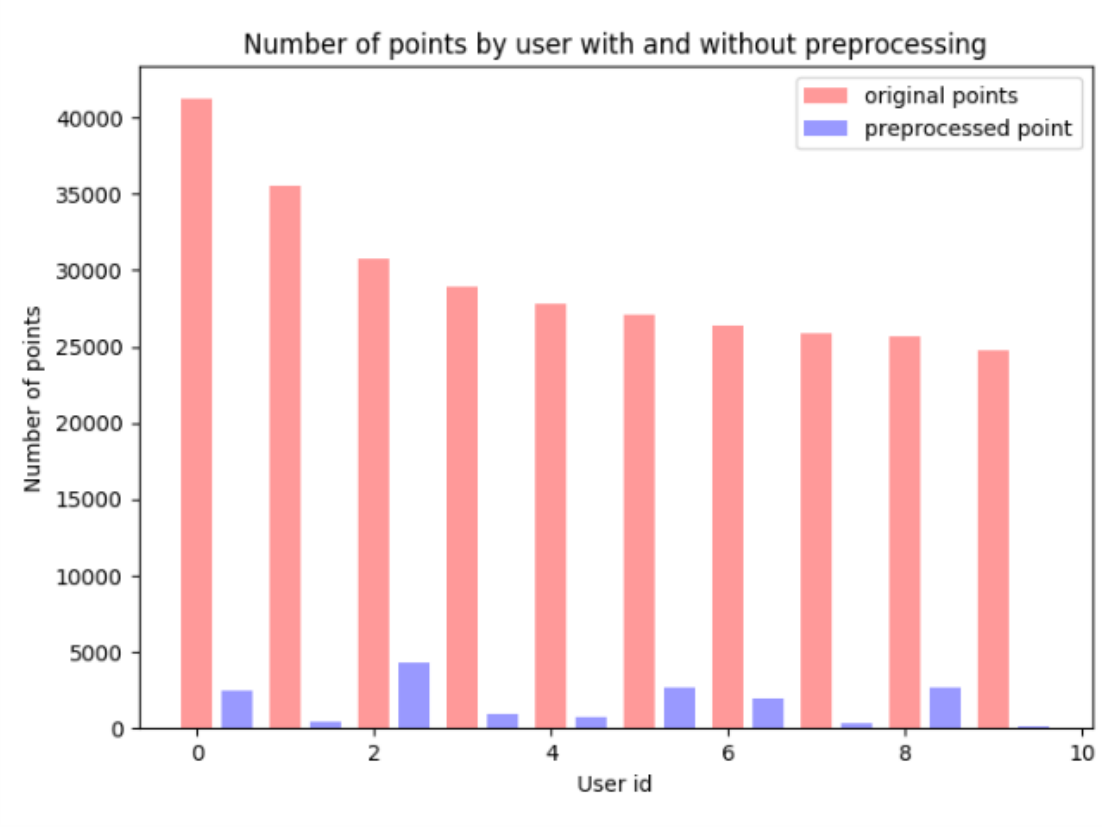


Figura 2.12: La figura mostra per 10 utenti il confronto tra numero di punti in rosso e il numero di punti dopo la procedura di preprocessing in blu.

points. Il risultato della procedura sarà un'elenco di **staypoints**, ogni staypoint sarà costituito dall'elenco dei punti che contiene.

Algorithm 2 Algoritmo DJ Cluster

```
1: t ← trajectory positions
2: loop:
3: p ← pop(t)
4: Compute desity based neighborhood N(p) of a point p wrt Eps and MinPts
5: if N(p) is null then
6:     label p as noise
7: else
8:     if N(p) is density joinable to an existing cluster then
9:         Merge N(p) and other density joinable clusters
10:    else
11:        Create a new cluster based on N(p)
12: if t! = empty list then
13:    goto loop.
14: return clusters
```

In breve: si procede con l'elaborazione di ogni punto all'interno della traiettoria calcolando il suo vicinato N(p) con i parametri Eps e MinPts, se il vicinato non è sufficientemente denso allora p viene contrassegnato come rumore. Se invece il vicinato soddisfa i requisiti di densità allora si crea un cluster nel caso non ce ne siano già che contengono uno qualsiasi dei punti di N(p), altrimenti si procede con il *merging* di tutti i cluster che siano *density joinable*. Il costo computazionale dell'algoritmo espresso in questo modo è O(n^2) in quanto per ogni punto deve essere processato il vicinato considerando tutti i punti rimanenti. Utilizzando un indice spaziale per la ricerca del vicinato è possibile diminuire il costo computazionale a O(n log n).

Capitolo 3

Il prototipo

3.1 Le Tecnologie

Il progetto si divide in tre componenti principali:

- La componente algoritmica di **elaborazione** si occupa dell'elaborazione dei dati. I dati sono elaborati in modalità batch attraverso un'applicazione sviluppata in Python, in quanto alcuni processi richiedono diverse ore.
- La componente di **memorizzazione** risiede su un server PostGIS ed è qui che tutti i dati, sia grezzi che elaborati, sono immagazzinati.
- La componente di **visualizzazione** rende disponibili i dati agli utenti, consiste in un sito web.

Per l'intero progetto è stato utilizzato **Git** come Sistema del Controllo di Versione (VCS), sfruttando le pipelines di BitBucket per la **continuos integration**, il tutto organizzato secondo modello **GitFlow**.

Elaborazione in Python

L'elaborazione dei dati è sviluppata in **Python** 3.6, adottando un approccio **Test-Driven Development** (sviluppo guidato dai test) in cui si cerca di creare test automatici in grado di verificare in maniera più approfondita possibile il corretto

funzionamento dell'applicazione. Per verificare che il codice rispetti le *convenzioni* sullo stile sono stati utilizzati degli strumenti come **flake8**[1], un tool di verifica delle convenzioni PEP8[2] (Python Enhancement Proposals #8 - Style Guide for Python Code, le linee guida per le convenzioni sul codice). Anche l'esecuzione di questo tool è stato inserito nelle Pipelines di BitBucket in modo da verificare in automatico ad ogni *push* (invio del codice al repository remoto) che il codice rispetti tutte le convenzioni. *PMD* è stato un tool preso in considerazione per l'analisi del codice sorgente ma purtroppo ad oggi le uniche regole supportate per python[3] sono quelle per il CPD (Copy/Paste Detector, regole per la ricerca del codice sorgente duplicato).

Memorizzazione su PostGIS

Tutti i dati risiedono su un database **PostGIS** all'interno della rete universitaria. Per la verifica dei dati inseriti è stato utilizzato PgAdmin. La figura 3.1 mostra la parte principale dello schema er del database. Le tabelle dj_cluster_runs e ce_cluster_runs registrano le esecuzioni degli algoritmi con tutti i loro parametri e altri dati utili ai fini di verifica come il tempo di esecuzione. A queste tabelle sono associate rispettivamente dj_cluster_staypoints e ce_cluster_stay_points con una relazione uno a molti. Ad ogni tupla di queste tabelle sono associate con una relazione uno a molti le tuple di significative_data_enriched in modo da tenere traccia di come sono composti i cluster. Le righe di quest'ultima tabella sono associate uno a uno con quelle di significative_data in modo da identificare l'origine dei dati arricchiti.

Visualizzazione su sito web PHP

I dati sono consultabili attraverso l'utilizzo di un portale che mette a disposizione dell'utente la possibilità di visualizzare diversi livelli informativi su una mappa interattiva. Il sito prevede l'autenticazione tramite login per proteggere la privacy degli utenti di cui sono stati raccolti i dati ed è stato sviluppato con un design

[1] `http://flake8.pycqa.org/en/latest/index.html`
[2] `https://www.python.org/dev/peps/pep-0008/`
[3] `https://pmd.github.io/pmd-5.8.1/pmd-python/index.html`

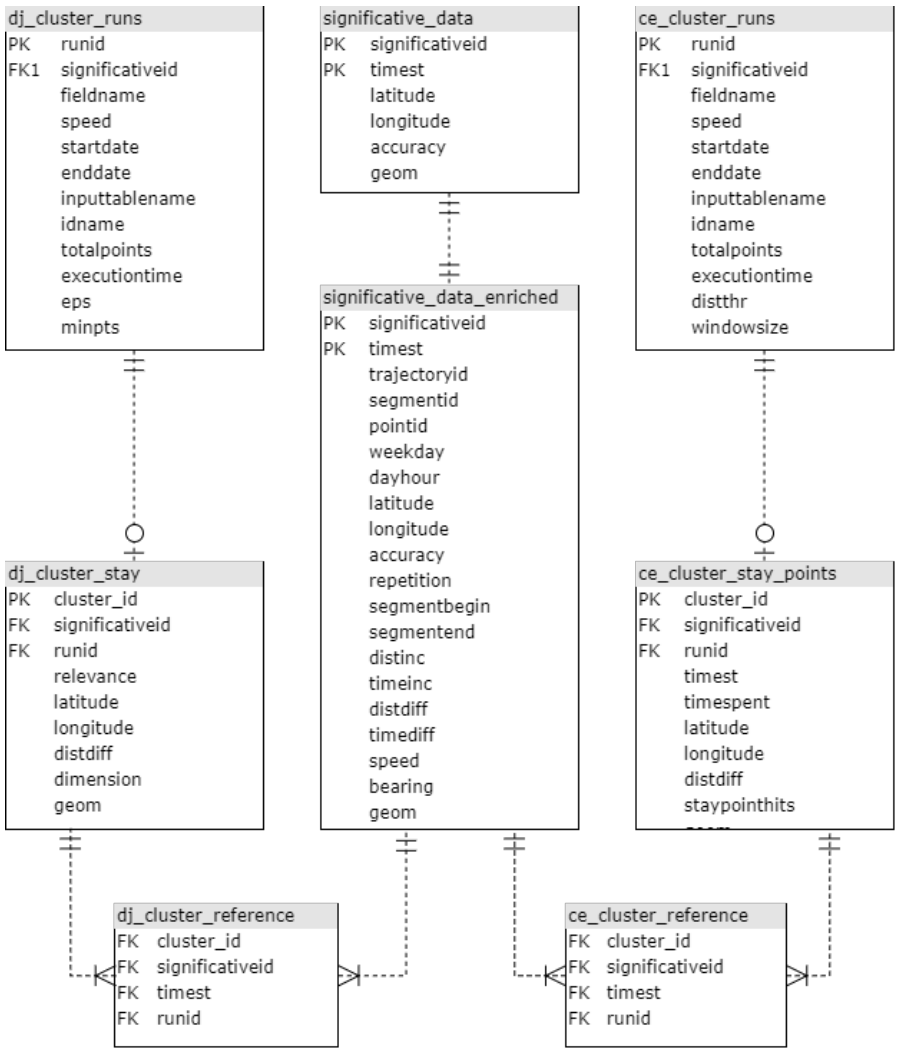


Figura 3.1: Parte principale dello schema ER del database.

responsive (design fluido che si adatta alla dimensione dello schermo su cui è visualizzato) con il supporto al mobile.

