BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

TRAJECTORY DATA PREPROCESSING FRAMEWORK FOR DISCOVERING SEMANTIC LOCATIONS

RÁMEC PRO PŘEDZPRACOVÁNÍ DOPRAVNÍCH DAT PRO OBJEVOVÁNÍ SÉMANTICKÝCH TRAJEKTORIÍ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **ANNA OSTROUKH**
AUTOR PRÁCE

SUPERVISOR **Ing. MAZEN ISMAEL, MSc.**
VEDOUCÍ PRÁCE

BRNO 2018

Master's Thesis Specification/19631/2017/xostro00

Brno University of Technology - Faculty of Information Technology

Department of Information Systems | Academic year 2017/2018

Master's Thesis Specification

For: **Ostroukh Anna, Ing.**
Branch of study: Information Systems
Title: **Trajectory Data Preprocessing Framework for Discovering Semantic Locations**
Category: Data Mining

Instructions for project work:

1. Study the existing methods and algorithms for preprocessing trajectories with focus on semantic trajectories.
2. Design and framework for discovering semantic locations.
3. Implement the designed method.
4. Evaluate the implemented method on the appropriate datasets based on the consultations with the tutor.
5. Summarize the results and usability in other future research.

Basic references:

- Zheng, Yu, Zhou, and Xiaofang (Eds.): Computing with Spatial Trajectories, Springer, 2011.
- Choi, Pei: Efficient Mining of Regional Movement Patterns in Semantic Trajectories, VLDB Endowment, 2017.
- Palma, A. T: A clustering-based approach for discovering interesting places in trajectories. In Proceedings of the 2008 ACM Symposium on Applied Computing. ACM. 2008. pp. 863-868.
- Zheng, Y.: Trajectory Data Mining: An Overview. ACM Transaction on Intelligent Systems and Technology. September 2015. Retrieved from: https://www.microsoft.com/en-us/research/publication/trajectory-data-mining-an-overview/

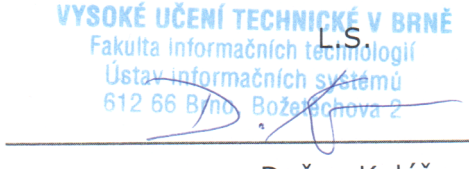
Requirements for the semestral defence:
Items 1, 2.

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Master's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Ismael Mazen, Ing., MSc**, DIFS FIT BUT
Beginning of work: November 1, 2017
Date of delivery: May 23, 2018



L.S.

Dušan Kolář
Associate Professor and Head of Department

Abstract

The main goal of this thesis is to study existing approaches for trajectory data preprocessing with a focus on semantic trajectories, discovery and development of a framework, which integrates trajectory data from GPS sensors with semantics. The problem of raw trajectories analysis is that it cannot be as comprehensive as the analysis of trajectories containing meaningful context. The study of different approaches and algorithms are followed up by the design and implementation of the framework, which finds semantic locations by application of a density-based clustering algorithm on trajectories stops. The proposed framework is evaluated on real datasets containing raw GPS records.

Abstrakt

Cílem práce je vytvoření přehledu o existujících přístupech pro předzpracování dopravních dat se zaměřením na objevování sémantických trajektorií a návrh a vývoj rámce, který integruje dopravní data z GPS senzorů se sémantikou. Problém analýzy nezpracovaných trajektorií spočíva v tom, že není natolik vyčerpávající, jako analýza trajektorií, které obsahují smysluplný kontext. Po nastudování různých přístupů a algoritmů sleduje návrh a vývoj rámce, který objevuje semantická místa pomocí schlukovací metody založené na hustotě, aplikované na body zastavení v trajektoriích. Návrh a implementace rámce byl zhodnotěn na veřejně přístupných datových souborech obsahujících nezpracované GPS záznamy.

Keywords

Semantic trajectories, traffic data preprocessing, semantics enrichment.

Klíčová slova

Sémantické trajektorie, předzpracování dopravních dat, integrace semantiky

Reference

OSTROUKH, Anna. *Trajectory data preprocessing framework for discovering semantic locations.* Brno, 2018. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Mazen Ismael, MSc.

Rozšířený abstrakt

Cílem práce je vytvoření přehledu o existujících přístupech pro předzpracování dopravních dat se zaměřením na objevování sémantických trajektorií a návrh a vývoj rámce, který integruje dopravní data z GPS senzorů se sémantikou. Diplomová práce na začátku vysvětluje důležité pojmy, jako například co je trajektorie a trajektorie integrovaná se sémantikou. Dále popisuje problém analýzy nezpracovaných trajektorií. Poté práce věnuje pozornost obecným metodám předzpracování dopravních dat, zaměřených na vylepšení kvality (redukce šumu a odlehlých hodnot) a redukci výpočetní náročnosti (komprese dat) pro budoucí analýzu databáze s trajektorií. Značná část je věnována studiu shlukovacích algoritmů, jelikož jsou základem přístupu pro objevování sémantických míst. Práce uvádí příklady použití sémanticky zpracovaných dopravních dat za účelem dotazování a získání znalostí, které nejsou dostupné pro nezpracovaná dopravní data.

Po úvodním nastudování obecných metod pro předzpracování dat práce dále sleduje studium přístupů, které se používají pro objevování sémantických míst z nezpracovaných dopravních dat. Objevování sémantických míst se dělí na dvě etapy: objevování fyzických lokací neboli označení míst zájmu, pro které chceme získat sémantický kontext, a integrace se sémantickým kontextem za použití veřejně přístupných geografických databází. Detailně jsou popsány shlukovací metody založené na hustotě, které se dají aplikovat pro automatické objevování fyzických lokací, a následně jsou popsané přístupy pro integraci objevených takzvaných bodů zájmu se sémantickým kontextem. Mezi ně patří sémantický kontext popisující určité chování a geograficky sémantický kontext.

Diplomová práce se zabývá návrhem rámce, který objevuje fyzická místa pomocí časoprostorové shlukovací metody ST-DBSCAN, založené na hustotě a aplikované na segmentované podle bodů zastavení trajektorie. Poté vrací takzvaný obdélník vazeb pro každý shluk. Tento obdélník se používá pro dotazování veřejně přístupné geografické databáze OpenStreetMap za účelem objevování sémantických míst a následnou integraci příslušných bodů zastavení ve shluku se sémantikou. Práce také navrhuje použití metody OPTICS pro heuristické vyhodnocování shlukovacích parametrů za pomoci grafu dosažitelnosti.

Práce dále uvádí popis vývoje navrženého rámce a to s použitím programovacího jazyka Java a principů Objektově orientovaného programování (OOP). Uživatelské rozhraní vyvinuté aplikace je rozděleno na segmentaci trajektorií, evaluaci shlukovacích parametrů a shlukování a integraci se sémantikou. Následný popis vysvětluje architekturu rámce, vyplněnou v souladu s MVC modelem a RESTful services. V práci jsou uvedené implementační detaily použitých algoritmů OPTICS, ST-DBSCAN a modulu zodpovědného za odeslání požadavků do OpenStreetMap databáze se zpracováním zpětné vazby.

Implementace rámce byla zhodnocena na dvou veřejně přístupných datových souborech obsahujících nezpracované GPS záznamy: T-drive dataset od Microsoft Research, obsahující trajektorie vozů taxi v Beijingu a dále dataset Berlin MOD obsahující simulované trajektorie vozidel pohybujících se na dopravní síti v Berlíně. Cílem shlukovaní je objevení seskupení stojících vozidel na malé ploše v relativně shodný časový okamžik. V obou případech ukázalo implementované vyhodnocení shlukovacích parametrů na grafu dosažitelnosti OPTICS dobrou použitelnost pro nalezení nejbližší možné prostorové a časové vzdálenosti pro počet vozidel, pro které je shlukování proveditelné. Pro všechny nalezené shluky v obou datasetech byly objevené sémantické lokace a to i s tím, že chování objektů v datasetu s taxíky a datasetu se simulovanými trajektoriemi jsou odlišné. Pro dataset s taxíky byly objeveny zejména názvy cest a ulic, kde se obvykle taxíky pohybují. V datasetu se simulovanými trajektoriemi byly nalezeny názvy budov se správnými adresami a parkovací místa.

V závěru diplomové práce se uvádí možnosti praktického využití vyvinutého rámce a to především na etapě předzpracování dat. Rámec může sloužit jako podpůrný nástroj před provedením datové analýzy, která vyžaduje sémantické trajektorie jako vstupní data.

Trajectory data preprocessing framework for discovering semantic locations

Declaration

I hereby declare that this Master's thesis is my own work created under the supervision of Ing. Mazen Ismael and that the bibliography contains all the literature that I have used in writing the thesis.

.....
Anna Ostroukh
May 15, 2018

Acknowledgements

I would like to express my gratitude to my supervisor Ing. Mazen Ismael for his assistance and cooperation during the work on this thesis.

Contents

Chapter 1

Introduction

Nowadays there is a fast development of mobile communication technologies along with global positioning and navigational systems. Positioning services are provided by such well-known technologies as GPS (Global Position Systems), GSM (Global System for Mobile Communications), RFID (Radio Frequency Identification) and many others. As a consequence a big amount of spatial data is obtained by acquisition systems and used in different areas like traffic analysis, security management, location based services, anomaly detection, etc.

The data obtained from sensors or other collecting devices contain raw, unprocessed data point sequences in geographic space mapped with time and object identity and do not contain any context about the particular location they belong to. By context in this work connotes a meaningful description of places like streets and building names. Knowing the context can be critical for some mining tasks as mining frequent patterns of moving objects for the objects' location prediction or improving location-based services (advertisement, advisers, etc.).

The aim of this thesis is to perform a research of existing methods and algorithms of trajectory data preprocessing for semantic data analysis and data mining of traffic data. Design and implementation of an algorithm for trajectory data preprocessing in order to discover semantic locations. The ST-DBSCAN algorithm is utilized for the semantic location discovery as a density-based clustering approach which allows clustering in both, spatial and temporal dimensions. The follow up semantic enrichment of locations performed with utilization of the online GIS database OpenStreetMap through the Overpass API and the Nominatim API.

In chapter 2 an overview of semantic trajectories and main definitions are provided. The chapter also deals with the general overview of trajectory preprocessing techniques and basic mining tasks on trajectories as well as a specific type of mining tasks on trajectories with semantics. Chapter 3 describes basic steps, existing methods and algorithms for trajectory data preprocessing for semantics. Chapter 4 contains the design of trajectory data preprocessing framework for semantic enrichment of trajectories. The following Chapter 5 provides the design and implementation details of the proposed solution. The final Chapter 6 evaluates implemented framework on real datasets.

Chapter 2

Semantic trajectories and preprocessing methods of moving objects data

This chapter gives main definitions for trajectory and semantic trajectory which will accompany this work hereinafter. The chapter also provides an overview of general trajectory data preprocessing techniques and common data mining tasks which are performed on trajectories as well as on semantic trajectories.

2.1 Semantic trajectories

The most common way of gathering information about a moving object is using small GPS loggers or GPS-equipped mobile phones. This information is represented as GPS-records which are a sequence of time-stamped latitude/longitude points shaping a GPS-trajectory. An example of such GPS-trajectory is shown in Figure 2.1.

Figure 2.1: Example of a GPS-trajectory with inset showing individual points [14]

Definition 2.1. ***GPS point*** p is a pair $p = (lng, lat)$, representing the longitude-latitude location and corresponding to a unique point in the geographical space.

Definition 2.2. ***GPS record*** G is a tuple $G = < u, t, p, s >$ where u is the ID of the moving object for which G is recorded, t is a timestamp, p is a GPS point, and s is a vehicle's speed as reported by the GPS device [10].

Definition 2.3. ***GPS trajectory*** T is a sequence of GPS records $T = < (G_0, ..., G_n)>$ ordered by their timestamps so that $t_k < t_{k+1}$.

The problem in data representation of a moving object as GPS trajectory is that it does not contain any meaningful information which is easy understandable by humans. Comprehension of the context of particular sets of points from a trajectory may significantly improve certain mining tasks. For example, instead of dealing with a sequence of points we can define mining tasks or perform analysis with respect of certain semantics as street names, buildings, parking spots, etc. In a general meaning, semantic trajectory is a trajectory where GPS location points are associated with semantic entities. Semantic entities can be different depending on the application. For example, for touristic applications as semantic entities attractions, restaurants and other popular touristic destinations can be used. When considering traffic study, semantic entities usually refer to a road and street names, buildings, parking spots, etc.

In this work as an application area we consider traffic study and by semantic trajectory we will define the data model where physical location is integrated with geographic information.

Definition 2.4. ***Semantic place*** P_s is a pair $P_s = <(p_0, ..., p_k), C_0>$ where p_n is a point from physical location and C_m is a geographic information (street name, building, etc.).

Definition 2.5. ***Semantic trajectory*** ST is a trajectory which has been enhanced with annotations and therefore contains semantic places $T_s = <(P_{s_1}, (t_0, ..., t_n)), ..., (P_{s_k}, (t_0, ..., t_m))>$. For each semantic place its starting/ending instants are known.