I layer che è possibile mostrare sono:

- Raw data, mostra i dati grezzi senza che siano state fatte elaborazioni su di essi. La colorazione indica l'utente a cui appartengono i dati.
- Statistical Area, mostra le aree statistiche del comune di Bologna, la colorazione di questo layer indica il reddito medio dei residenti nelle varie aree.
- Segments, mostra una segmentazione dei dati, su questo particolare layer è possibile indicare due diversi significati per la colorazione delle traiettorie, un gradiente per indicare la velocità delle traiettorie oppure un gradiente per indicare l'orientamento (bearing).
- Risultati del DJ-Cluster, questo layer aggiunge alla mappa i centroidi risultanti dall'esecuzione del DJ Cluster con un punto giallo con il bordo del colore dell'utente a cui appartiene.
- Risultati del Curve Extrema, questo layer mostra i centroidi risultanti dall'esecuzione del Curve Extrena con un punto verde con il bordo del colore dell'utente a cui appartiene.
- Nodi che indicano i pattern **AbitaIn** e **LavoraIn** rispettivamente colorati di rosso e blu con il contorno del colore identificativo dell'utente, hanno un raggio maggiore rispetto agli altri punti sulla mappa.

È possibile applicare diversi filtri ai dati estrapolati

- La sorgente dei dati, con cui è possibile scegliere il dataset da utilizzare, se quello proveniente dal BIG oppure quello con il set di dati anonimi.
- Date di inizio e di fine, per applicare un filtro temporale ai risultati.
- Fascia oraria, per recuperare dati solo in determinare ore.

- Id utente, per filtrare i dati di un determinato utente.
- Accuratezza, per visualizzare solo i dati che abbiano un'accuratezza maggiore di quella scelta.

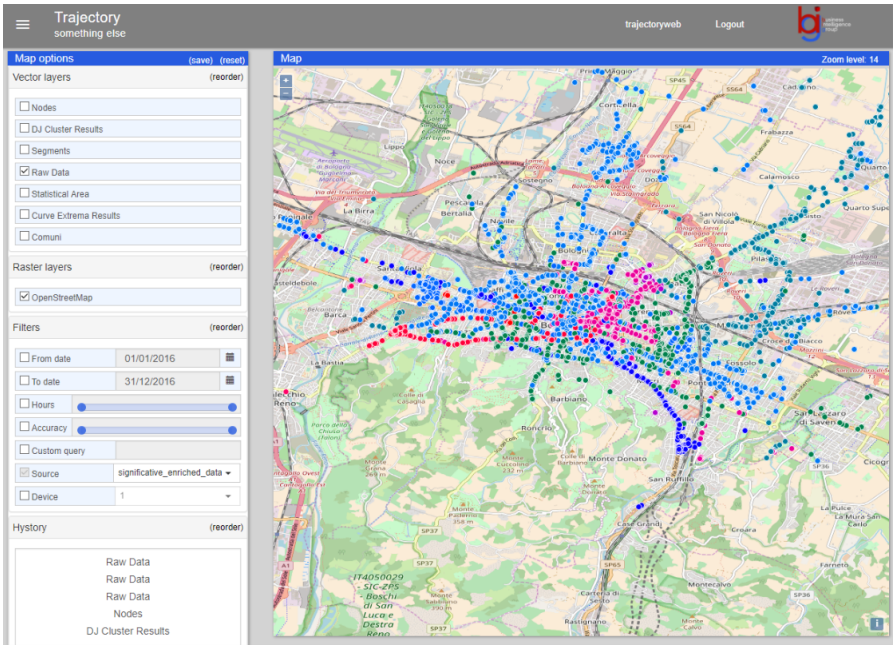


Figura 3.2: Visualizzazione dei dati raw sul sito web, sulla mappa sono mostrati i raw data dei primi 10 utenti dei dataset anonimi.

Inoltre è presente una sezione che permette di rieseguire le ultime interrogazioni.

La figura 3.2 mostra la schermata principale del sito, che consiste in una mappa interattiva su cui è possibile applicare vari layer da mostrare.

3.2 Architettura

La figura 3.3 mostra il diagramma di deployment del progetto, si può notare la suddivisione in 3 componenti illustrata precedentemente.

Come mostrato dal diagramma le interazioni tra utente e sistema avvengono solo attraverso il sito web, raggiungibile su internet. Il server web a sua volta recupera i dati dal Server PostGIS richiedendo di volta in volta le feature necessarie

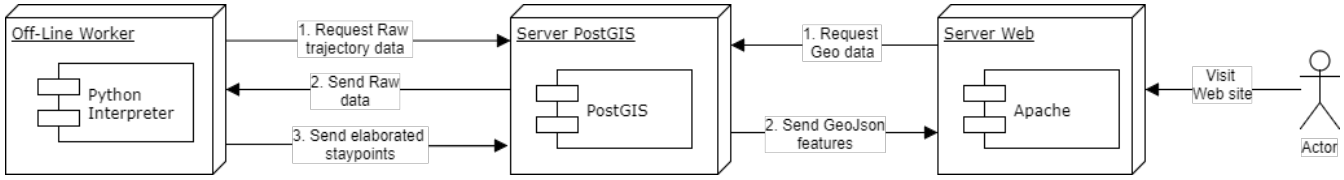


Figura 3.3: Diagramma di Deployment del progetto

codificate in GeoJson. Perchè il sistema funzioni l'applicazione Python deve aver precedentemente elaborato i dati grezzi in maniera off-line.

Di seguito viene riportata l'analisi del sottostistema Python. Come semplificazione si è deciso di dividere i componenti dell'applicazione Python in sottogruppi, in questo modo è più semplice un'eventuale suddivisione del lavoro. Per chiarezza viene riportato in figura 3.8 l'UML completo e di seguito vengono mostrati i sottogruppi.

3.2.1 MyData

I componenti di questo gruppo, mostrati in figura 3.4 sono votati all'interfacciamento con il database, infatti sono presenti 4 entità che riflettono la necessità di contenere informazioni in maniera tabellare e incapsulano i metodi di creazione delle tabelle su database di tipo diverso (inizialmente PostGIS e Oracle 11G, ma è stato organizzato in modo da poter aggiungere nuovi tipi di database molto semplicemente). La classe *Column* associa tra di loro una coppia di stringhe Nome - Tipo. *Table* è un contenitore di Column e in più contiene il nome della tabella. La classe *AlgorithmUtils* contiene un'elenco di Table e alcuni metodi statici che vengono utilizzati dagli algoritmi per effettuare calcoli, filtrare dati o fare l'upload su database. *Content* incapsula i dati veri e propri all'interno di un DataFrame (oggetto che rappresenta una tabella), più le strutture necessarie per estrapolare l'sql per la creazione della tabella su database. Infine la classe *MyData* contiene i campi per distinguere a quale database e quale tabella i dati appartengono e che stringhe sql utilizzare, i metodi per l'aggiunta di dati a al contenitore e altri metodi per la gestione del codice sql.

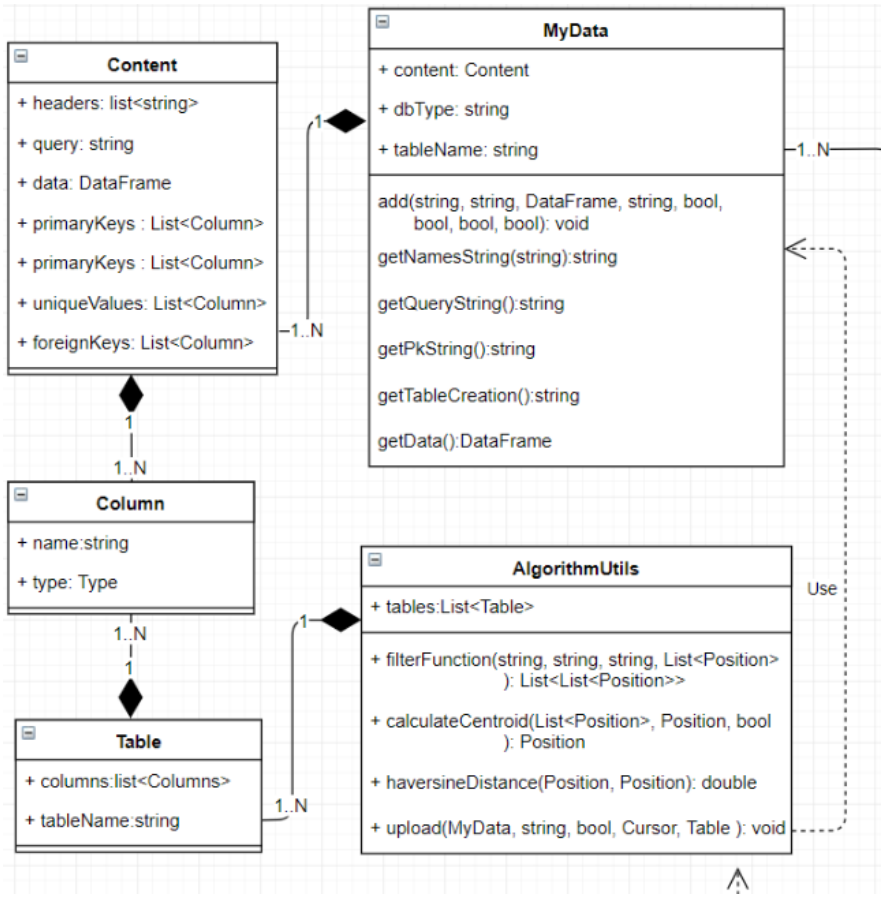


Figura 3.4: UML MyData

3.2.2 Algoritmi

La figura 3.5 mostra la struttura delle classi che riguardano gli algoritmi, qui è facilmente intuibile l'utilizzo del pattern *Command*, infatti tutti gli algoritmi fanno riferimento a una classe base *BaseAlgorithm* che contiene al suo interno una classe (*MyThreadPool*) per la parallelizzazione del lavoro, per il logging e la funzione da eseguire. Tutti gli altri algoritmi fanno riferimento alla classe base estendendo la parte di configurazione e ridefinendo la funzione da eseguire. Fanno eccezione le classi *DJClusterAlgorithm* e *CurveExtremaAlgorithm* che hanno in comune una parte della configurazione e quindi si è deciso di farli estendere una classe *SpatialAlgorithm* che contiene la configurazione necessaria ad algoritmi di tipo spaziale (in futuro si prevede di implementare nuovi algoritmi spaziali per l'elaborazione dei dati grezzi). La classe *Runner* presenta la particolarità di poter eseguire altri algoritmi. Al fine di incrementare le performance è stato introdotto il componente MyThreadPool che permette di utilizzare un pool di processi indipendenti. Questo ha permesso di utilizzare più processi Python alla volta in grado di elaborare singolarmente i vari utenti. In futuro si prevede di riuscire ad aumentare ulteriormente il livello di parallelizzazione.