2.2 Trajectory data preprocessing techniques

Data preprocessing is a basic and very first step of any mining task which aims at improving the quality of the trajectory database. Trajectory data may contain noise caused by sensors and other factors which can be eliminated by noise reduction techniques [24]. High sampling rates in acquiring location points cause high data load which significantly reduces computation performance. Appropriate data compression techniques can improve the efficiency.

Clustering trajectories, segmentation and integrating semantics to trajectories may be also a part of the preprocessing step, but as this is the key topic of this thesis, bigger attention to it is given in chapter 3. This chapter contains a quick overview of basic clustering algorithms for better understanding their modifications described in chapter 3.

2.2.1 Noise reduction

Data cleaning is performed for identifying and eliminating missing, inconsistent or incomplete data points from a database. For example, there is no sense in studying trajectories which contain too few location points or trajectories containing noise and outliers can lead to errors in future computations. Trajectories containing few points can simply be removed from the studying dataset and outliers as well as noise can be eliminated or smoothed.

Mean and median filters

Mean and median filters are simple filters which are used for smoothing noise in trajectory location points. These filters use previously measured values to approximate the estimated value. The amount of previously measured points is set by the user defined sliding window. The mean filter is sensitive to outliers therefore the approximated curve will not be smooth enough in places where the outlier appeared. A better solution is to use the median filter as it is robust to outliers and provides better smoothing.

The main disadvantage of these filters is that they can perform measurements just for the spatial dimension omitting the temporal part.

Kalman and Particle filter

The Kalman filter is a more advanced method and considers other dimensions like speed or acceleration. The filter is able to model the measurement noise and the dynamics of the trajectory. The Kalman's filter prediction of the estimated value is based on measurements of speed, distance and acceleration but also takes into account simple laws of physics like gravity.

The Particle filter is similar to the Kalman filter. The Kalman filter gains efficiency by matrix multiplication plus Gaussian noise whereas the Particle filter uses a less efficient algorithm and therefore is computationally less effective.

Outlier detection

Previously mentioned methods substitute noise in trajectory by an estimated value. The outlier detection method removes noisy points from the trajectory. This algorithm computes the travel speed of each point in a trajectory based on the time interval and distance between a point and its successor. If the computed interval has a speed larger than a threshold, it is removed from the trajectory.

2.2.2 Data compression

Data compression methods are used for reduction of the communication and storage overhead of trajectory data representation. The main idea of compression algorithms is to keep the data precision in a new, compressed trajectory. Reduction techniques are grouped into two categories: offline and online data reduction. Offline data reduction techniques collect the full dataset of location data points and then compress the data by discarding redundant points. Online techniques work during the data points collection. For every obtained new data point it has to be determined whether it can be preserved in a trajectory.

Distance metric

Distance metrics are used in compression algorithms as an error measuring metric. In Figure 2.2 perpendicular Euclidian distance for the compressed trajectory of 12 points into a representation of three points $p_1,\ p_7,\ p_{12}$ is illustrated. The distance metric is the summation of the lengths of the segments connecting p_i and p'_i. Therefore this error measure takes the geometric shape of the trajectory into account.

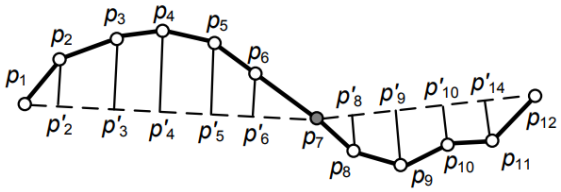


Figure 2.2: Perpendicular Euclidian distance [24]

Offline compression

For offline compression a trajectory with a full set of location points is given. The compression algorithm generates an approximated trajectory by removing points with errors from the original trajectory. Following some algorithms are described:

The **Douglas-Peucker** algorithm attempts to replace the original trajectory by an approximate line segment. The replacement has to meet the defined error metric requirement otherwise the algorithm recursively splits the segment into two sub-segments choosing the point with biggest error as a splitting point (Figure 2.3). The stop condition for this recursive algorithm is met when the error between the original and approximated trajectory is below a threshold.

The Douglas-Peucker algorithm is widely used, but there are a couple of other modifications of it. The **Top-down time-ratio** algorithm uses an error metric which takes the time dimension into account, whereas the **Bellman**'s algorithm applies a dynamic programming technique to ensure that the approximated trajectory is optimal.

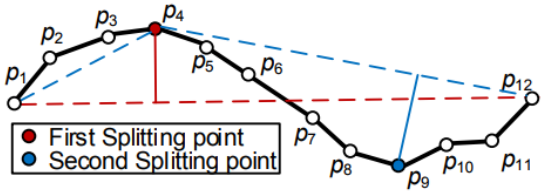


Figure 2.3: Douglas-Peucker algorithm [24]

Online compression

Online compression algorithms do computations during the data acquisition and have to ensure the provision of efficient on-line decisions when a new location point is obtained and thus decide, whether it can be present in a trajectory. Those methods can be divided into two major categories: methods based on a window and methods based on the speed and direction of a moving object.

The **Sliding window** algorithm in the beginning initializes the first location point and starts growing the sliding window by including the next point. After adding a new point, the algorithm checks on fulfilment of the distance error threshold by computing all distance errors for all the location points in a window against the potential approximated line segment. The algorithm continues the growth of the window if the check is positive. Otherwise the last valid line segment is included as a part of approximated trajectory and the next point is set as the next first location point.

The **Open window** algorithm uses the Douglas-Pecker method for identification of the point with the maximum error in the window. This point then is used as a first location point to approximate the next line segment of the trajectory.

Speed and direction based algorithms are based on prediction of the incoming location point from computed speed and direction. One of such algorithms is the *threshold-guided sampling* algorithm. This method defines a safe area which is derived from the last two location points and by a given thresholds defines whether a newly obtained point has some significant changes in direction or speed. If the new data point is located within the safe area, such point is considered as redundant and is removed from the approximated trajectory.

2.2.3 Data clustering

Clustering is a way to organize a dataset into groups by objects' similarity. A cluster contains objects which are similar to each other and dissimilar to objects of another cluster.

In this subsection basic clustering methods are described. These clustering algorithms are divided into four major [23] groups.

Partitioning algorithms

Partitioning algorithms divide the dataset into k partitions. The number of partitions is defined by the user and partitioning is performed using some evaluation criteria. The most well-known algorithm is the *k-means* which randomly divides the dataset into k partitions, computes a centroid of each partition and assigns all objects with the closest centroid to it. The algorithm continues calculating centroids and assigns objects till all centroids stay unchanged.

The main disadvantage of this category of algorithms is that the user has to define the number of clusters as an input parameter which requires the knowledge of the approximate data distribution in the dataset. Another drawback is that partitioning algorithms are not robust to outliers.

Hierarchical clustering

Hierarchical algorithms group objects into a tree of clusters. There are two approaches: bottom-up and top-down. The bottom-up approach merges objects starting from leaves where they are represented by single clusters. The criterion of merging clusters is defined by a specified measure of cluster proximity.

The top-down approach has in the beginning all objects in a single cluster. They are recursively split by a certain measure of similarity like in the bottom-up approach.

Hierarchical clustering algorithms handle noise and outliers better than partitioning algorithms, but there is still the requirement to provide a number of clusters and stopping criteria.

Density-based clustering

Density-based clustering algorithms group objects in clusters by identification of dense regions. Identification of dense regions is based on the definition of a neighbourhood radius for an object and a minimal number of neighbour objects to shape a cluster. The most well

known algorithm is DBSCAN [11] and its extension OPTICS [6] which does not perform clustering but describes objects ordering in the dataset.

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

DBSCAN [11] is a density-based algorithm which does not require a number of clusters as an input and uses notions of density and noise. The key idea is that for each point of a cluster the neighbourhood of a given radius has to contain at least a minimum number of points($MinPts$), i.e. the density has to exceed a certain threshold. DBSCAN uses notions of $\varepsilon - neighbourhood$ and $density - reachability$ defined as following:

Definition 2.6. ***ε-neighbourhood*** of a point p is defined as

$$N_\varepsilon(p) = \{q \in D | dist(p,q) \leq \varepsilon\}, \tag{2.1}$$

where $dist(p,q)$ is the distance between point p and q and D is a database of points.

Definition 2.7. ***Density-reachability*** of a point p from a point q is when a chain of points $p_1, ..., p_n$, $p_1 = q, p_n = p$ exists such that p_{i+1} is in the ε-neighbourhood of q (N_ε(q)) and N_ε(q) satisfies a certain density threshold defined as $MinPts$.

To find a cluster, DBSCAN starts with an arbitrary point p from the database satisfying the threshold of $MinPts$ and retrieves all points that are density-reachable from p obtaining the cluster containing p as a core point. The input parameters of ε and $MinPts$ for the DBSCAN algorithm are defined globally for the whole dataset and therefore two clusters may be merged into one if they are close enough to each other.

OPTICS: Ordering Points To Identify the Clustering Structure

The OPTICS [6] algorithm does not perform explicit clustering of the dataset but instead creates an ordering for the dataset which represents a density-based clustering structure. The main feature of this algorithm is that it does not use global input parameters as other clustering algorithms do. The authors of OPTICS consider the approach of using global parameters for the whole dataset as incorrect as clusters in real-datasets do not have homogeneous intrinsic structure (have different local densities) and therefore some clusters cannot be revealed with global parameters. The basic idea of this algorithm is to produce an ordering of the dataset which contains the information about intrinsic structure of every cluster in it and which is easy to analyse.

OPTICS utilizes the idea of the DBSCAN algorithm and stores the order in which the particular data point is processed when expanding a cluster and the information which DBSCAN would use to assign the data point to a particular cluster. This information consists of two values for each object: the *core-distance* and a *reachability-distance* (graphically represented in Figure 2.4).

Definition 2.8. ***Core-distance*** of an object p is the smallest distance ϵ' between p and an object in its ϵ-neighbourhood such that p would be a core object with ϵ'-neighbourhood which contains at least $minPts$ objects.

Definition 2.9. ***Reachability-distance*** of an object p from core object o is the minimum radius value that makes p density-reachable from o.

As we see from the definitions of core-distance and reachability-distance, this information is sufficient for extracting all density-based clusters with a ϵ'-neighbourhood radius

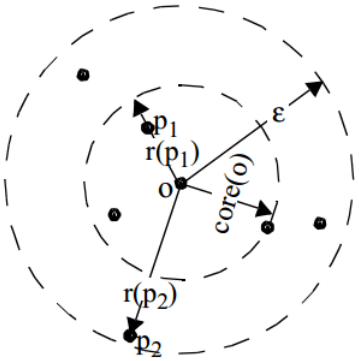


Figure 2.4: Core distance (o), reachability-distances r(p_1, o), r(p_2, o) for $minPts = 4$ [6]

smaller than the initially defined ϵ-neighbourhood radius on the DBSCAN input. This algorithm helps to reveal small nested clusters inside of big ones as shown in Figure 2.5. Clusters A, B and C only will be revealed with a global parameter of ϵ, but ordering the dataset with the OPTICS algorithm will discover the smaller nested clusters C_1, C_2 and C_3 inside of the C cluster as well.

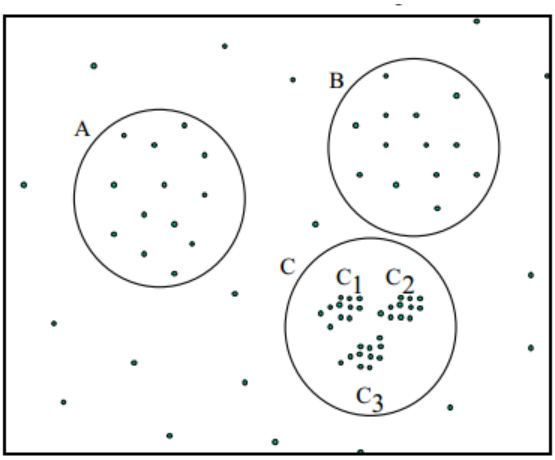


Figure 2.5: Clusters with different density parameters [6]

Grid-based clustering

Grid-based clustering algorithms partition the clustering space into a finite number of grid cells and then performs operations on the partitioned space. Cells containing points more than a certain threshold are marked as dense and then dense cells are connected to form clusters.

2.3 Trajectory data mining tasks

Trajectory data mining tasks try to answer two major questions: prediction and description of a moving object's behaviour. The goal of the prediction task is to determine the future state of objects based on information obtained from the database and the description task has to provide a meaningful interpretation describing a moving objects' behaviour. The most common data mining tasks are described below, the information for this overview of the main approaches is taken from [19], [22].

2.3.1 Clustering

Clustering is a widely used approach for detecting a group of trajectories moving together, sharing the same path or identify outliers.