3.2.3 Clusters

Questo gruppo di classi mostrato in figura 3.6 contiene la logica per l'elaborazione delle posizioni spaziali, la classe *Position* è una tupla di coppie chiave - valore e continene tutte le informazioni che contraddistinguono un ping GPS, nell'uml sono riportate le più significative. La classe *SimpleCluster* contiene i campi base per la gestione di un insieme di posizioni, inoltre contiene i metodi per il calcolo del centroide del cluster e della dimensione. Infine la classe *Cluster* estende SimpleCluster aggiungendo due metodi utili al fine di unire cluster differenti.

3.2.4 Draw

Le classi che fanno parte del gruppo draw come mostrato in figura 3.7 sono state ideate per la creazione dei report, e presentano la possibilità di mostrare

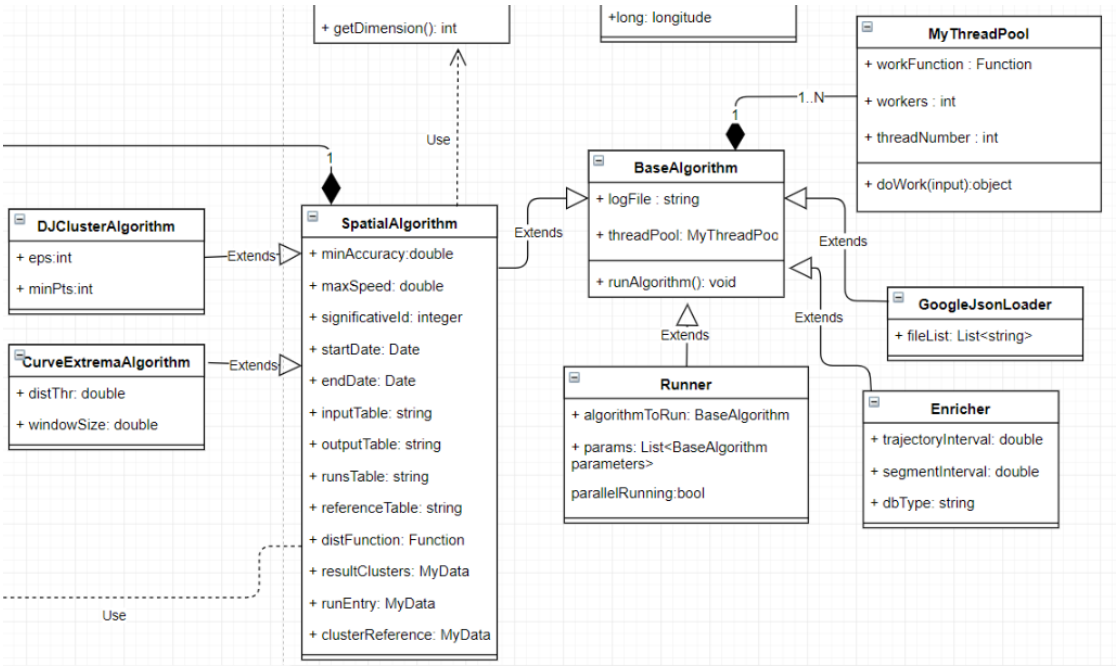


Figura 3.5: UML Algoritmi

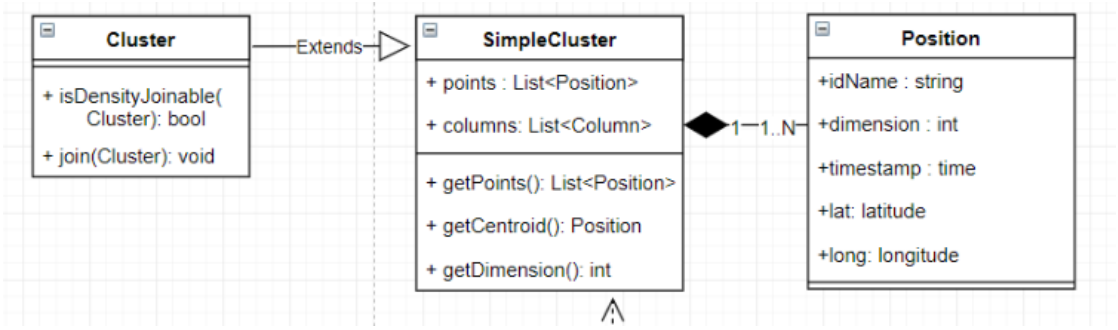


Figura 3.6: UML Clusters

a video grafici bidimensionali (punti o segmenti) oppure grafici a istogramma tridimensionali. La classe *BaseDraw* specifica le operazioni disponibili, mentre le altre classi (**CEDraw** e **DJDraw**) utilizzano i metodi della classe BaseDraw e operazioni specifiche per i disegni necessari.

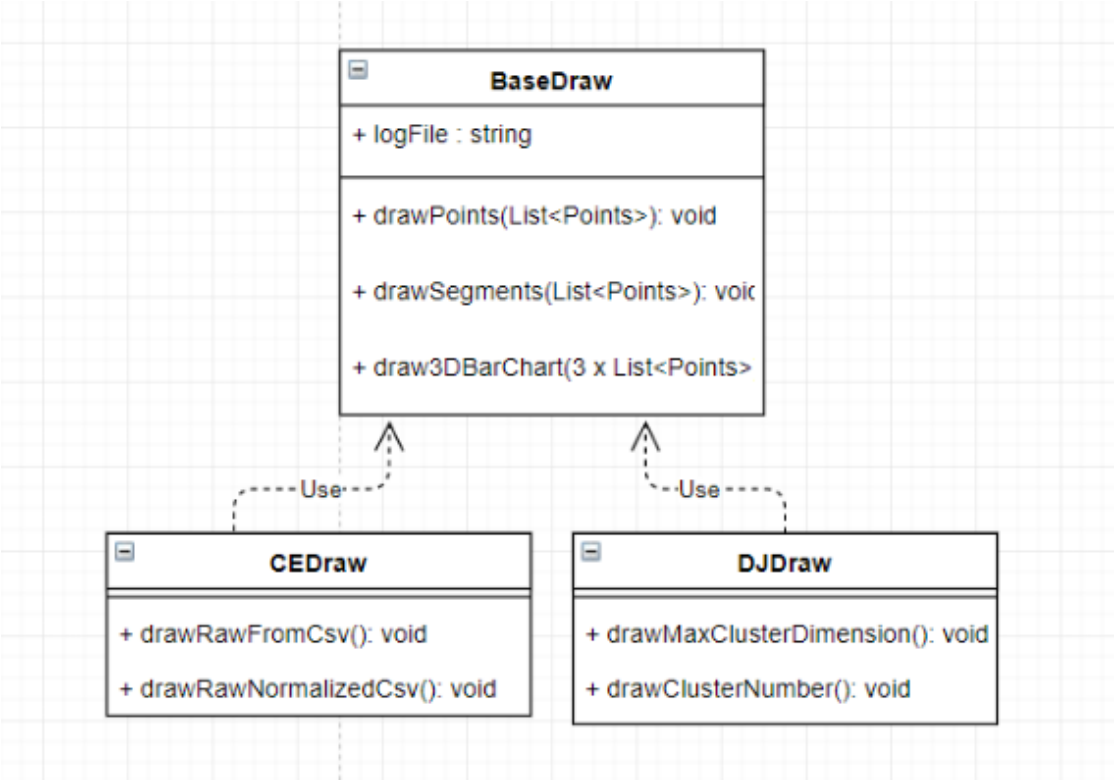


Figura 3.7: UML MyData

3.3 I data Set

I dataset attualmente in nostro possesso sono due:

- un dataset di dati anonimi registrati nell'arco di circa 40 giorni nella zona metropolitana di Bologna. Il numero di utenti di questo set è molto elevato (circa 200 mila) e il numero totale di posizioni arriva a circa 63 milioni, anche se non tutti hanno una quantità sufficiente di dati per poter estrapolare

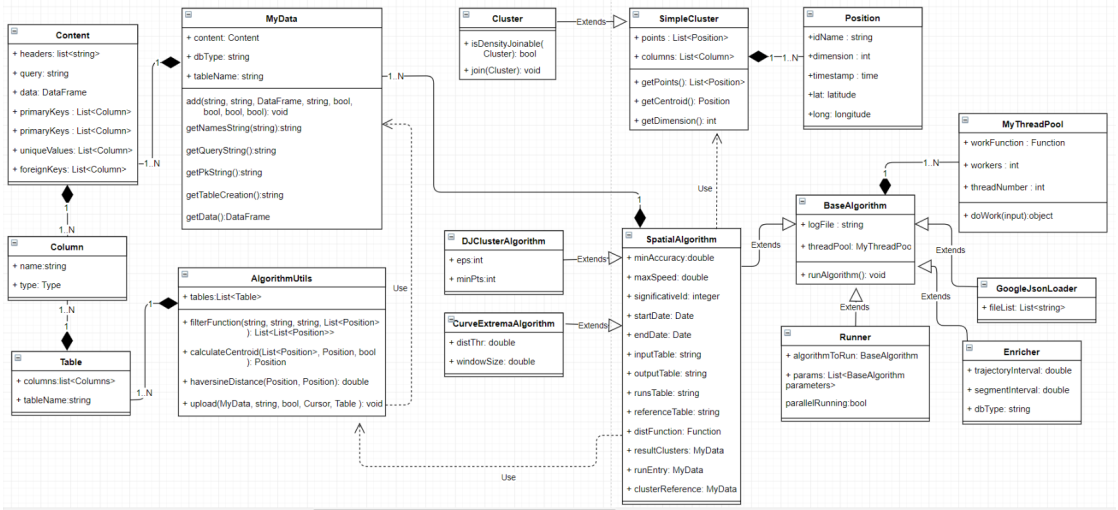


Figura 3.8: UML completo

informazioni utili. Gli utenti più significativi di questo dataset possiedono circa 30 mila ping GPS, distribuiti in modo omogeneo durante l'arco di tempo. L'accuratezza media degli utenti più significativi di questo dataset è di circa 22m.