Trajectory is a two dimensional spatio-temporal data type and therefore traditional clustering methods cannot be applied without modifications. There are attempts to develop trajectory specific clustering methods based on model-based clustering. For example, [12] proposed a method based on a mixture of trajectories' regression models. Each trajectory is represented by a function of time depending on a set of parameters. The algorithm groups trajectories together which likely were generated from a representative trajectory plus Gaussian noise. In another work [1] trajectory is represented as sequences of transitions and a hidden Markov model (HMM) estimates trajectories which best fit to the cluster.

Another approach for clustering trajectories is to extend existing clustering algorithms. Density-based algorithms like DBSCAN and OPTICS described above are widely used as a basis for extension. For example in T-OPTICS [20] the extension is in defining a spatio-temporal distance for clustering trajectories. ST-DBSCAN [8] uses two additional distance ($\varepsilon - neighbourhood$) parameters as input - for spatial and non-spatial attributes.

The choice of a clustering approach and its modification also depends on whether a group of trajectories or a single trajectory has to be clustered. The approach described in [21] finds density regions in a single trajectory as places of interest while the TRACLUS algorithm [16] finds similar groups of trajectories segments.

There are many other noticeable approaches in clustering trajectories [13] like a visual-aided approach based on human expertise [5], micro-clustering methods for grouping segments of trajectories [17], discovering moving clusters [7] and others. In this thesis we will use clustering in the preprocessing stage for discovering places of interest for future semantics integration and will concentrate on density-based clustering algorithms for groups of trajectories with more detailed description provided in chapter 3.

2.3.2 Classification

Classification tasks allow to divide the dataset into predefined classes. Identification of classification rules is based on a training dataset for that it is known to which classes each object from this dataset belongs. According to classification rules obtained during the training phase the remaining dataset is divided into classes.

Classification is applicable for tasks where there is a need to classify a special behaviour or nature of moving objects. For example, classification can divide a trajectories dataset into pedestrians, traffic, cyclists, etc.

Very often trajectory classification is performed after other algorithms like trajectories segmentation or clustering for discovering discriminative features. For example, in work [15]

the authors used segmentation and clustering to extract sub-trajectory and region features that are used afterwards in support vector machine (SVM) classification of the trajectories.

2.3.3 Prediction

Prediction is used for the forecast of future moving objects' location or route. There is a possibility to predict several features of moving objects like location, traffic congestion or traffic jams and route.

Prediction for trajectories is based on two approaches: Markov models and trajectory patterns (sequential rules). Markov models use probabilistic models for location prediction while trajectory patterns mine frequent moving patterns and association rules which are used for prediction.

2.3.4 Non-semantic trajectory knowledge discovery

According to a survey on semantic trajectories [22] for the non-semantic trajectory knowledge discovery the most popular mining task is analysis of collective behaviours. The analysis of collective behaviours can be divided into two categories: the analysis of behaviour patterns of single trajectories or a group of trajectories showing specific interactions.

The first category studies the sequence of regions traversed by the trajectories. The main idea is to discover behaviour which is frequently repeated in particular regions. Regions can be areas defined by the user or discovered by the algorithm based on discovering density areas (containing a minimal number of trajectory positions and frequently visited).

Another one category studies repeating activities of moving objects. The fundamental idea is that periodic behaviours are connected to frequently visited places of interest (POIs) like office, pub, home, etc. Frequently visited places are usually discovered with density-based clustering algorithms and for trajectories the time periods spent within the ROI is detected. Afterwards frequent periods can be identified and behaviours found over these periods.

2.3.5 Semantic-based trajectory knowledge discovery

As was mentioned in chapter 2.1 raw (unprocessed) trajectory data does not contain any application specific context. Semantic information associated with discovered POIs can significantly improve the analysis of behaviour patterns and help to discover meaningful patterns. According to [22] semantic-based behaviour discovery approaches can be divided in two main categories: discovering common behaviour which is previously unknown (e.g. frequent or repetitive pattern mining) and discovering some specific behaviour (e.g. finding stops and moves [3]). Both will be explained in the following paragraphs.

Discovering unknown behaviours

Enhancement of trajectories with geographic context plays a significant role in mining meaningful behaviour of moving objects. In Figure 2.6 on the left side a behaviour pattern is illustrated which can be discovered from raw data. Here we see that objects are moving towards one POI. On the right side, the same pattern enhanced with semantics which shapes a meaningful semantic behaviour 'Going from school to the cinema'.

Semantics cannot only be spatial but also temporal. Temporal semantics means for example, annotating time series with days of the week, month, etc. Annotating trajectories

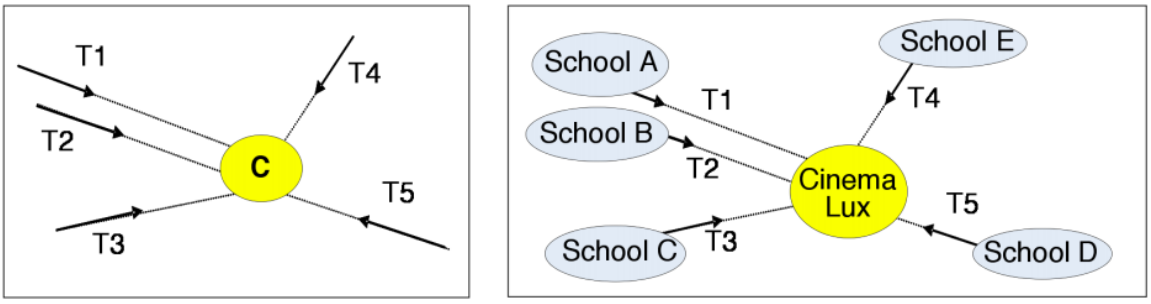


Figure 2.6: behaviour from raw data (left), behaviour with semantic (right) [22]

with spatio-temporal semantics allows the user to query semantic behaviour patterns like 'Going from work to pub on Friday' in the set of semantic trajectories. The ability to query semantic behaviour patterns is developed into the Semantic Trajectory Data Mining Query Language (ST-DMQL) tool [9].

Integration of trajectories with semantics helps to discover behaviour patterns which cannot be discovered from raw data. In the scenario shown in Figure 2.6 existing approaches (e.g. density-based clustering) would discover just a place C while annotating trajectories with context before mining will give a meaningful semantic behaviour.

Discovering specific behaviours

Semantic context does not necessary have to be geographic information. For some applications it is enough to discover places where an object stops, moves, accelerates, i.e. performs a certain kind of activity.

Alvares and others in [4] define avoidance behaviour and present an algorithm for the detection of trajectories that avoid some static object. An avoidance behaviour is defined as a moving of the object towards the target geographic object, turning around without intersecting it and returning to the original route. The same authors in [2] define stop and move activities. The algorithm will be in more detailed described in chapter 3

Chapter 3

Approaches of preprocessing trajectories for integrating semantics

The problem of trajectories data preprocessing for semantic analysis can be divided into two general sub-problems: first, obtaining physical locations which have significance or interest for the analysis (POIs) and second, integrating physical locations with context. Graphically the workflow of a trajectory preprocessing framework is represented in Figure 3.1.

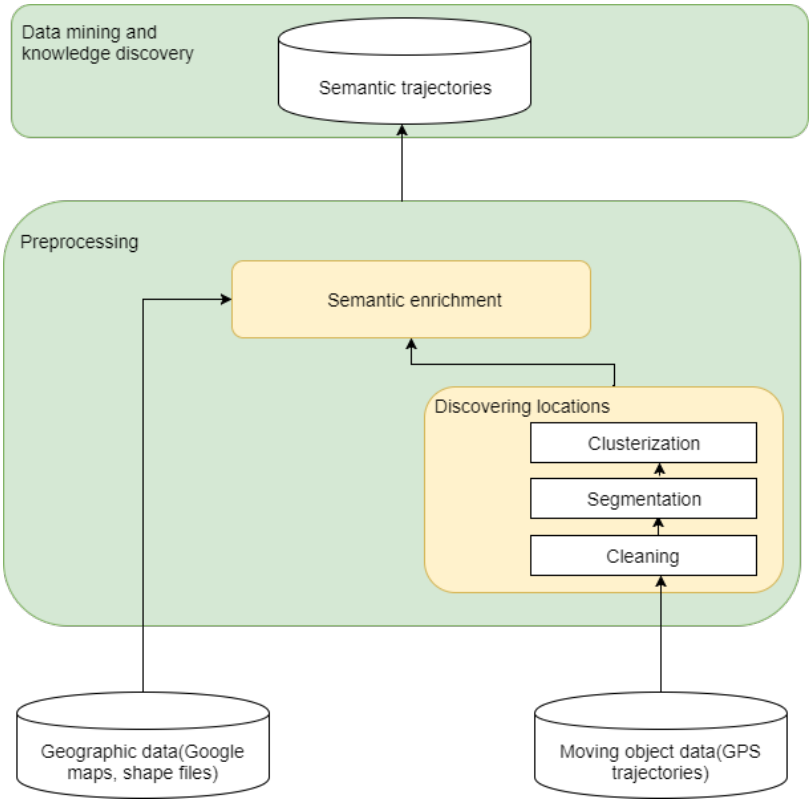


Figure 3.1: Framework for semantic trajectory discovery

This work stands for discovery of physical locations from GPS trajectories. The most commonly used approach is the application of some density-based clustering algorithms. There are several modifications of the well-known DBSCAN and OPTICS algorithms adopted for spatial-temporal data. This chapter gives an overview of the ST-DBSCAN, based on two ε parameters for both data dimensions, the TRACLUS algorithm which is useful for discovering clusters in line segments of trajectories and T-OPTICS which performs spatial clustering for a particular time interval.

Approaches for the semantic enrichment of discovered locations vary depending on the application. In this chapter are presented two main methods - discovering special type of activity, in our case Stops and Moves patterns, and discovering geographic context which is performed with utilization of online GIS (Geographic Information System) databases like Google Maps or OpenStreetMaps.

3.1 Discovering physical locations

There are two ways of obtaining physical locations: i) the user can define spatial regions of interest, ii) automatic discovery of regions from GPS trajectories. In this work methods for the automatic discovery of physical locations will be reviewed.

Discovering regions can be done for a single trajectory as well as for a group of trajectories depending on the type of the semantic context which will be added to the trajectories afterwards. The main goal in the automatic discovery of physical locations is to find places with a high density of objects. The majority of approaches for solving this problem use density-based clustering methods due to their good extensibility.

As was mentioned in chapter 2.3.1, trajectories are a spatio-temporal data type and therefore traditional density-based clustering methods will not give accurate results as they take just the spatial constituent into consideration. This chapter provides an overview of modifications of well-known algorithms which allow clustering spatio-temporal datasets with more accurate results.

3.1.1 ST-DBSCAN: Spatial-Temporal DBSCAN

The authors of the ST-DBSCAN algorithm [8] present a method which is able to discover clusters in dataset with non-spatial, spatial and temporal data types. Their algorithm is an extension of the DBSCAN and requires additional parameter $Eps2$ as input for the ε-neighbourhood by the second dimension. The algorithm also requires a parameter $\Delta\epsilon$ which is used for evaluation of the non-spatial value whether it has to be appended to the cluster or not. The distance function for the ε parameter can be Euclidian, Manhattan or Minkowski Distance Metric.

The ST-DBSCAN starts from an arbitrary point p from the database and retrieves all density-reachable points from p with respect to $Eps1$ and $Eps2$. If the amount of points is bigger than $MinPts$, a cluster is formed and the algorithm retrieves neighbours of other points within the cluster. For those points the algorithm checks: i) the object is not an outlier, ii) the object is not in the cluster, iii) the value of object is less than the threshold $\Delta\epsilon$, and after positive result appends the point to the cluster. The threshold $\Delta\epsilon$ has to be bigger than the absolute difference between the average (or mean) value of a cluster and a new object's value.

The algorithm also solves the main issue of the DBSCAN algorithm - identifying noise objects when clusters of different densities exist by computing a so called density factor for

each cluster. This density factor captures the scope of the density for the cluster. For its computing, the maximum and the minimum distance in the cluster is evaluated according to Formula 3.1. The density factor then can be found by applying Formula 3.2

$$\begin{aligned} \Theta_{max}(p) &= max\{dist(p,q)|q \in D \wedge dist(p,q) \leq Eps\} \\ \Theta_{min}(p) &= min\{dist(p,q)|q \in D \wedge dist(p,q) \leq Eps\}, \end{aligned} \tag{3.1}$$

where Θ_{max} and Θ_{min} are the minimum and maximum density distance, p and q are points and D is a database.

$$\begin{aligned} \vartheta(C) &= 1/\frac{\sum_{p \in C} \Theta(p)}{|C|}, \\ \Theta(p) &= \frac{\Theta_{max}(p)}{\Theta_{min}(p)} \end{aligned} \tag{3.2}$$

where $\vartheta(C)$ is a density-factor, $\Theta(p)$ is a density distance and C is a cluster.

3.1.2 TRACLUS: TRAjectory CLUStering

The TRACLUS algorithm [16] performs clustering of trajectories' line segments identifying common sub-trajectories illustrated in Figure 3.2. The algorithm is based on the DBSCAN with the same set of input parameters.

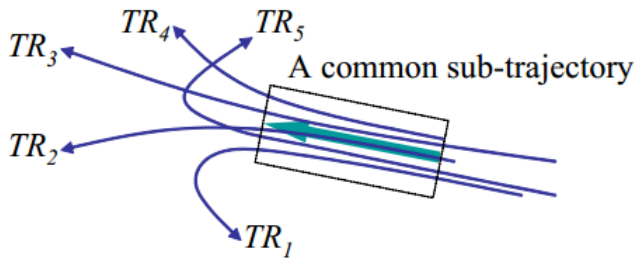


Figure 3.2: Example of a common sub-trajectory [16]

The authors of the algorithm suggest to use a distance function based on the weighted sum of three types of distances: the perpendicular distance between line segments from different trajectories, the parallel distance between line segments of the same trajectory (to check the correct adjacent) and the angular distance measures the directional difference between line segments.

The TRACLUS reciprocally redefines the main definitions of the DBSCAN algorithm in favour of line segments. In Figure 3.3 for a minimum number of line segments ($MinLns$) of 3, thick line segments indicate core line segments. ε-neighbourhoods are represented by ellipses. Segments L_2 and L_3 are **directly** density-reachable from L_1, segments L_1, L_4 and L_5 are density-connected and L_6 is density-reachable from L_3 (but not vice versa)

The authors suggest their own heuristic for the input parameter evaluation. The evaluation of the parameter ε is adopted from the entropy theory. There are two assumptions: i) ε is too small and the amount of line segments in the ε-neighbourhood $\left|N_\varepsilon(L)\right|$ equals 1 for almost all line segments, ii) ε is too large and the $\left|N_\varepsilon(L)\right|$ equals to num_{ln}, where num_{ln} is the total number of line segments. Both assumptions are valid for the worst case of clustering where the entropy is maximal. In a good case, the $\left|N_\varepsilon(L)\right|$ is deviating and

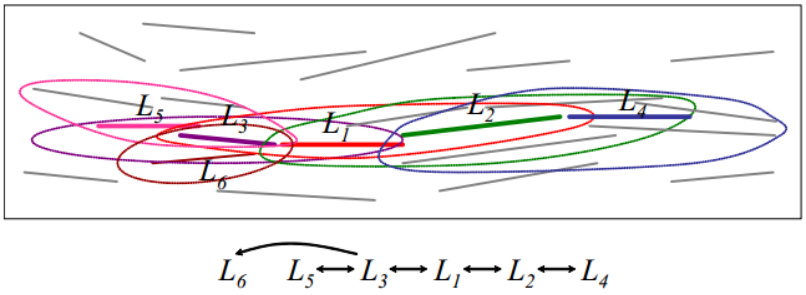


Figure 3.3: Density-reachability and density-connectivity [16]

the entropy becomes smaller. The authors suggest to compute an optimal value of ε which minimizes the entropy $H(X)$ using Formula 3.3.

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \\ \text{where } p(x_i) &= \frac{\left|N_\varepsilon(x_i)\right|}{\sum_{j=1}^{n} \left|N_\varepsilon(x_j)\right|} \text{ and } n = num_{ln} \end{aligned} \tag{3.3}$$

The heuristic for the evaluation of the parameter $MinLns$ is computed from the average $avg_{\left|N_\varepsilon(L)\right|}$ of $\left|N_\varepsilon(L)\right|$ at the optimal ε. Then the minimal number of line segments is computed as $MinLns = avg_{\left|N_\varepsilon(L)\right|} + 1 \sim 3$.

Unlike DBSCAN not all discovered density-connected sets can become clusters. The algorithm assumes that all density-connected line segments in a set belong to different trajectories. The method checks on the trajectory cardinality which cannot be less than a certain threshold.

The TRACLUS algorithm is suitable for mining tasks where we need to discover regions of line segments with common behaviour in a group of trajectories. The algorithm also has the very useful feature in computing of a representative trajectory of discovered cluster which can be used for example in maps mapping.

3.1.3 T-OPTICS: Trajectory Ordering Points

The algorithm T-OPTICS (Trajectory Ordering Points To Identify the Clustering Structure) [20] adopts the OPTICS algorithm's concepts [6] and is a spatio-temporal clustering algorithm. The algorithm uses a distance function depending on the Euclidean distance in the temporal interval (Formula 3.4)

$$D(\tau_1, \tau_2) \mid_T = \frac{\int_T d(\tau_1(t), \tau_2(t))dt}{|T|}, \tag{3.4}$$

where $d()$ is the Euclidean distance over $\mathbb{R}^2$, T is the temporal interval over which the trajectories τ_1 and τ_2 exist and $\tau_i(t)$ $(i \in \{1, 2\})$ is the position of the object τ_i at time t.

The T-OPTICS algorithm performs trajectories clustering for the specific time interval and takes the density threshold parameter ε, and the time interval value as input.

As a quality measure the authors suggest to use the reachability plot (represented in Figure 3.4) which is returned by OPTICS. The graph shows the intrinsic structure of the dataset and distances between the objects. The ε value chosen in the range of the three

central protrusions would discover all 4 clusters (displayed as valleys on the graph). The clusterization can be considered good when there are high density clusters which are clearly separated (the noise is minimal).

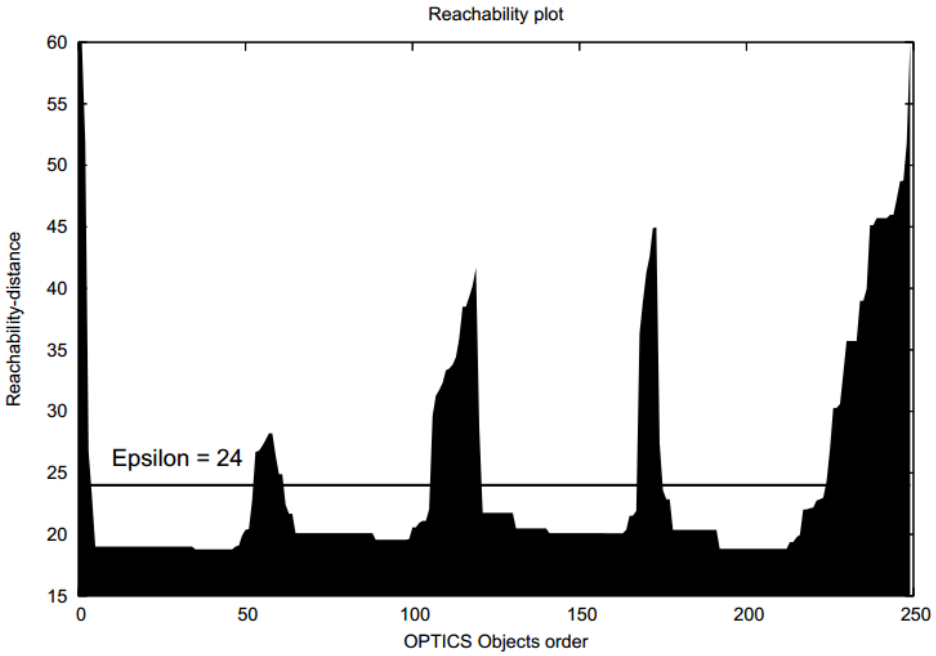


Figure 3.4: OPTICS reachability plot [20]

3.2 Semantic enrichment of trajectories

The process of applying application context to trajectory data is called semantic enrichment [22]. Such trajectories with context are called semantic trajectories and are used for supporting knowledge discovery.

A basic trajectory semantic enrichment process takes as input the set of GPS trajectories and a contextual data repository and as output produces a set of trajectories integrated with some application context. The application context may be a set of geographical places as well as semantic definitions of a certain trajectories' behaviour like 'stop', 'move', 'acceleration', etc.

This chapter describes approaches which are used for semantic enrichment of trajectories. The overview of both, the semantic as a behaviour definition and the semantic as a geographical context enrichment is provided.

3.2.1 Algorithm SMoT: Stops and Moves of Trajectories

The algorithm SMoT suggested by Alvares and others [3] is based on the idea of the semantic trajectories representation as a set of stops and moves. Stops represent a place of interest where the object has stayed for a predefined minimal amount of time.

This algorithm requires a set of GPS trajectories and a set of candidate stops with a minimum time threshold as input. The candidate stop is a topologically closed polygon R_C

defined in R^2 with a minimum time duration Δ_C. R_C has a relevant spatial feature type (e.g parking spot, hotel, etc.) which can be obtained for example from a shape file. The concept of candidate stops is represented in Figure 3.5.

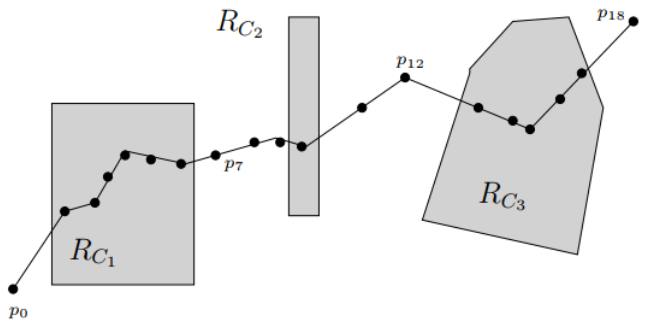


Figure 3.5: Example of a set with three candidate stops [3]

For each point of a trajectory the algorithm verifies, if it intersects a geometry of a candidate stop R_C. If the check is positive, a followed up verification on satisfying the predefined time threshold is performed. If both, geometry and time conditions are met, the candidate stop is recorded as a place of interest (stop). A period between the last timestamp of a previous stop and the first timestamp of the next stop is recorded as a move.

In Figure 3.5 are three candidate stops $R_{C_1}, R_{C_2}, R_{C_3}$. In the beginning the trajectory of an object does not intersect any candidate stop and therefore is considered as a move. In the end two stops (R_{C_1}, R_{C_3}) will be recorded as they satisfy the time duration threshold and three moves (in the beginning till the intersection with geometry R_{C_1}, between R_{C_1} and R_{C_3}, and in the end after the object leaves the R_{C_3} geometry).

The example output of the SMoT algorithm is a semantic trajectory dataset represented in table 3.1. In table (a) for stops dataset the *Tid* attribute corresponds to a trajectory identifier, *Sid* corresponds to a stop identifier, *SFTid* is a geometry identifier, *SFTname* represents a geographic information (parking, airport, etc.) where the stop occurs, *Sbegint* and *Sendt* are the beginning and the end timestamps of the stop respectively.

Table 3.1 (b) for moves dataset has following attributes: *Mid* is a move identifier, *S1id* and *S2id* are previous and next stop identifiers, *geometry* and *timest* corresponds to geometry coordinates and the time during the move.

The algorithm is designed in a way that it can be applicable to any domain - basically the user can define candidate stop set as input from any area he is interested in (traffic management, touristic application, etc.). The decomposition of GPS trajectories into stops and moves helps to reduce the computational complexity during the knowledge discovery as the preprocessing step is performed just once and afterwards the user can query the semantic trajectories in more effective way. The model of stops and moves allows to perform traditional data mining tasks as frequent pattern mining and mining association rules.

The main disadvantage of this algorithm is that the user has to define the candidate places of interest and some potential candidate places can be missed as they are not known by the user in advance.

Table 3.1: Example dataset of SMoT algorithm [3]

(a) Stops

Tid	Sid	SFTid	SFTname	Sbegint	Sendt
1	1	1	Hotel	08:25	08:40
1	2	1	TouristicPlace	09:05	09:30
1	3	3	TouristicPlace	10:01	14:20
...	...	...	...	...	...

(b) Moves

Tid	Mid	S1id	S2id	geometry	timest
1	1	1	2	48.888880 2.246102	08:41
1	1	1	2	48.885732 2.255031	08:42
...	...	...	...	...	...
1	1	1	2	48.860021 2.336105	09:04
1	2	2	3	48.860515 2.349018	09:41
...	...	...	...	...	...

3.2.2 Algorithm CB-SMoT: Clustering-Based SMoT

The CB-SMoT algorithm [21] is based on the SMoT model of stops and moves, but uses a clustering algorithm for discovering candidate stops. The clustering algorithm on which this method is based is a density-based DBSCAN with some changes.