- un dataset di dati raccolti all'interno del gruppo di ricerca BIG (Busines Intelligence Group) per poter verificare l'efficacia di quanto sviluppato. Questo dataset comprende i dati di una decina di persone. Sebbene il numero di utenti in questo set sia relativamente basso, i dati sono distribuiti lungo un arco di tempo molto maggiore, andando dall'inizio del 2012 ad oggi. In questo modo alcuni utenti di questo set hanno fino ad un milione di posizioni ciascuno.

3.4 L'applicazione

L'applicazione Python è in sostanza un elaboratore batch di grosse quantità di dati, essendo sprovvista di interfaccia grafica si è scelto di implementare questi pattern:

Behavioral Patterns

- **Iterator**: Necessario per la navigazione dei dati e la generazione di liste di risultati, accoppiandoli con il concetto di **generatori** è possibile crearne il contenuto solamente quando strettamente necessario. Questo pattern è stato sfruttato in tutte le classi che utilizzano dati provenienti dal database, è stato utilizzato anche per l'iterazione dei risultati generati dagli algoritmi.

- **Command**: Questo patter viene utilizzato nel momento in cui abbiamo la necessità di "configurare" o preparare ciò che verrà eseguito successivamente. In questo modo si separa il codice dell'azione da eseguire (che potrebbe essere anche molto computazionalmente onerosa) dal codice che effettivamente ne richiede l'esecuzione. La classe Runner sfrutta in modo intenso questo pattern per configurare tutti gli algoritmi da eseguire prima dell'effettiva esecuzione.

Structural Patterns

- **Facade**: Il pattern Facade permette di semplificare l'utilizzo di un insieme di oggetti (o di API) mascherando come effettivamente vengono eseguite le chiamate, e semplificandone quindi l'utilizzo esponendo un sottoinsieme delle azioni eseguibili. Questo pattern è stato utilizzato all'interno di una classe che incapsula la connessione al database per evitare di lasciare connessioni aperte dopo il suo utilizzo.

- **Adapter**: Il pattern adapter si occupa di modificare l'interfaccia di un oggetto per renderlo *adattabile* alla situazione. Il fine è quello di rendere interoperabili diverse interfacce. Ad esempio nella classe MyData sono presenti dei metodi per poter sfruttare diversi tipi di database e di strutture dati.

Esempio di Workflow di funzionamento

L'applicazione Python può permettere la creazione dell'intera base di dati attraverso una sola linea di comando, eseguendo le operazioni nella sequenza:

- Creazione del database PostGis
- Upload di dati Google: in aggiunta alla possibilità di elaborare i dati grezzi viene inserito un componente che si occupa di assimilare dati json provenienti dalla cronologia delle posizioni di Google (precedentemente esportata) e di trasformarli in base alle necessità dell'applicazione per poi caricarli sul database per sfruttarli successivamente.
- Arricchimento dei dati: i dati grezzi per poter essere elaborati dagli algoritmi necessitano di un processo di arricchimento (*enrichment*), questo permette di aggiungere proprietà ai singoli ping GPS utili ai fini dell'applicazione.
- DJCluster: per la scoperta del *personal gazetteer*.
- CurveExtrema: per la scoperta degli *stay points*.
- Report grafici.

Per quanto riguarda i parametri di funzionamento possono essere specificati nello stesso file di configurazione, se non diversamente indicato sono utilizzati dei valori di default.

La figura 3.9 mostra l'ordine degli step precedentemente illustrati

Figura 3.9: Step di funzionamento dell'applicazione

3.4.1 Risultati

Al termine dello sviluppo sono stati generati dei test per poter verificare gli indici di bontà degli algoritmi. Nello specifico sono stati creati:

- test di **efficienza**: calcolare come varia il tempo di esecuzione al variare dei punti (*costo computazionale*).
- test di **efficacia**: verifica sulla base dei risultati ottenuti a partire dai dataset conosciuti (*ground truth*).
- test di **robustezza**: verifica che gli algoritmi funzionino correttamente in base ai parametri.
- test di **corrispondenza**: verifica che entrambe le tecniche adottate diano risultati compatibili.

Test di efficienza

I test di efficienza mostrano la complessità computazionale individuata nel capitolo precedente. In particolare come mostrato in figura 3.10 la complessità dell'algoritmo DJ cluster è $O(n^2)$, la figura mostra come il tempo di esecuzione salga molto velocemente passando da pochi secondi per elaborazioni di traiettorie con poche centinaia di punti a ore quando i punti sono qualche decina di migliaia. Il numero di punti presi in considerazione è quello ottenuto dopo la fase di preprocessing descritta nel capitolo 2.2.2.

Per quanto riguarda l'algoritmo Curve Extrema il risultato ottenuto dal test di efficienza è particolarmente interessante. Infatti si è dimostrato che anche se la complessità è lineare, il parametro "distThr" necessario per la fase di preprocessing incide molto sul tempo di esecuzione. La fase di preprocessing diminuisce il numero totale di punti distinti in base al parametro, rendendo di fatto più efficiente il calcolo totale. Sebbene i risultati siano stati ottenuti dalle stesse traiettorie elaborate con il DJ cluster il numero di punti è diverso a causa della fase di preprocessing. L'algoritmo Curve Extrema si dimostra nettamente più veloce del DJ Cluster su traiettorie con decine di migliaia di punti conseguentemente al fatto che la

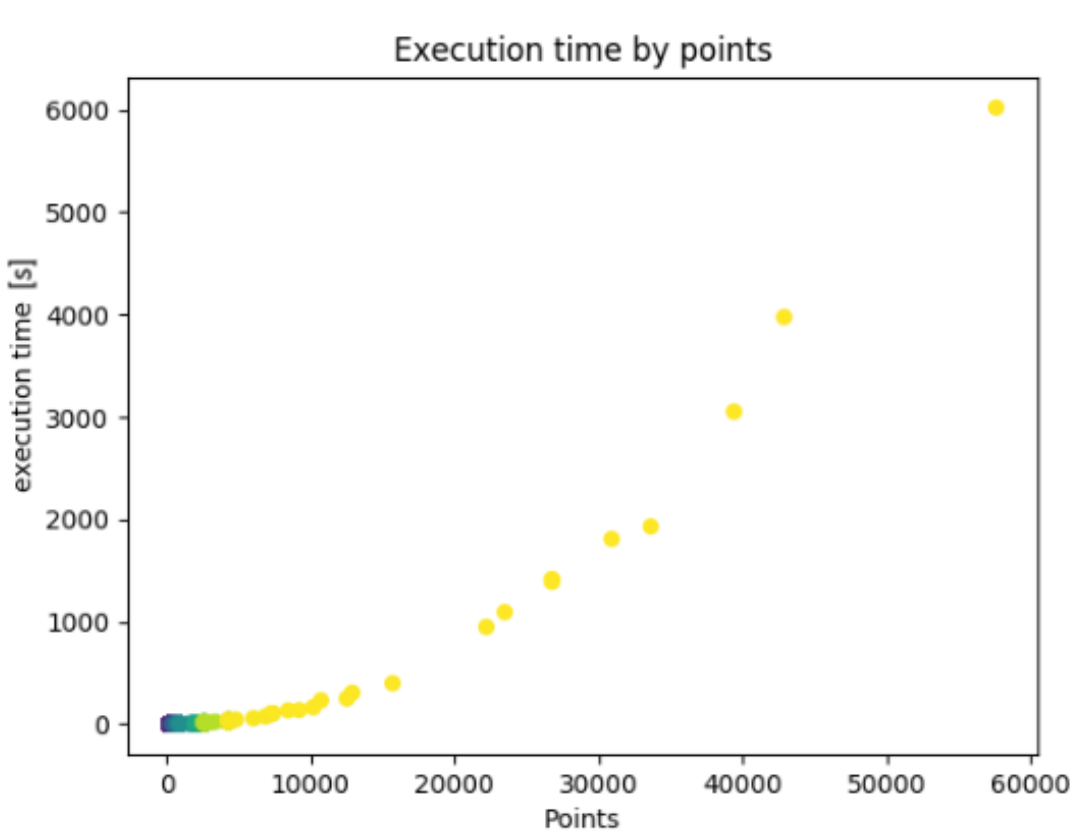


Figura 3.10: Correlazione esponenziale tra numero di punti e tempo di esecuzione in DJ Cluster.

complessità computazionale è lineare, eseguendo sempre al di sotto del minuto anche con 200 mila posizioni distinte.

La figura 3.11 mostra di fatto una correlazione lineare tra numero dei punti e tempo di esecuzione come indicato in precedenza. Mentre la figura 3.12 evidenzia che la procedura di preprocessing è fondamentale per l'incremento dell'efficienza dell'algoritmo, mostrando che al crescere del parametro distThr il tempo di esecuzione medio scende velocemente.

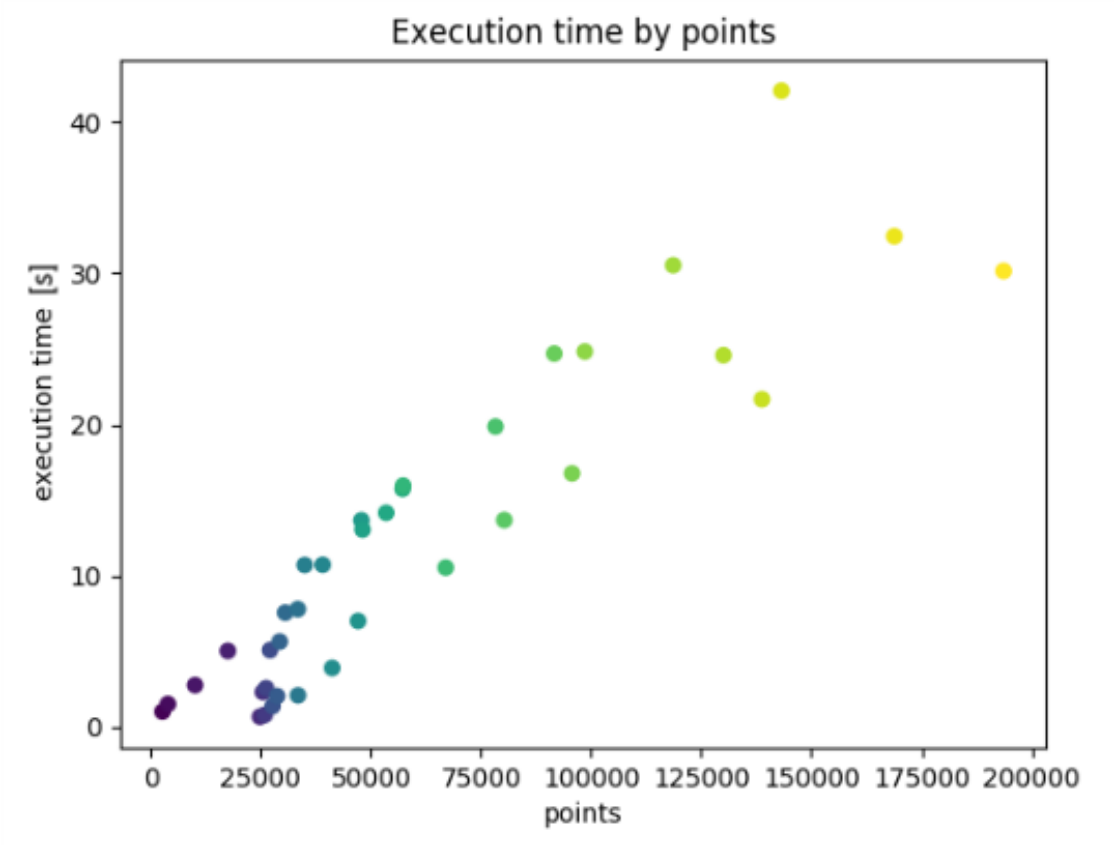


Figura 3.11: Correlazione direttamente proporzionale tra numero di punti e tempo di esecuzione in Curve Extrema.