The DBSCAN is changed in the CB-SMoT in order to find clusters in a single trajectory and take the time dimension into consideration. The authors of the CB-SMoT algorithm defined the ε-neighbourhood as the ε-linear-neighbourhood where instead of a radius around a point p, a set of points before and after p is considered, whose distance from p is less or equal to ε. In other words, ε represents the maximum distance between a point p and its neighbours on the trajectory. Another change in the DBSCAN is the definition of the density for a region. Instead of the minimum number of points *minPts* the authors suggest to use the minimal time duration *MinTime*. As the neighbourhood is linear and ordered in time, it is possible to compute a time between the very first and last points of a cluster and define a core point according to Formula 3.5. Therefore the ratio $\varepsilon/MinTime$ gives the maximum average speed of the respective neighbourhood.

$$|t_n - t_m| \geq MinTime \tag{3.5}$$

The ε value is chosen differently in the CB-SMoT as well. In the DBSCAN the user has to set up the value of the neighbourhood, but for the CB-SMoT this approach cannot be used due to the different characteristics of single trajectories. Thence in the CB-SMoT it is suggested to use an adjustable ε value based on a relative parameter related to the mean and the standard deviation. An arithmetic mean μ and a standard deviation σ is computed from a list of distances d_i between two consecutive points p_i and p_{i+1} of the trajectory T. Knowing properties of the trajectory, the quantile function can be used: $[0, 1] \rightarrow \Re$ defined as in 3.6

$$F^{-1}(p, \mu, \sigma) = \mu + \sigma\sqrt{2}erf^{-1}(2p-1),$$
$$\text{and } erf^{-1}(x) = \sum_{k=0}^{\infty} \frac{c_k}{2k+1}(\frac{\sqrt{\pi}}{2}x)^{2k+1}, \tag{3.6}$$

where $c_0 = 1$ and $c_k = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)}$

The user needs to know the approximate proportion of points that generate potential stops in the relation to the total number of points in the trajectory. Therefore the parameter called *area* $\in$ [0, 1] is required for the computation of the ε value.

3.2.3 Reverse geocoding method

Geocoding is the process of obtaining coordinates (lat, lon) from the address for locating of the marker on a map. The reverse geocoding method, on the contrary, uses (lat, lon) coordinates to obtain the location address[1]. The reverse geocoding approach uses online GIS (Geographic Information System) databases like Google Maps or OpenStreetMap [2] to retrieve the data.

In [10], [18] reverse geocoding is used for mining semantic locations from raw GPS data. First, the nearest to the query position addressable location with the coordinates of the street address is returned. Secondly, the yellow and white pages directory is used for obtaining a list of semantic locations which are near the query street address. This traditional reverse geocoding method may produce incorrect results. As shown in Figure 3.6 two buildings B_1 and B_2 have entrances E_1 and E_2. The reverse geocoding will return the 'W Taylor St.' instead of the actual address 'S. Halsted St.' for the query point E_2 as it is closer to the 'W Taylor St.'. Therefore it is important to consider a returned set of street addresses as candidate streets and define some distance tolerance threshold from the query position to the physical location of each street in a set.

Querying the street addresses available on the internet yellow pages directory will deliver additional information like correspondent business names and business types. The final information about physical location can be represented as a triple containing location name, semantic category and street address.

Figure 3.6: Example of reverse geocoding [18]

[1] https://developers.google.com/maps/documentation/geocoding/intro
[2] https://www.openstreetmap.org/

Chapter 4

Design of the trajectory data preprocessing framework

In this chapter the proposed trajectory data preprocessing framework is presented. The main application problem which has to be solved is discovering semantic locations from the street traffic dataset. The dataset consists of GPS trajectories of vehicles and therefore the trajectories have to be annotated with geographic context for future mining tasks.

First, each trajectory is segmented into stops as stops indicate places of interest. Secondly, a density-based clustering algorithm is applied to group stops in order to discover interesting places for a group of objects. Thirdly, the reverse-geocoding technique by bounding box is utilized for discovering semantic locations. The design of the framework is inspired by ideas of [10].

4.1 Trajectories segmentation

For choosing the proper approach for the trajectories segmentation, the main requirements for discovering interesting places have to be defined. Our framework will preprocess data from the GPS sensors of vehicles and the primary interest for us represent places where several vehicles did a stop as such physical places may have a certain context (shopping mall, parking spot, etc.). Therefore the segmentation of trajectories has to be performed in order to obtain information about vehicles' stop points.

The best approach is to use the Stops and Moves model. The GPS sensor of a vehicle is collecting data during the whole period of driving and stops recording when the vehicle is turned off. This feature can be used for identifying stops in a vehicle's trajectory. The stop point will be identified by computing the time duration Δt between two consecutive points p_i and p_{i+1}. The point p_i will be marked as a stop if the time duration Δt is bigger than the required minimum duration threshold. The stop P_{stop} is presented as $P_{stop} = (p_i, T_{arr}, \Delta t)$, where p_i is a location point of the stop, T_{arr} is the time of arrival to the stop and Δt is the stop duration. An example of the output dataset after stops extraction is represented in Table 4.1, where Tid and Sid are the trajectory ID and the stop ID respectively.

Table 4.1: Example of a stops dataset

Tid	Sid	P_{stop}	T_{arr}	Δt
1	1	49.195416,16.609122	08:25	30
1	2	49.196583,16.60296	14:15	45
1	3	49.193884,16.609424	18:00	75
...	...	...	...	...

4.2 Discovering places of interest

The main interest of this work is to discover semantic locations which are places of interest for groups of vehicles. Therefore the density-based clustering algorithm can be applied to extracted stops in order to discover regions of interest.

The use of traditional density-based clustering algorithms like the DBSCAN or the OPTICS considers just the spatial constituent which can lead to inaccurate results. Let us consider an example of the average gas station which serves 200 cars per day, but it cannot serve more than 10 cars at approximately the same short interval of time. The application of the clustering algorithm on the spatial dimension only would discover such gas station as a significant place which is not correct. The other situation is vehicles parked at the same area for a long period of time (e.g in residential areas over night). In order to eliminate the discovery of such places the time threshold requirements have to be set for the clustering method.

As a basis for the clustering algorithm ST-DBSCAN (chapter 3.1.1) is chosen, which allows to set requirements on the distance function for the non-spatial values and is able to discover clusters of various densities. The density-based clustering algorithm of the proposed preprocessing framework takes on input the following set of parameters:

$< D, Eps1, Eps2, MinPts, \Delta\epsilon, \Delta t_{max} >$, where D is a stops dataset filtered by t_{min} and t_{max} thresholds. Then the algorithm consequently retrieves neighbours for each point with respect of the input parameters values.

4.2.1 Distance function

The choice of an appropriate distance function depends on the clustered objects' features. As we have stop points distributed in space and time, the Euclidean distance can be applied for both, spatial and temporal values.

Let us consider two stop points $P_{stop1} = (x_1,\ y_1,\ T_{arr1},\ \Delta t_1)$ and $P_{stop2} = (x_2,\ y_2,\ T_{arr2},\ \Delta t_2)$. The Euclidean distance for the parameters $Eps1$ and $Eps2$ can then be found according to 4.1.

$$\begin{aligned} Eps1(Pstop1, Pstop2) &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ Eps2(Pstop1, Pstop2) &= \sqrt{(T_{arr1} - T_{arr2})^2} \end{aligned} \tag{4.1}$$

4.2.2 Computing the cluster centroid

Computing of the cluster's centroid can be a part of the clustering algorithm like in $k-means$. Density-based algorithms usually do not require computing of clusters' centroids. Knowing of the centroids' values of discovered clusters can be useful for application mining algorithms in the future.

As we deal with geospatial data points, the method for computing the cluster centroid has to respect their features. Therefore, computing of the geographic midpoint method [1] has to be used in this work to find clusters' centroids.

First, each point $p_i = (lat, lon)$ of the cluster will be converted into a Cartesian coordinates vector $p_i = (x_i, y_i, z_i)$. Secondly, these vectors will be summarized and the resulting vector will be normalized. Finally, the result will be converted back to spherical coordinates.

4.2.3 Parameters evaluation heuristic

The authors of the ST-DBSCAN suggest their own heuristic. In the beginning, the value $MinPts$ needs to be computed as $\ln(n)$, where n is the size of the database. From the value $MinPts$ the Eps parameters will be found. The $MinPts$ parameter is used to obtain $MinPts$-nearest neighbours for each object. Then the $MinPts$-distance values will be sorted in descending order to get the sorted graph. The Eps parameter value should be less than the distance defined by the first valley on the graph.

In this thesis we will use the reachability plot heuristic suggested by the OPTICS algorithm to graphically evaluate the quality of clustering. Note that for the ST-DBSCAN algorithm two reachability graphs will be used, one is for spatial and another one is for temporal dimensions.

4.3 Semantic enrichment of trajectories

The semantic enrichment module requires the set of discovered clusters as input. Each cluster has an arbitrary shape and contains several stops.

The reverse-geocoding method by bounding-box will be used in this work. First, the query containing bounding box coordinates of the cluster is sent to the OpenStreetMap online GIS database for retrieving semantic locations. Consequently, the user can select semantic locations found in the cluster's area which are interesting for the analysis. Finally, trajectories will be integrated with semantics.

4.3.1 Semantic locations discovery

Unlike the authors of proposed reverse-geocoding method in [10], each point from the cluster will not be queried to obtain the address as it generates a lot of requests and therefore is computationally not effective. The online yellow pages source also will not be used in this work as it may return inaccurate results and might be unavailable for some countries.

Instead, the bounding box of each cluster will be retrieved from the stops with minimal values (lat_{min}, lon_{min}) and maximum values (lat_{max}, lon_{max}). Then, using the Overpass API [2] service, all location objects which are inside or intersect a bounding box will be extracted as shown in Figure 4.1.

[1] http://www.geomidpoint.com/calculation.html
[2] https://wiki.openstreetmap.org/wiki/Overpass_API

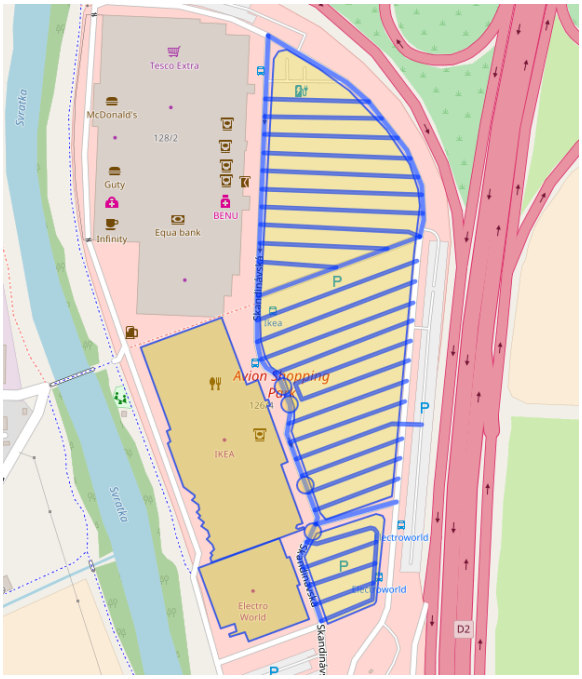


Figure 4.1: Example of semantic locations' geometry extracted by bounding box

4.3.2 Annotating trajectories with semantics

Extracted data from the Overpass API contains location points within the bounding box. The service will return just those location points which contain semantics. The user can filter locations which are valuable for the analysis. For example, places marked as a 'bus stop' can be excluded, but places with the category 'supermarket' can be interesting for future study. Note that within one cluster several semantic locations can be. An example of output data with integrated semantics is represented in Table 4.2 where Sid, Cluster_ID, Cluster_centroid and Sem_pl_ID are a stop, cluster identifier with cluster centroid coordinates, semantic place identifier and Loc_coord is a location coordinate of the semantic location.

Table 4.2: Example of a dataset with semantics

Sid	Cluster_ID	Cluster_centroid	Sem_pl_ID	Loc_coord
20	10	49.1531256,16.6283180	1	49.1544447,16.6292356
21	10	49.1531256,16.6283180	2	49.1582668,16.6279795
15	9	49.1534894,16.6281212	14	49.1534894,16.6281212
...	...	...	...	...

Chapter 5

Implementation

This chapter describes the implementation of the proposed data preprocessing framework for discovering semantic locations. For the implementation of the back-end part the programming language Java 8 was chosen. This version of Java provides a big advantage in having the ability to use functional features such as lambdas, which allows to build highly performant data science projects. Implementation of front-end has been done with AngularJS thanks to its high performance in data binding and built-in wrappers for communication with RESTful services.

First, a quick overview of the application's GUI will be provided. Secondly, the overall architecture of the application will be described. Thirdly, the design solution for the implementation of clusters' parameters heuristics, the OPTICS algorithm, will be marked out. Lastly, the design of the ST-DBSCAN algorithm and the semantic locations discovery module will be overviewed.

5.1 GUI overview

The *Sminer* application is designed as a single page web application divided into four tabs: Extract stops, OPTICS reachability plot, Extract semantic locations and Configuration.

Input file format description

The application takes as input a `.csv` format file. For successful data parsing the following set of columns is required: moving object ID, timestamp, latitude and longitude coordinates. Supported data formats of columns are listed in table 5.1. The *Sminer* also assumes that the file contains a header and skips the first row.

Table 5.1: Supported formats of data in the input file

ModId	Time	Latitude	Longitude
Integer	Date: yyyy-MM-dd hh:mm:ss	Double: wgs84 coords	Double: wgs84 coords

Configuration tab

To provide better user experience, the application provides the ability to configure the order of the columns of the input file which will be parsed into the internal data structure `Record`. Therefore the user does not need to edit the code to parse the file with various columns ordering.