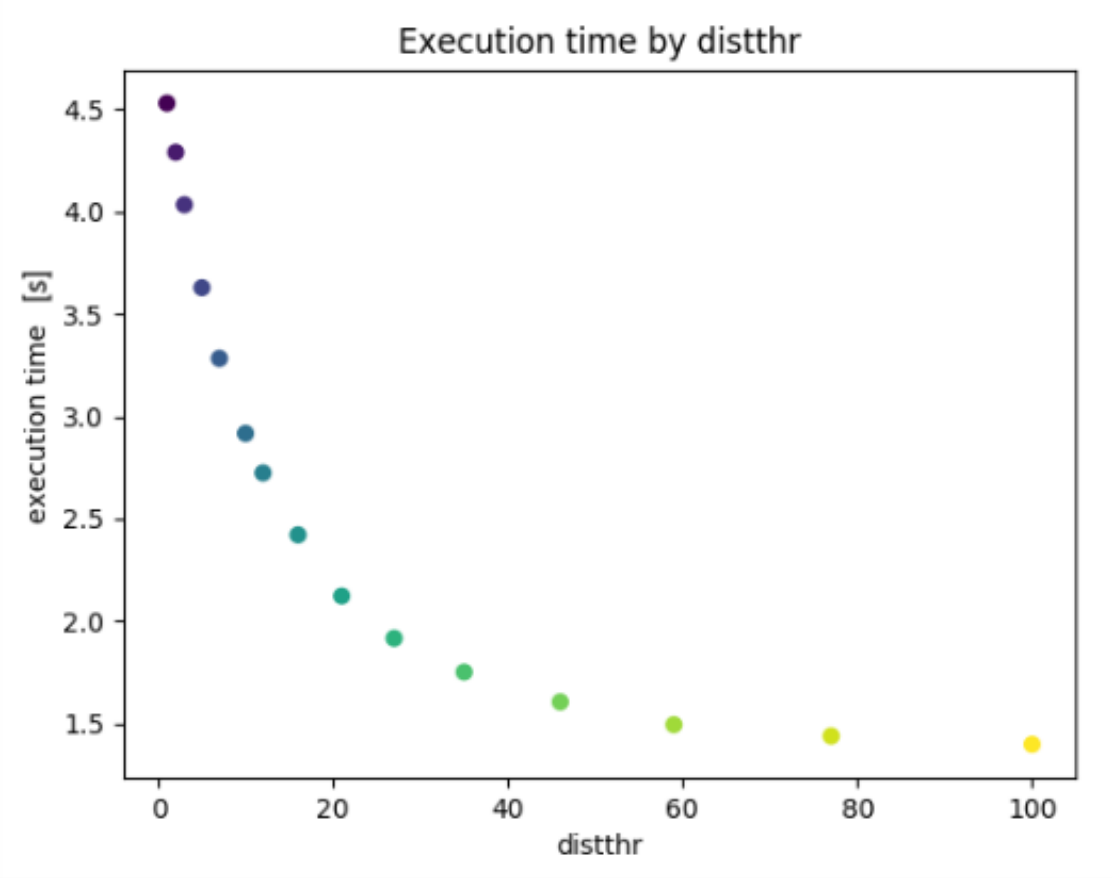
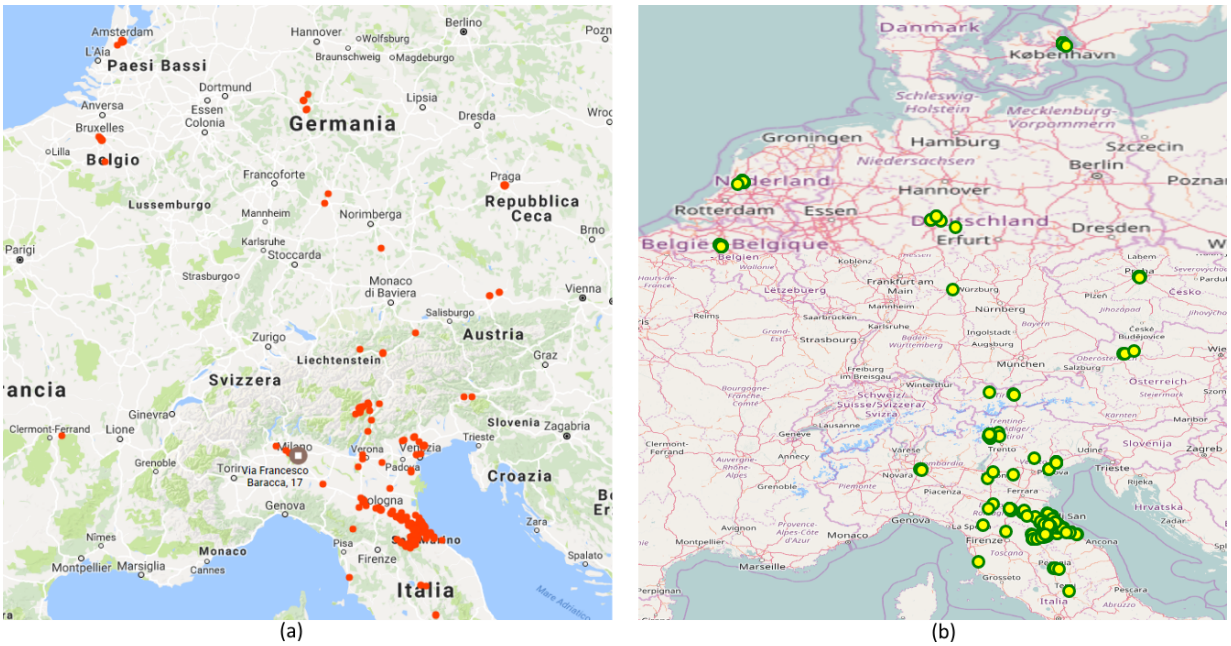


Figura 3.12: Correlazione inversamente proporzionale tra parametro distThr e tempo di esecuzione medio.

Test di efficacia

I risultati ottenuti da entrambi gli algoritmi si sono dimostrati particolarmente precisi, riconoscendo in tutti i casi le posizioni relative ai pattern AbitaIn e LavoraIn con una precisione molto elevata (nell'ordine dei metri). Una riflessione interessante è che la creazione del personal gazetteer se confrontato visivamente con i luoghi visitati proposti dal servizio Google Timeline, nominato precedentemente, sia estremamente simile. La figura 3.13 (a) mostra una schermata recuperata dai luoghi visitati su Google Timeline di un utente appartenente al dataset degli utenti noti, mentre la figura 3.13 (b) mostra il dizionario dei luoghi importanti creati attraverso l'elaborazione dei dati grezzi.



(a) (b)

Figura 3.13: Confronto tra luoghi visitati di Google Timeline e dizionario dei luoghi importanti creato attraverso questo progetto.

Test di robustezza

Per l'algoritmo DJ cluster in figura 3.14 sono messi in relazione i due parametri di input "minpts" e "eps". È possibile notare come Eps e MinPts siano inversamente proporzionali al numero totale di cluster. Le figure 3.15 e 3.16 mostrano nel dettaglio le proiezioni del grafico 3.14 rispetto ai valori dei due parametri.

Nella figura 3.17 invece è mostrata la relazione tra i parametri di input e la dimensione massima dei cluster formati. Un'osservazione interessante è che la dimensione massima dei cluster dipende esclusivamente dal parametro eps nei dataset in nostro possesso, in quanto il cluster maggiore di solito si aggrega nella posizione di casa, luogo in cui le posizioni sovrapposte sono in numero molto elevato. In questo caso dato che per l'utente in questione si hanno circa 30 mila posizioni sovrapposte (o comunque nell'intorno di pochi metri) il parametro minpts che varia tra 1 e 100 è ininfluente. Perché il parametro minpts incidesse sulla dimensione massima del cluster avremmo dovuto scegliere un valore molto maggiore.

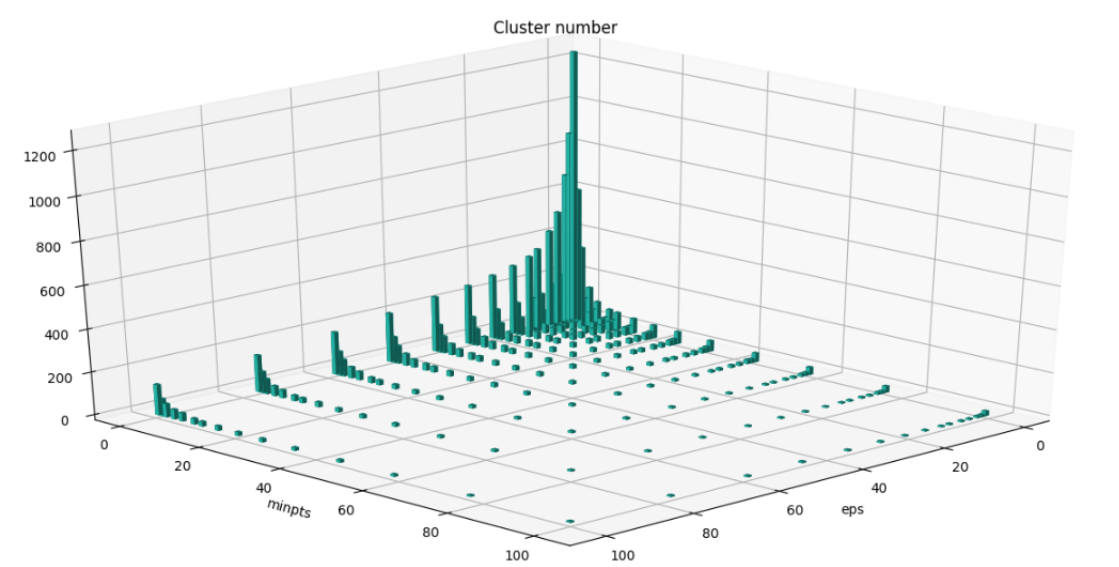


Figura 3.14: Istogramma che mette in relazione il numero di cluster al variare dei parametri "minpts" e "eps" a parità di traiettoria.