Extract stops tab

This is the start screen of the application represented in Fig. 5.1 and necessary step to continue the data preprocessing. The upload file functionality is also located on this tab and after reading the file, some file statistics are shown to the user. These statistics help to identify the total number of records in the file as well as the number of valid records during parsing of the dataset into the internal data structure. After reading the file, there is a possibility to extract stops from the trajectories for which it is enough to set a minimum stop duration in minutes. However, it is better to set the maximum stop duration as well, as this additional limitation can improve the future analysis in terms of efficiency and can help in more detailed data analysis. The user also has the possibility to filter the dataset by particular days.

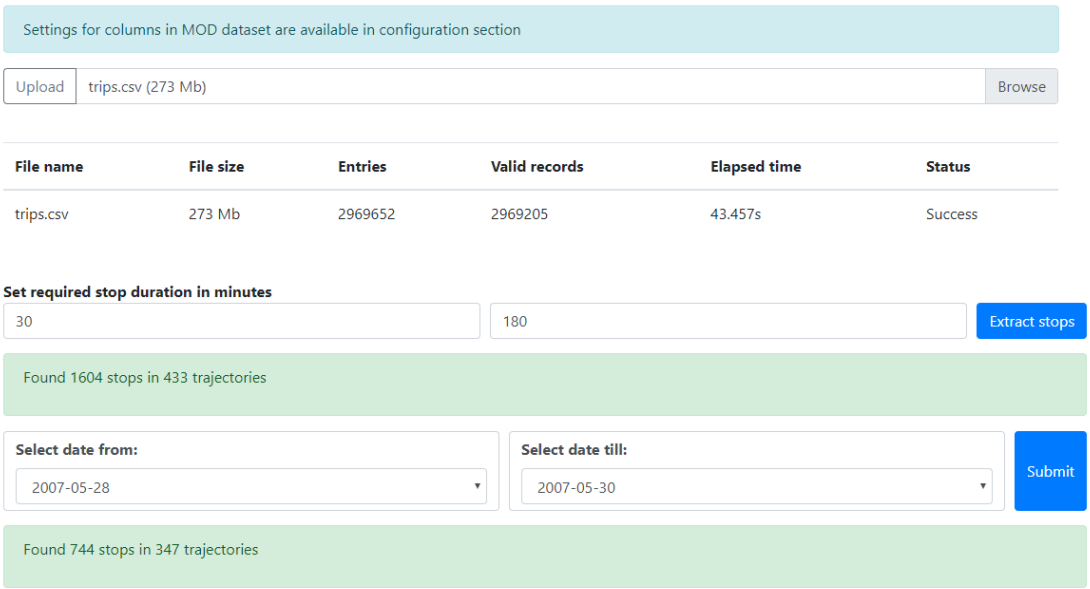


Figure 5.1: Extract stops view with file statistics and filtering stops functionality

OPTICS reachability plot tab

The OPTICS reachability plot tab provides the possibility to evaluate the dataset distribution and choose appropriate parameters for clustering. There is an implemented functionality to display the dataset distribution by temporal, spatial and both, temporal and spatial dimensions. Further description of the module will be provided in chapter 6.

Extract semantic locations tab

The last tab allows to perform clustering depending on the input parameters of temporal and spatial $\varepsilon - neighbourhood$ and the minimal amount of points to shape a cluster $MinPts$. The user can view discovered clusters with stop points and discover locations for clusters (Fig. 5.3a), display clusters' area and found locations' geometry on a map (Fig. 5.3b). There is also the option to edit locations if needed (Fig. 5.2). In the end the user can export preprocessed data to a JSON file, which contains all information about discovered clusters, locations and locations' geometry.

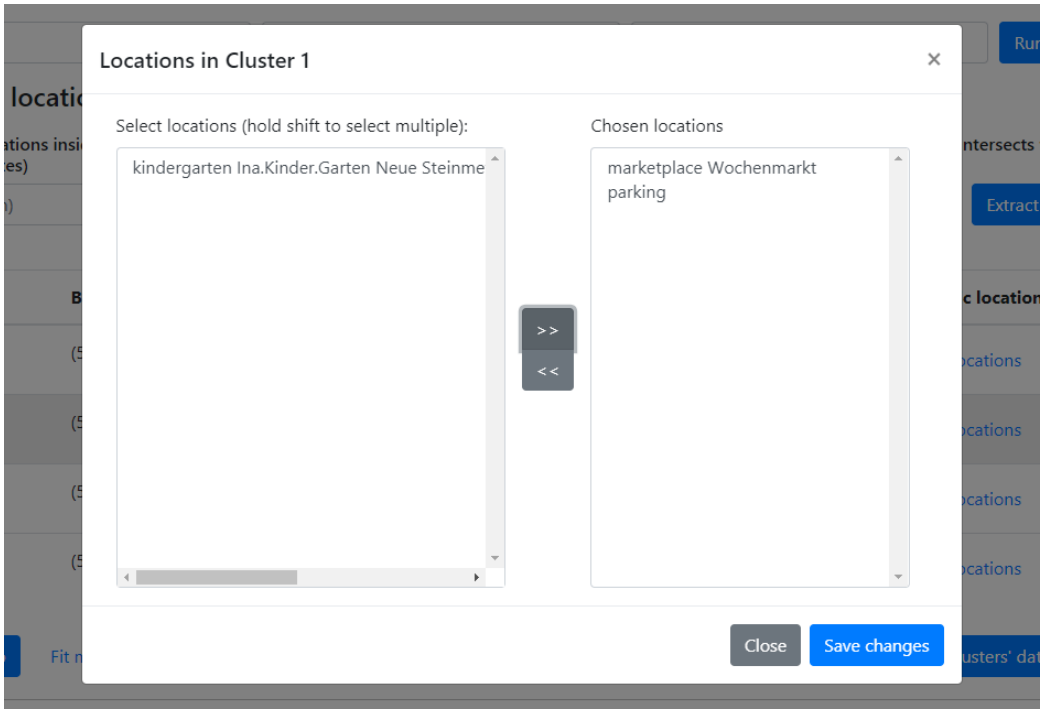
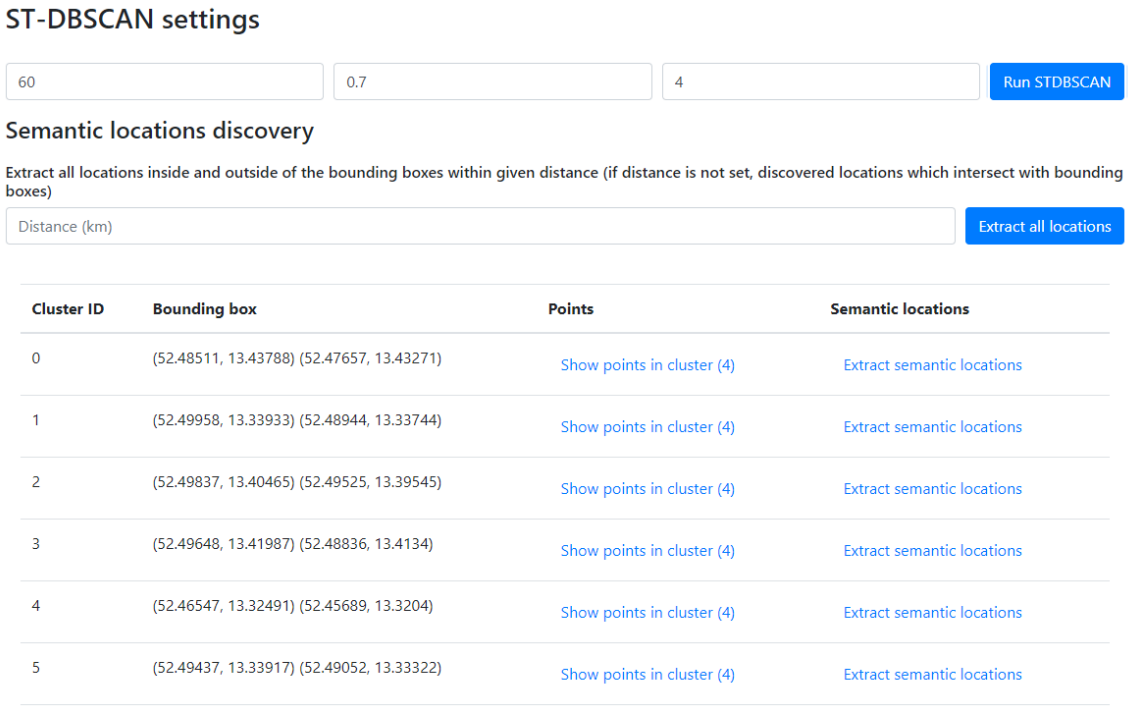
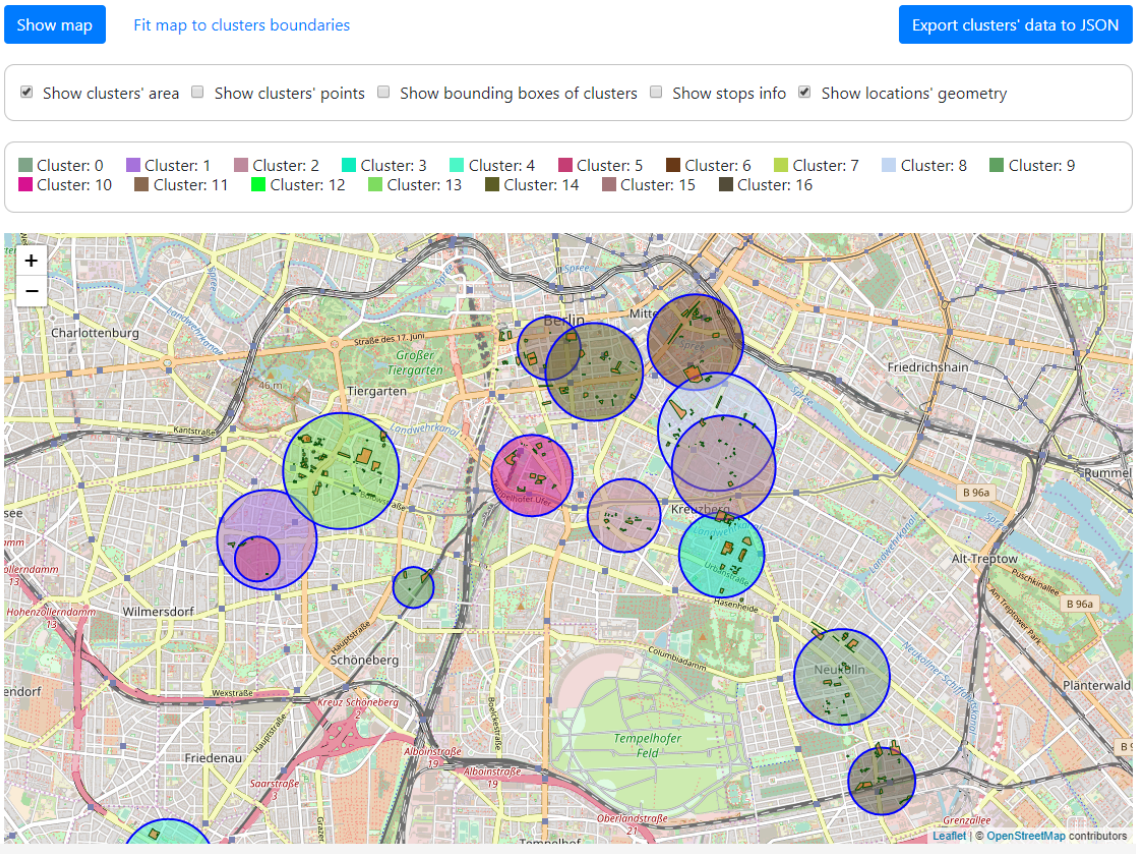


Figure 5.2: Modal window to edit discovered locations



(a) View with discovered clusters in a list



(b) View with clusters' area and locations on a map

Figure 5.3: GUI of the ST-DBSCAN tab with discovered clusters

5.2 Architecture

The framework is designed as a REST (Representational State Transfer) application and runs on an application server. The applications' architecture is built according to the MVCS (Model - View - Controller - Service) model and is divided into the following modules: input dataset configuration and parsing, data analysis operations provider, OPTICS, clustering and semantic locations extraction. In Figure 5.4 the REST architecture with all endpoints is illustrated. The diagram contains just names of the public methods of interfaces, further description of particular services will be provided below.

Requests from the client are handled in `RestApiController` and are delegated to specific modules. All operations related to the input dataset preprocessing in order to convert it into the main data structure `Record` are handled by the `IDocumentService` interface. For the dataset analysis the `IDataAnalysisService` interface is used as a provider. Utilization of the provider has two main functions: delegating of the request to particular algorithm implementation and additional data preprocessing from the response in order to provide appropriate data structure to the consumer layer.