Per quanto riguarda Curve Extrema un grafico interessante, in figura 3.18, mostra come il numero totale degli staypoints trovati dall'algoritmo Curve Extrema

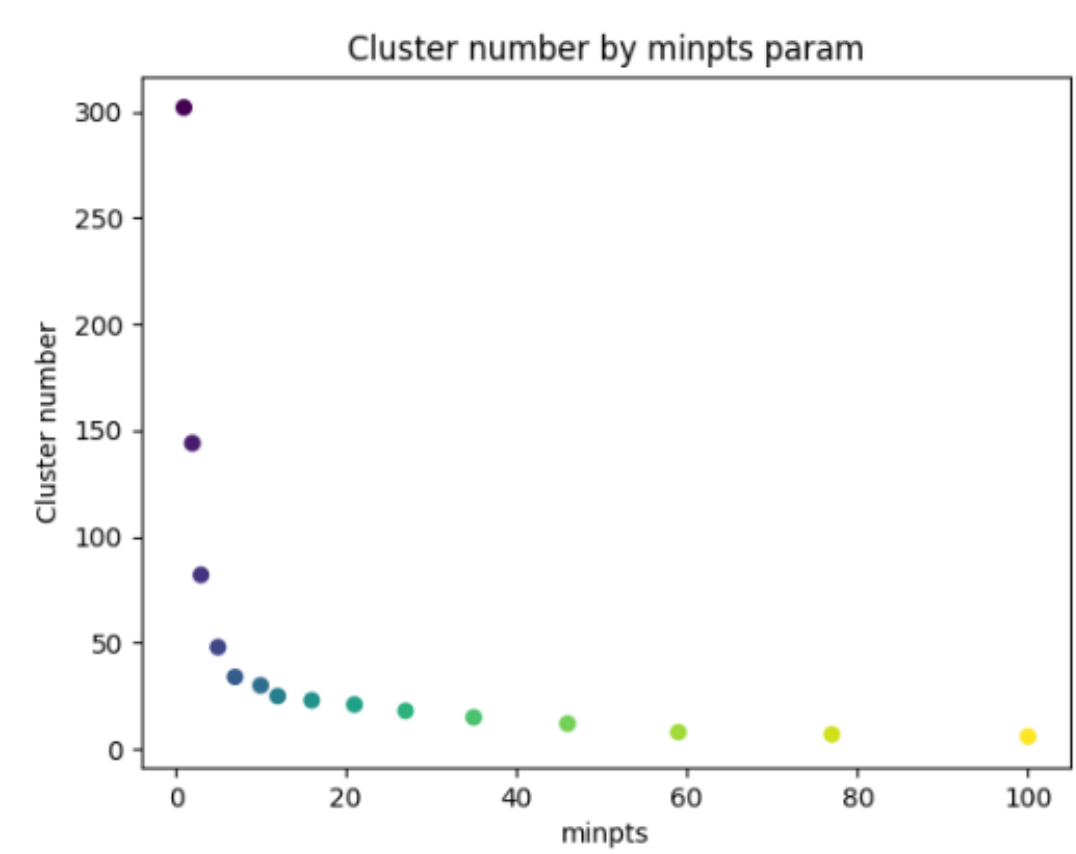


Figura 3.15: Grafico che mette in relazione il numero dei cluster al variare del parametro “minpts” a parità di traiettoria e di parametro “eps” (in questo caso impostato a 20).

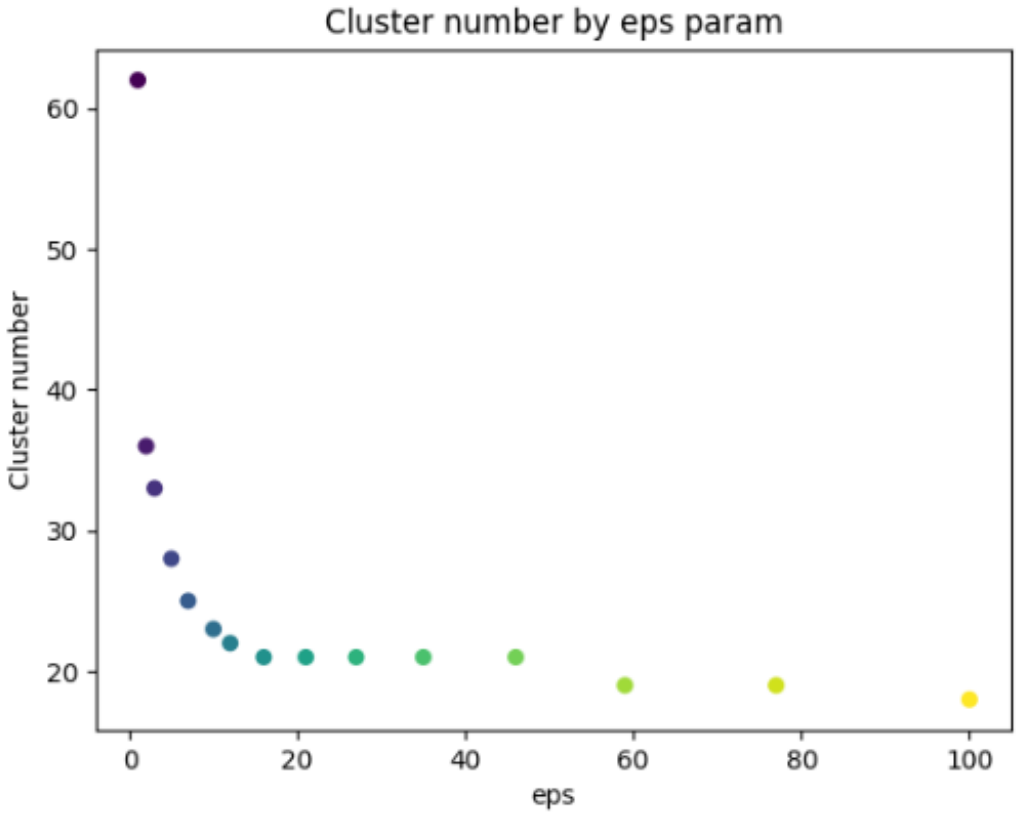


Figura 3.16: Grafico che mette in relazione il numero dei cluster al variare del parametro "eps" a parità di traiettoria e di parametro "minpts" (in questo caso impostato a 20).

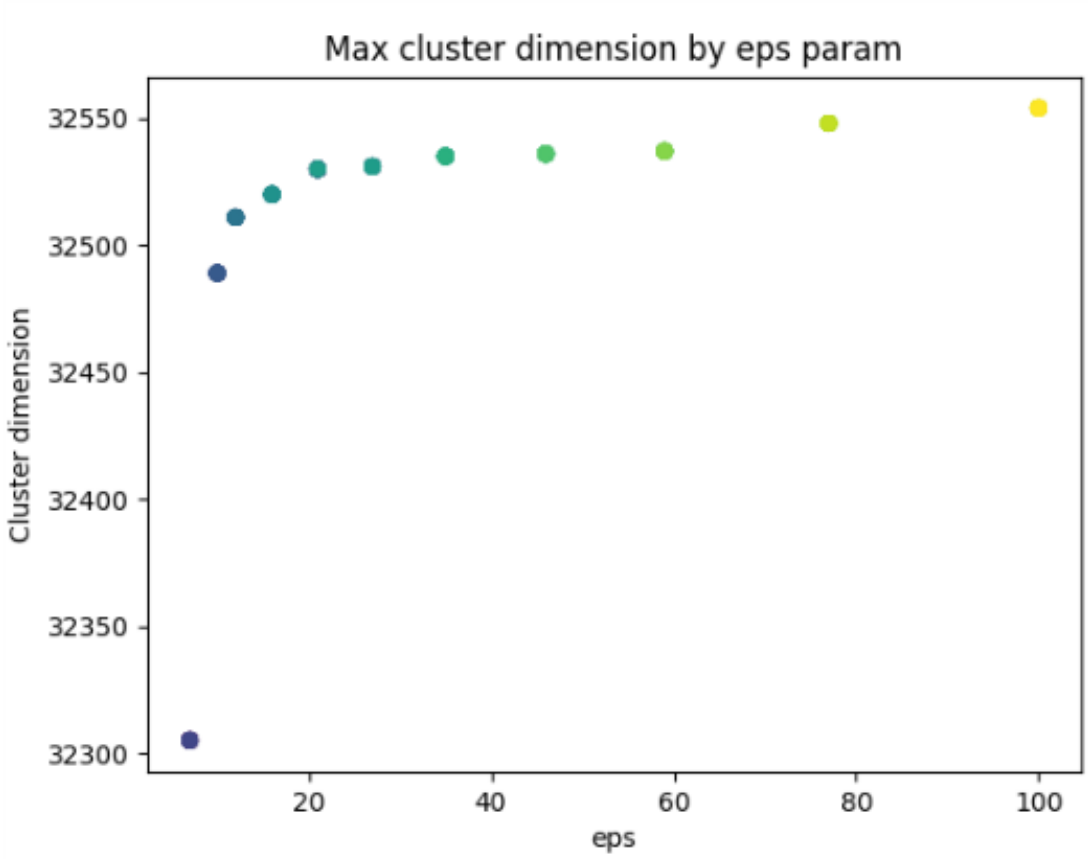


Figura 3.17: Grafico che mette in relazione la dimensione massima dei cluster creati con l'algoritmo DJ Cluster al variare del parametro "eps" a parità di traiettoria.

cambia al variare del parametro "distthr" che incide sul preprocessing.