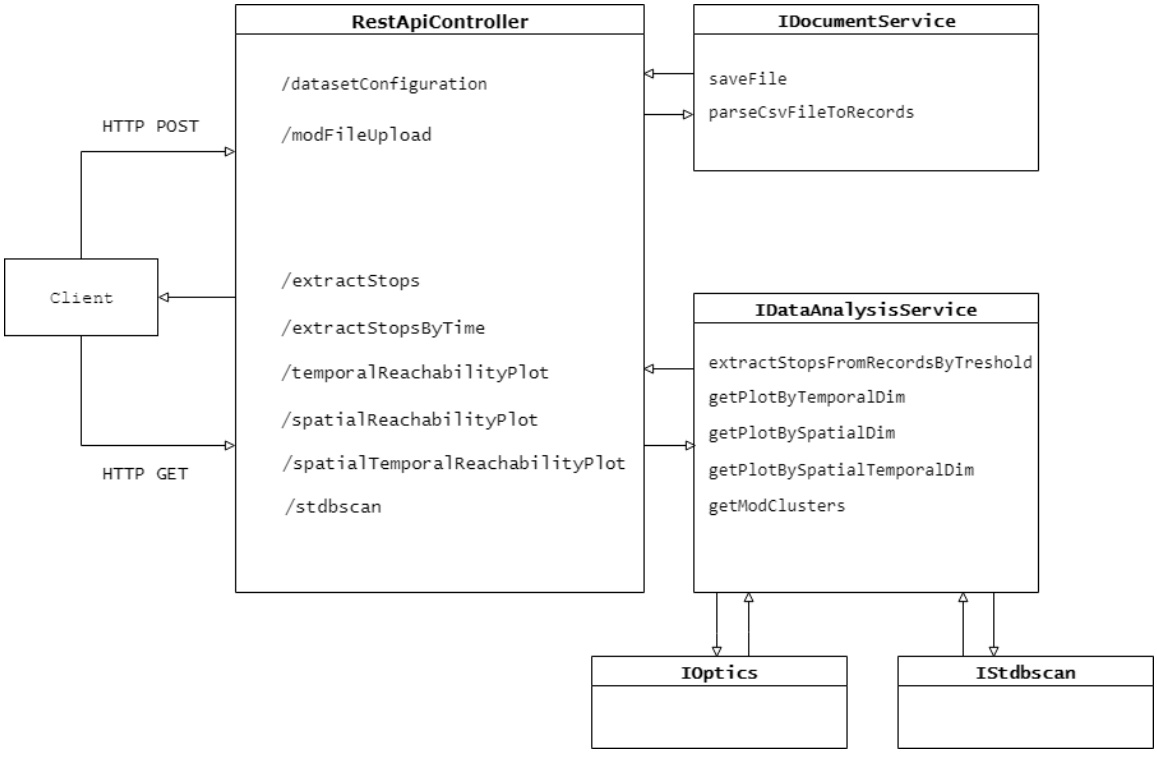


Figure 5.4: REST architecture diagram

5.3 Implementation of the OPTICS algorithm

The OPTICS algorithm is used in this work as a heuristics for parameters evaluation for future clustering work. First, the OPTICS algorithm running on a single dimension (temporal or spatial) was implemented. However, after the experimental evaluation of the implemented method, it became obvious that this implementation of OPTICS is not able to provide the full picture of the dataset distribution to satisfy the framework's requirements. Since the requirement is to perform clustering in both dimensions, spatial and temporal, a two-dimensional OPTICS was implemented in this work.

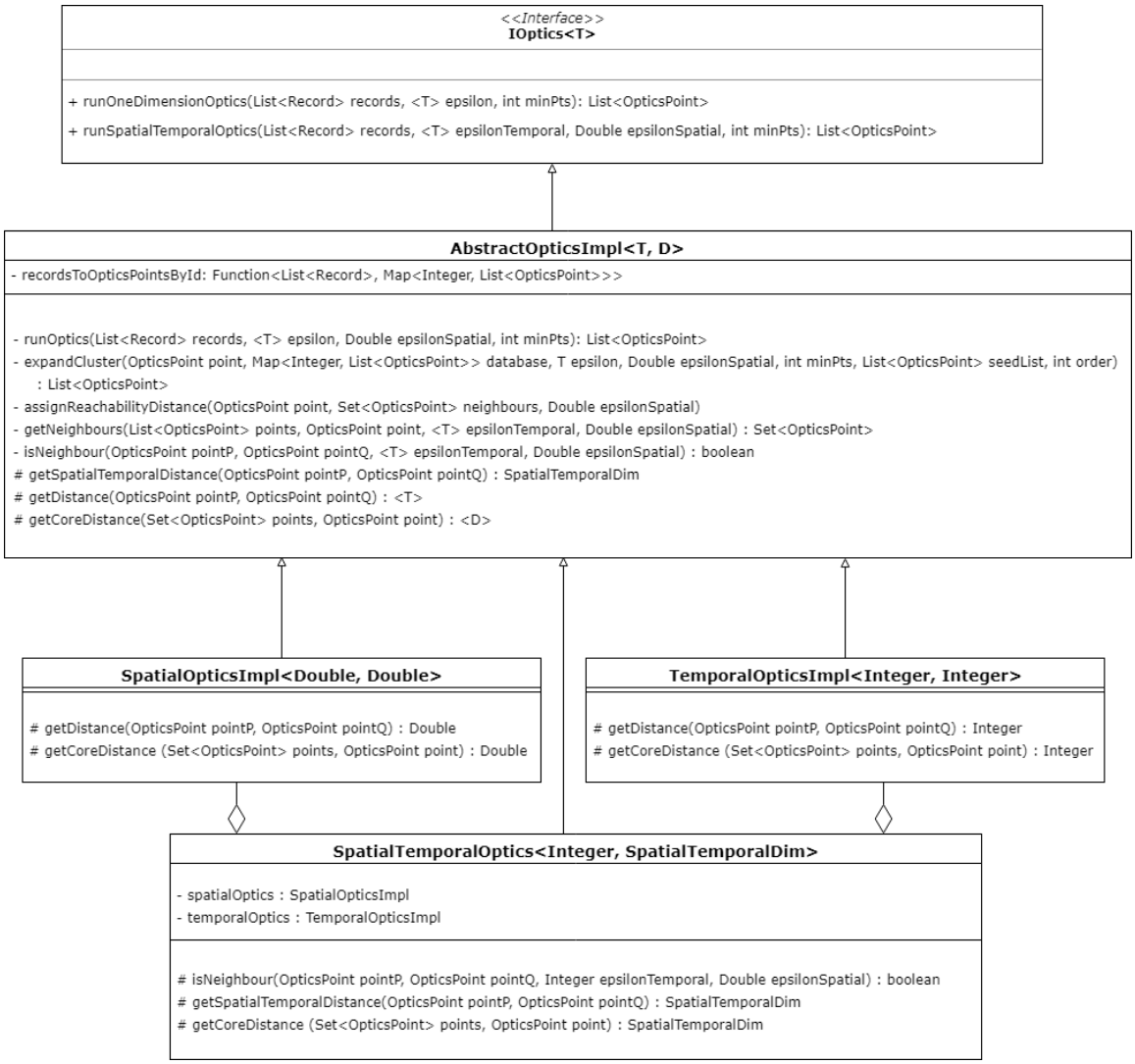


Figure 5.5: OPTICS implementation class diagram

In Figure 5.5 the OPTICS class diagram is illustrated. The generic interface `IOptics<T>` provides access to two abstract methods `runOneDimensionOptics`, which calls an implementation of the one-dimensional OPTICS, and `ruSpatialTemporalOptics`, which provides data analysis in two dimensions. The method `runOneDimensionOptics` contains a generic parameter `<T>`. For the temporal dimension this parameter is substituted by the

`Integer` type, for the spatial it is substituted by `Double`. The usage of a generic type allows to distinguish higher level of abstraction and better code reuse.

Class `AbstractOpticsImpl<T, D>` implements methods of the `IOptics<T>` interface. The usage of the abstract class is beneficial as each child class can override particular parent's methods implementation depending on the type of the OPTICS dimension.

5.3.1 Data structure

OPTICS takes on input a basic data structure `Record`, which stores the GPS records representation from the `.csv` file. The functional interface `recordsToOpticsPointsById` converts the `Record` entity to the `OpticsPoint` entity and groups elements by moving object IDs to form trajectories. Since the `OpticsPoint` data structure is similar to `Record`, `OpticsPoint` inherits from the `Record` data structure and extends its properties. To operate with data in two dimensional OPTICS, the `SpatialTemporalDim` data structure was created, which allows to store the result of the distance function `getCoreDistance` and perform a comparison in order to sort discovered points in a seed list while processing the dataset.

5.3.2 Processing the dataset

The method `runOptics` is an entry point for both methods, `runOneDimensionOptics` and `ruSpatialTemporalOptics`. It iterates through trajectories' stops and calls the method `expandCluster`, which discovers neighbours according to set parameters of ε value and minimum number of points($MinPts$). The `expandCluster` method accepts the current data point for which neighbours might be discovered, the database with stops grouped by trajectories, parameters for clustering, current list with already discovered neighbours (if present) and current order of processing the point. It filters out elements from the database, which belong to the same trajectory as the point on input. Afterwards all neighbours are discovered using the lambda expression. If the size of the neighbours set satisfies the parameter $MinPts$ on input, the `assignReachabilityDistance` method chooses an appropriate implementation of the distance function to compute core distance and reachability distance and assigns it to the respective neighbours. The current point is marked as processed and the current order of processing is assigned to it. The `expandCluster` method returns all neighbours which were discovered. In case `expandCluster` has already a list of previously discovered neighbours on its input, it joins two lists together and returns the result. The returned list with points is sorted in ascending order and the algorithm continues processing this list. The cycle stops when all points of the dataset of stops have been processed.

The neighbourhood of the point is discovered by the `getNeighbours` method, which applies the `isNeighbour` method in the lambda expression. The `isNeighbour` method utilizes the distance function to identify the neighbour of the point. For the temporal dimension OPTICS Euclidean as the distance function (as described in 4.2.1) is used. For the spatial OPTICS the Pythagoras' theorem[1] 5.1 is used, as this method is efficient on computing short distances between two points which are represented as geospatial coordinates.

[1] https://www.movable-type.co.uk/scripts/latlong.html

$$\begin{aligned} \Delta x &= (lon_2 - lon_1)\cos(\frac{(lat_1 + lat_2)\frac{\pi}{180}}{2}) \\ \Delta y &= (lat_2 - lat_1) \\ Eps2(Pstop1, Pstop2) &= R\sqrt{\Delta x^2 + \Delta y^2} \end{aligned} \tag{5.1}$$

where R is the Earth's radius $R = 6371e3$.

5.4 Semantic locations discovery

Semantic locations discovery is a follow up step after the evaluation of the dataset structure by the OPTICS algorithm. It consists of two main parts, the first is running the two-dimensional clustering algorithm ST-DBSCAN, the second is querying the OpenStreetMap service through the Overpass API and Nominatim API in order to get clusters' locations and semantic places' names and geometry.

5.4.1 Implementation of the ST-DBSCAN algorithm

Implementation of the ST-DBSCAN algorithm is quite similar to the implementation of OPTICS, but not polymorphic. Figure 5.6 illustrates the ST-DBSCAN class diagram. The interface `IStdbscan` contains one public method `runStdbscan` to invoke clustering. Similar to the OPTICS implementation, the `runStdbscan` accepts the basic data structure `Record` and converts it into the ST-DBSCAN data structure `StdbscanPoint`, which extends the `Record` entity object. Processing the dataset works in a similar way as in OPTICS, but the difference is that it does not sort the set of discovered neighbour points. While processing points in the dataset, the predicate `isValidPoint` is applied to filter points, which have not been assigned to any cluster yet and not marked as noise.

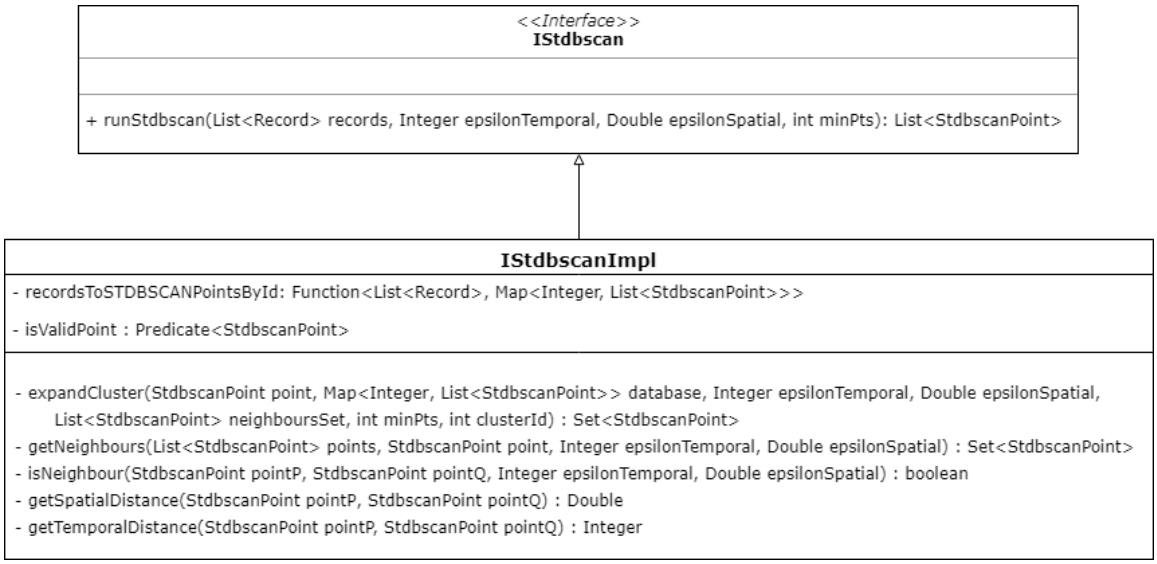


Figure 5.6: ST-DBSCAN implementation class diagram

5.4.2 Implementation of semantic locations discovery

Semantic locations discovery is the last step in data preprocessing and fully implemented on the front-end side. There are two OSM data services provided by OpenStreetMap, which are used in this project: Overpass API[2] and Nominatim[3]. The motivation of using two data services instead of one lays in better reliability of discovering semantic locations.