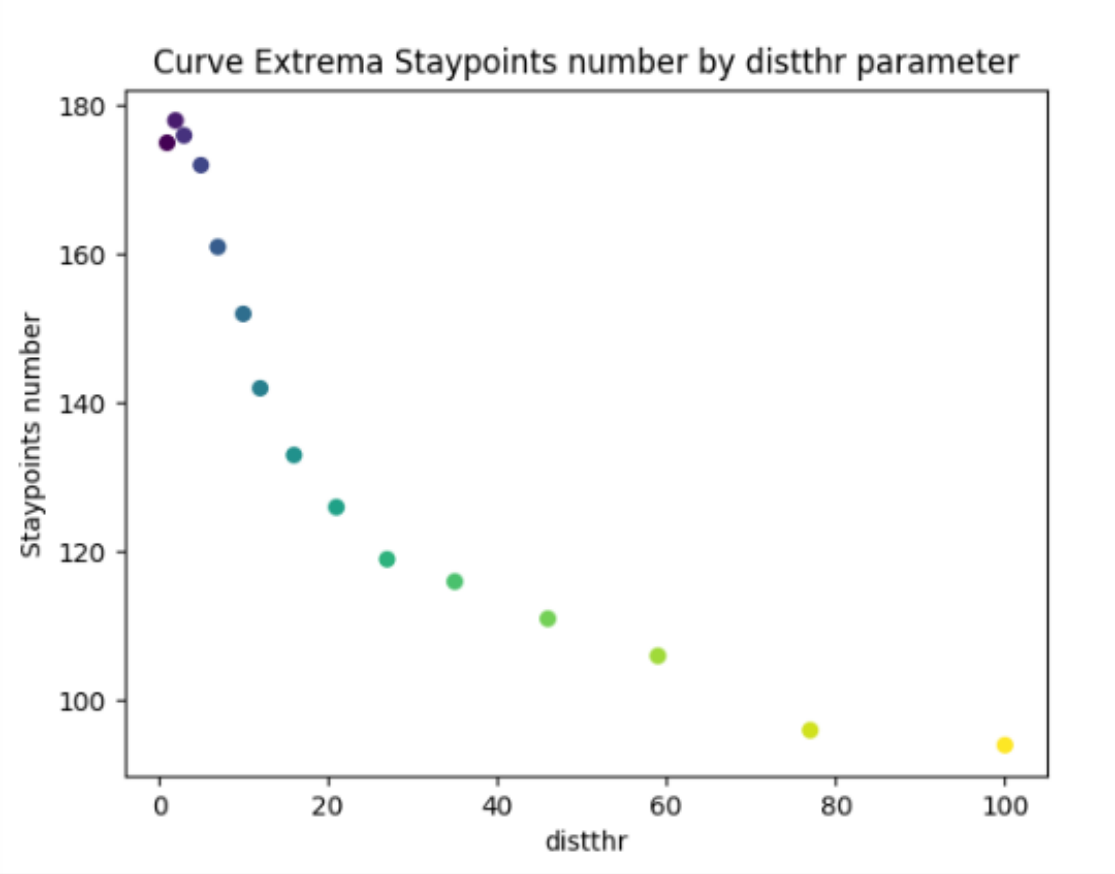


Figura 3.18: Grafico che mette in relazione il numero degli staypoints riconosciuti al variare del parametro "distthr" a parità di traiettoria e di parametro windowsize (in questo caso settata al 5% dei punti totali della traiettoria).

Test di corrispondenza

Questo test si occupa di verificare che l'esecuzione dei due algoritmi dia risultati simili per quanto riguarda le posizioni fondamentali di un utente (AbitaIn e LavoraIn). Il test si dimostra passato in quanto i risultati del Curve Extrema e del DJ Cluster dei vari utenti per gli staypoints con rilevanza maggiore (i cluster con dimensione maggiore per il DJ cluster e quelli in cui si è trascorso più tempo per Curve Extrema) non si scostano per più di una decina di metri dalle posizioni

impostate come *ground truth*. Nello specifico è stata calcolata media e varianza per un utente campione sui centroidi dei cluster creati con DJ Cluster con dimensione maggiore per 5 esecuzioni diverse, le distanze in metri dalla posizione indicata come *ground truth* sono:

$$[\ 4.18,\ 5.98,\ 6.13,\ 6.24,\ 6.68\ ]$$
$$\text{con media } \mu = 5.84$$
$$\text{e varianza } \sigma^2 = 0.74$$

Per quanto riguarda invece i valori di media e varianza per gli staypoint ottenuti dall'elaborazione di Curve Extrema in cui si è trascorso più di 160 mila secondi consecutivi (2 giorni) sono:

$$[\ 10.52,\ 10.52,\ 10.52,\ 10.52,\ 11.73,\ 21.85]$$
$$\text{con media } \mu = 12.61$$
$$\text{e varianza } \sigma^2 = 17.2$$

3.4.2 Casi d'uso

Esempio di esecuzione dell'applicazione Python in cui è specificato il file di configurazione.

```
PS C:\trajectory> python -m src.main.python.algorithm.runner .\default_config.py
```

Di seguito vengono riportati alcuni esempi significativi di utilizzo del portale:

- Nella figura 3.19 è mostrato il layer che grafica i dati grezzi di un utente. Questi sono i dati che ancora non hanno subito il processo di enrichment, ogni puntino sulla mappa corrisponde ad una posizione registrata.
- Nella figura 3.20 è mostrata la suddivisione in segmenti delle traiettorie di un determinato utente, questo layer mostra le traiettorie che hanno subito il processo di enrichment, diventando segmenti visualizzabili sulla mappa. La colorazione indica la velocità media del segmento.
- Nella figura 3.21 sono mostrati i pattern *AbitaIn* e *LavoraIn* elaborati attraverso l'algoritmo Curve Extrema.

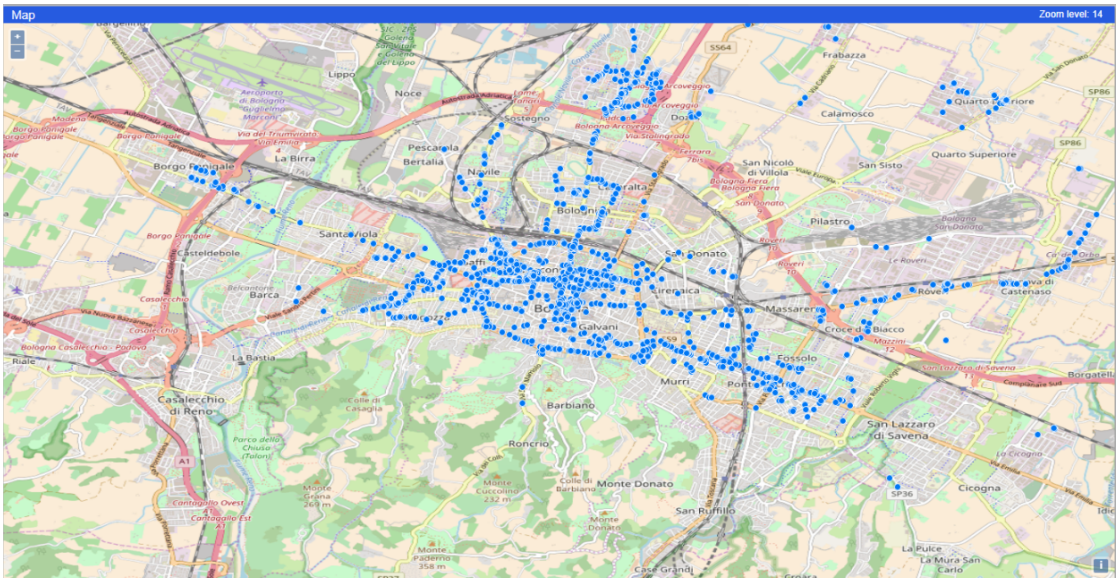


Figura 3.19: Layer che mostra i dati grezzi di un utente.

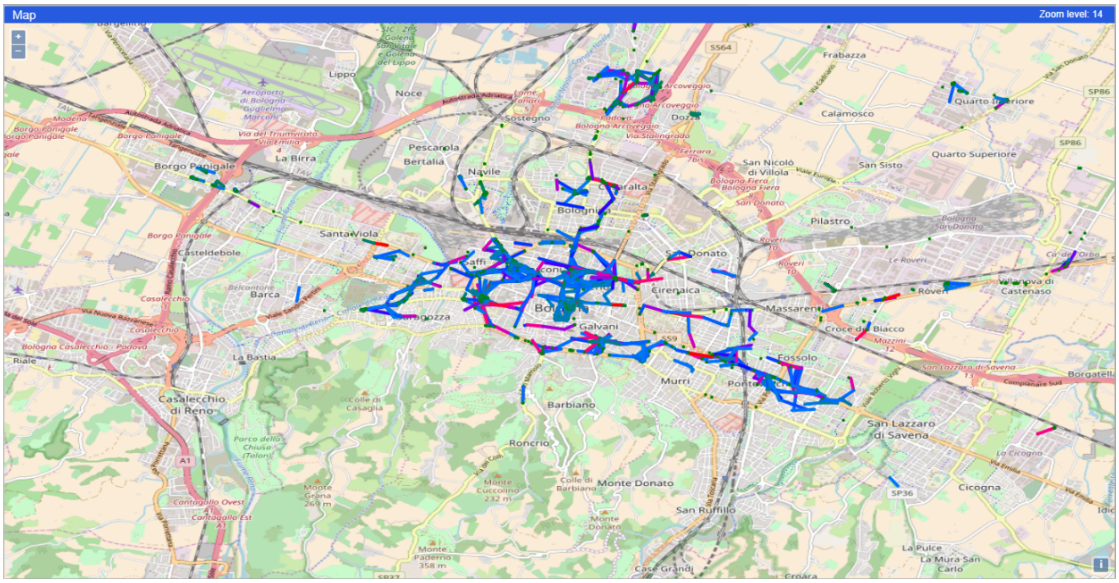


Figura 3.20: Layer che mostra la segmentazione dei dati di un utente.

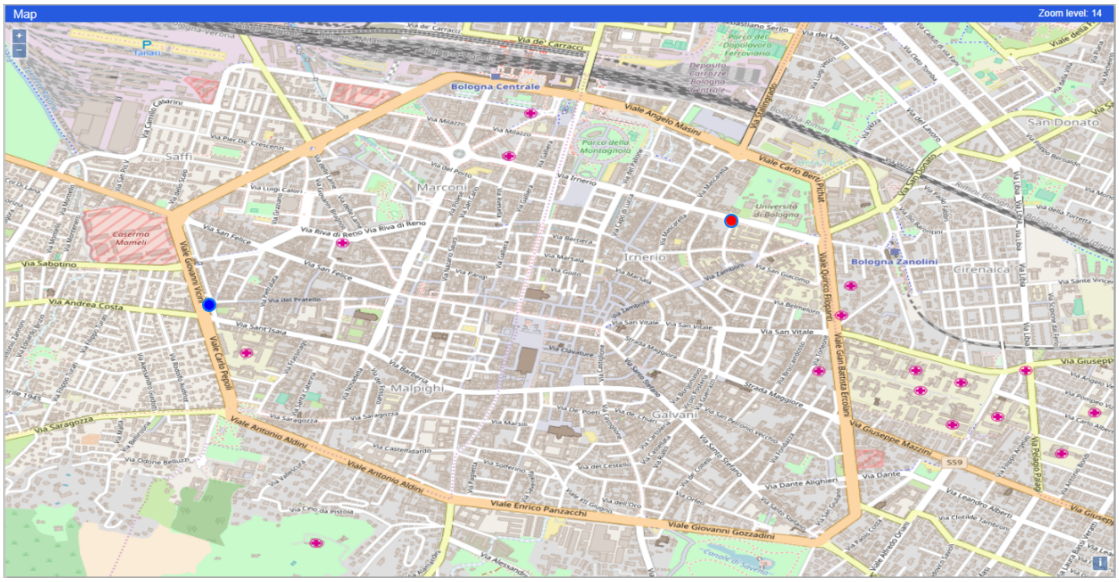


Figura 3.21: Layer che mostra i pattern AbitaIn e LavoraIn di un utente.

3.5 Note di sviluppo

Testing automatizzato

Parallelamente allo sviluppo si sono implementati tutti i test possibili (Test Driven Development), verificandone il funzionamento sia manualmente che attraverso le pipeline di Bitbucket. Infatti è stato impostato nel file di configurazione *bitbucket-pipelines.yml* la linea di comando per eseguire tutti i moduli di test. Più in generale si è seguita la filosofia

Analysis a bit, design a bit, code a bit, test what you can.
Test early, test often, test enough.

Prof. A.Natali

Sono stati creati nuovi test ogni qual volta venisse sviluppato un nuovo componente oppure appena corretto un bug veniva creato un nuovo test di regressione. I test sono stati eseguiti tramite il modulo *unittest* di Python. Il numero di test è in costante aumento con l'incremento del progetto e la parte testata in maniera più intensa è la parte algoritmica.

Flusso delle eccezioni

Entrambi gli algoritmi sono stati sviluppati con particolare attenzione alla gestione del flusso delle eccezioni. Questo infatti ha permesso di rendere più efficace la scoperta di problemi (ad esempio il caso di un esecuzione che avrebbe portato ad un risultato nullo) e interrompere preventivamente l'elaborazione notificando quanto avvenuto nel file di log.

Pip

Data la grandissima diffusione di Python è presente un enorme quantità di librerie già sviluppate e facilmente ottenibili tramite lo strumento integrato **pip**. Tra le librerie che sono state utilizzate sono presenti alcune per la manipolazione dei dati (Pandas e Numpy), altre che implementano funzioni di calcolo della distanza (Haversine), altre per la graficazione (Matplotlib) oppure per il dialogo con il database (Psycopg2). Tutte queste librerie sono state inserite tra i requisiti.

Capitolo 4

Conclusioni e sviluppi futuri

L'obiettivo principale di questo progetto è stato sviluppare uno strumento che permettesse di elaborare le tracce di movimento grezze raccolte. Per prima cosa si è proceduto al processo di enrichment aggiungendo ai dataset le informazioni utili ai fini delle elaborazioni. Successivamente sono stati applicati gli algoritmi ai dati arricchiti, i risultati di queste elaborazioni sono verificabili attraverso il sito web che è stato sviluppato espressamente per questo scopo.

I risultati conseguiti al completamento del progetto sono molteplici: in primo luogo si è notato che risalire agli indirizzi di casa e di lavoro approssimativi è estremamente semplice anche attraverso un'analisi visuale (graficando i dati sulla mappa interattiva) sui dati grezzi, infatti applicando un semplice filtro per fascia oraria si nota subito dove la concentrazione di posizioni durante la notte oppure durante le ore lavorative è maggiore. Inoltre si è notato che i risultati degli algoritmi, sebbene non eseguiti su dati con un'accuratezza non molto elevata, sono migliori del previsto. In sostanza il numero elevato di posizioni ha compensato la scarsa accuratezza essendo il centroide del cluster una media pesata delle posizioni. In conclusione si può affermare che il progetto sia stato portato a termine con successo, raggiungendo gli obiettivi prefissati.

Un aspetto emerso da queste analisi che non va sicuramente trascurato è quella della privacy. Infatti anche se i dataset ci sono stati forniti in forma anonima è possibile risalire al proprietario delle posizioni incrociando informazioni disponibili.

I dati inoltre permetterebbero di studiare le abitudini degli utenti, rendendo del tutto inesistente il concetto di "dati anonimi"

Un possibile sviluppo futuro di questo progetto potrebbe essere la trasformazione del sito web in un portale di consultazione per gli utenti che caricano i propri dati. Se si mettessero a disposizione degli strumenti per eseguire query SOLAP del tipo "Quali sono i luoghi che le persone visitano dopo essere state in questo negozio a Bologna" o "Quali luoghi ho visitato per più di un ora nella mia città questo mese" gli utenti sarebbero incentivati a condividere i propri dati e i dataset a disposizione per avviare altri studi crescerebbero rapidamente. Oppure si potrebbero importare nuovi dataset per verificare come gli algoritmi si sposano con diversi valori di accuratezza e rumore.

Infine questo progetto è stato collocato all'interno del progetto di ricerca "Modelling social behaviours from trajectory" presso il Business Intelligence Group il cui obiettivo sarà quello di estendere il lavoro con arricchimento semantico ed estrazione dei profili utenti del sistema.

Bibliografia

[1] GUDMUNDSSON J. LAUBE P. ANDERSSON, M. and T. WOLLE. Reporting leaders and followers among trajectories of moving point objects. 2008.

[2] MARUYAMA K. SATO A. ASAHARA, A. and K. SETO. Pedestrian-movement prediction based on mixed markov-chain model. 2011.

[3] Starner T. Ashbrook, D. Learning significant Locations and Predicting User Movement with GPS. *Proceedings of IEEE Sixth Internatuibal Symposium on Wearable Computing*, 2002.

[4] GUDMUNDSSON J. HÜBNER F. BENKERT, M. and T. WOLLE. Reporting flock patterns. 2008.

[5] DODGE S. BUCHIN, M. and B. SPECKMANN. Similarity of trajectories taking into account geographic context. *Journal of Spatial Information Science 9*, pages 101—124, 2014.

[6] PEREIRA F. C. LORENZO G. D. LIU L. CALABRESE, F. and C. RATTI. The geography of taste: Analyzing cell-phone mobility and social events. 2010.

[7] BACKSTROM L. COSLEY D. SURI S. HUTTENLOCHER D. CRANDALL, D. J. and J. KLEINBERG. Inferring social ties from geographic coincidences. 2010.

[8] F. P. DA SILVA and R. FILETO. A method to detect and classify inconsistencies of moving objects' stops with requested and reported tasks. *Journal of Information and Data Management 6*, pages 71—-80, 2015.

[9] LAUBE P. DODGE, S. and R. WEIBEL. Movement similarity assessment using symbolic representation of trajectories. *International Journal of Geographical Information Science 26*, pages 1563—-1588, 2012.

[10] WEIBEL R. DODGE, S. and A.K. LAUTENSCHÜTZ. Taking a systematic look at movement: Developing a taxonomy of movement patterns. 2008.

[11] Martin Ester, Hans P Kriegel, Jorg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[12] ROSSI G. P. GAITO, S. and M. ZIGNANI. From mobility data to social attitudes: A complex network approach. 2011.

[13] NANNI M. PEDRESCHI D. PINELLI F. RENSO C. RINZIVILLO S. GIANNOTTI, F. and R. TRASARTI. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *The VLDB Journal 20, 5*, 2011.

[14] NANNI M. PINELLI F. PEDRESCHI D. GIANNOTTI, F. Trajectory pattern mining. *KDD '07: Proc. 13th International Conference on Knowledge Discovery and Data Mining*, 2007.

[15] Kuijpers B. Moelans B. Vaisman A. Gomez, L. A Survey on Spatio-Temporal Data Warehousing. *nternational Journal of Data Warehousing and Mining*, 2009.

[16] R. Kimball. The data warehouse toolkit. 1996.

[17] Ross M. Kimball, R. The data warehouse toolkit: The complete guide to dimensional modeling. 2002.

[18] Miao Lin and Wen Jing Hsu. Mining GPS data for mobility patterns: A survey. *Pervasive and Mobile Computing*, 12:1–16, 2014.

[19] BIDERMAN A. LIU, L. and C. RATTI. Urban mobility landscape: Real time monitoring of urban mobility patterns. 2009.

[20] JD Mazimpaka and S Timpf. Trajectory data mining-A review of methods and applications. *Journal of Spatial Information Science*, 13(13):1–45, 2015.

[21] Kuper G. Libkin L. Paredaens, J. Constraint databases. 2000.

[22] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4):42:1–42:32, 2013.

[23] Bédard Y. Marchand P. Rivest, S. Towards better support for spatial decision making: Defining the characteristics of spatial online analytical processing (SOLAP). *Geomatica*, 2001.

[24] PARENT C. DAMIANI M.L. MACEDO J.A. PORTO F. SPACCAPIETRA, S. and C. VANGENOT. A conceptual view on trajectories. *Data and Knowledge Engineering 65*, pages 126—-146, 2008.

[25] Georgios Stylianou. Stay-point Identification as Curve Extrema. 2017.

[26] WOLFSON O. VAZIRGIANNIS, M. A Spatiotemporal Model and Language for Moving Objects on Road Networks. pages 20–35, 2001.

[27] NEUTENS T. DELAFONTAINE M. VERSICHELE, M. and N. V. DE WEGHE. The use of bluetooth for analyzing spatiotemporal dynamics of human movement at mass events: A case study of the ghent festivities. *Applied Geography 32*, pages 208—-220, 2012.

[28] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. Discovering personal gazetteers. *Proceedings of the 12th annual ACM international workshop on Geographic information systems - GIS '04*, page 266, 2004.

Ringraziamenti

Un ringraziamento speciale va alla mia famiglia, papà, mamma, Maddalena e Michela, a voi che mi avete da sempre spronato a seguire la mia strada, a tutti gli amici che mi hanno supportato (e sopportato) in questi anni, spronandomi a fare di meglio (o di peggio a seconda dei punti di vista, forse è il caso che quell'annuncio lo metta in vetrina). Vorrei ringraziare anche il prof. Golfarelli per il bellissimo progetto a cui mi sono potuto dedicare e il dott. Matteo Francia correlatore, ma soprattutto amico, con cui ho condiviso gran parte di questo lavoro.