The method `getSemanticLocations`, which is responsible for semantic locations' discovery, is located in the AngularJS controller `sminerController.js`. It sends requests to the Overpass API interpreter first, parses the response getting such data as node names and addresses (if present) and assigns semantic locations to the respective cluster. If no semantic locations were discovered in this step, the second request to Nominatim service is sent.

Overpass API

After grouping points into clusters, we are able to compute the bounding boxes in which every cluster is enclosed. This information is enough to query the OpenStreetMap service via the Overpass API. The Overpass API allows to query for OSM data by specific search criteria. For this purpose, it has a specifically crafted query language Overpass QL. The basic semantics of the Overpass API is that flows of OSM data (nodes, ways, relations) are generated and modified by statements, which are executed one after another[4]. A big advantage of using this API is that within a single HTTP GET request it is possible to get all data we are interested in. In this project query 5.1 is used to get the geometry and names of places inside the bounding boxes.

```
https://overpass-api.de/api/interpreter?data=[out:json];way[amenity]
(latMin, lonMin, latMax, lonMax);out geom;
```

Listing 5.1: Overpass query

where `(latMin, lonMin, latMax, lonMax)` parameters define a cluster's rectangular bounding box.

This query returns only places with a special tag `amenity`, with which useful and important facilities for visitors and residents[5] are marked. The node name `way` allows to get all ways of the bounding box. A way is found not only if it has a node inside the bounding box, but also if it just crosses the bounding box somewhere. The combination of query parameters `way[amenity](latMin, lonMin, latMax, lonMax)` and `out geom` allows to get the geometry of all important places, which adjoin the way and this way can be either inside or cross the bounding box.

Nominatim

Nominatim is a service for searching OSM data by name and address, but the main feature, which is utilized in this project, is its ability to generate addresses of OSM points by reverse geocoding. The Nominatim service is used when no semantic locations were identified by the Overpass API (no nodes with tag `amenity` were found). The query used in this project is listed in 5.2.

[2] https://wiki.openstreetmap.org/wiki/Overpass_API
[3] https://wiki.openstreetmap.org/wiki/Nominatim
[4] https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide
[5] https://wiki.openstreetmap.org/wiki/Key:amenity

```
https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat=<X>&lon=<Y>
&osm_type=W&zoom=18&polygon_geojson=1&addressdetails=1&accept-language=en
```

Listing 5.2: Nominatim query

Meaning of used parameters is listed below:

- `reverse?` - reverse geocoding mode generates an address from a latitude and longitude
- `jsonv2` - set up response format to JSON
- `lat=<X>&lon=<Y>` - latitude and longitude of point to reverse geocode. Cluster's centroid coordinates are used in this project
- `osm_type=W` - query for node name „way“
- `zoom=18` - level of detail, where 0 is country and 18 is house/building
- `polygon_geojson=1` - output geometry of result in geojson format
- `addressdetails=1` - include a breakdown of the address into elements to response

Chapter 6

Evaluation

This chapter provides an overview of experiments performed on the implemented modules described in Chapter 5. First, the description of experimental datasets will be provided. Secondly, the main goal and workflow of experiments will be described. Thirdly, evaluation of the experimental datasets distribution will be shown using the OPTICS reachability graph. Finally, the ST-DBSCAN results and semantic locations discovery module will be evaluated.

6.1 Datasets

The implementation was tested on two datasets. The first dataset contains of a real data from GPS sensors of taxis within Beijing. The second one is a synthetic dataset of simulated cars driving on the street network of Berlin. More detailed overview of both datasets are listed below.

T-Drive trajectory dataset

T-Drive trajectory dataset[1] contains the GPS trajectories of 10 357 taxis during the period of Feb. 2 to Feb. 8, 2008 within Beijing. The total number of GPS records is 17,5 million. The total distance of the trajectories reaches to 9 million kilometres. The original dataset consists of smaller `.txt` files which for the purpose of experiments were merged into one `.csv` file. The file size of the merged dataset is 824 Mb. There were no further corrections of this dataset.

BerlinMOD

Berlin MOD dataset[2] is a benchmark data created using the Secondo DBMS[3] generator software and is a result of people's behaviour simulation, commuting between homes, work and leisure time places within the street network of Berlin. The experimental dataset contains data for 6 days of simulation, 447 vehicles with 2 969 652 GPS records in total and has a size of 273 Mb. The dataset was used as it is, without any additional corrections.

[1] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/
[2] http://dna.fernuni-hagen.de/secondo/BerlinMOD/BerlinMOD.html
[3] http://dna.fernuni-hagen.de/secondo/

6.2 Results

6.2.1 Goals and workflow

Our goal is to discover semantic locations in small areas with high density of parked vehicles in relatively short interval of time. We want to exclude cases where cars are parked in residential areas (e.g. private cars over night) or near working places for a long period of time and places with high throughput like gas stations. Therefore, short stop duration (from 20 minutes to 3 hours) and small area (up to 400 meters) will be considered in this work. This would lay the foundation for possible data mining, e.g. identification of frequently visited areas in the city or most popular parking areas near significant locations.

The workflow of experiment is as follows:

- The stops extraction from trajectories will be conducted before clustering. This step is supposed to limit the total amount of GPS points to cluster and filter the duration of stops which are interesting for the analysis. In case of a big amount of stop points (more than 10 000), filtering by days will be applied to reduce computational complexity.
- Identification of the clustering structure will be performed with the OPTICS algorithm in order to choose appropriate parameters for the next step
- Clustering of trajectories' stops using the ST-DBSCAN algorithm
- Discovery of semantic locations for each cluster
- Project results on a map

6.2.2 Identifying the clustering structure

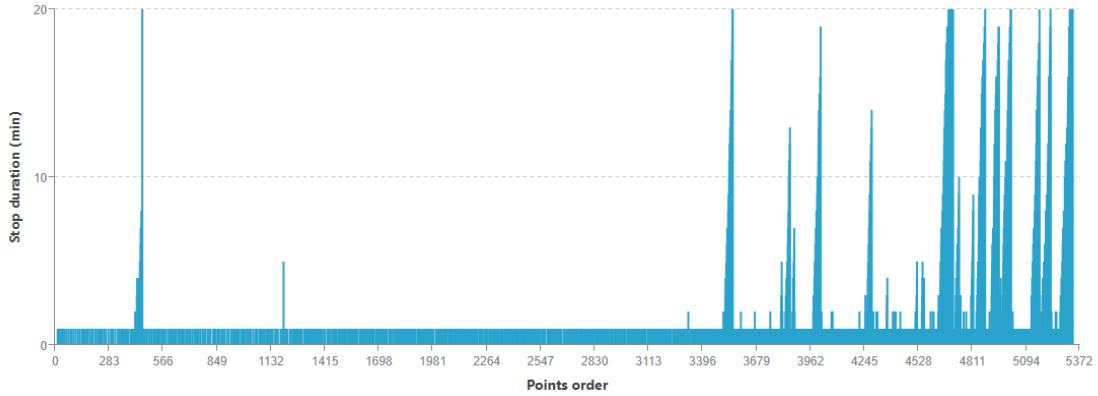
Identification of clustering structure for both datasets was conducted using the OPTICS algorithm. For evaluation of points ordering in the dataset, two combinations were used: i) small generating distance ε between points with bigger $MinPts$ and ii) bigger generating distance ε with smaller $MinPts$. Results for various parameters of $\varepsilon - neighbourhood$ and $MinPts$ for temporal, spatial and spatial temporal dimensions are presented below.

T-Drive trajectory dataset

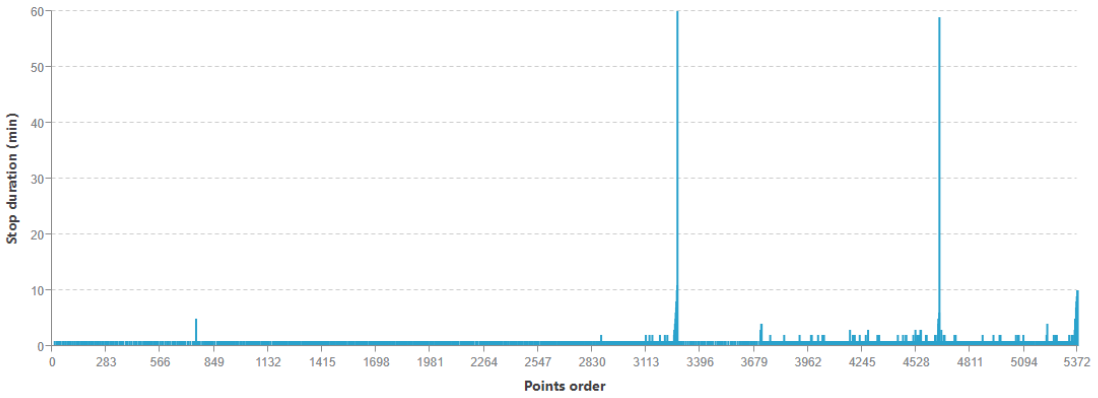
The T-Drive trajectory dataset is a large dataset, which needs a certain data reduction. After empirical evaluation of the data in the dataset, it was noticed that GPS points by some vehicles were reported with 10 - 20 minutes intervals. Therefore, the minimal limit for the stop duration should be higher than those values to filter out incorrect GPS records. For the experiment the data was filtered by stop duration within an $t_{min} = 1$ hour and $t_{max} = 3$ hours interval. The analysis was performed separately for 3 consequent days (Feb. 2 - Feb. 4, 2008). The average amount of stops per each day is around 5 000 reported in average by 3 500 taxis.

The dataset contains a big amount of stop points and therefore we can easily set up bounds for $MinPts$ to 20 - 50 points. First experiment was done with a short temporal distance 20 minutes between stops and a high number of points in a cluster $MinPts = 50$. The reachability plot (Fig. 6.1a) shows several big clusters (≈ 5) and smaller ones. With a bigger time interval equal to 1 hour and a smaller number of points $MinPts = 20$, the stops

were grouped into two large clusters (Fig. 6.1b). Based on these results we can come to a conclusion that for better separated clusters by the temporal dimension we should consider smaller time intervals with a high enough value of $MinPts$.



(a) smaller generating distance $\varepsilon = 20$ minutes, $MinPts = 50$

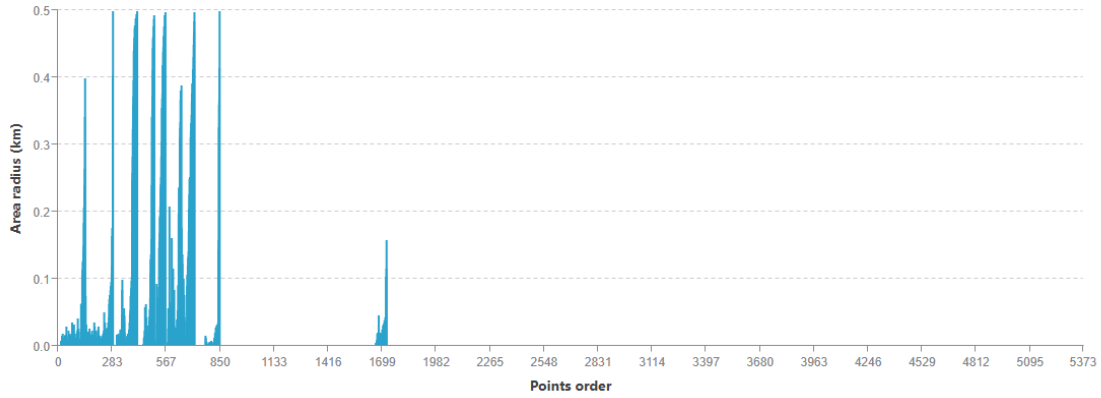


(b) bigger generating distance $\varepsilon = 1$ hour, $MinPts = 20$

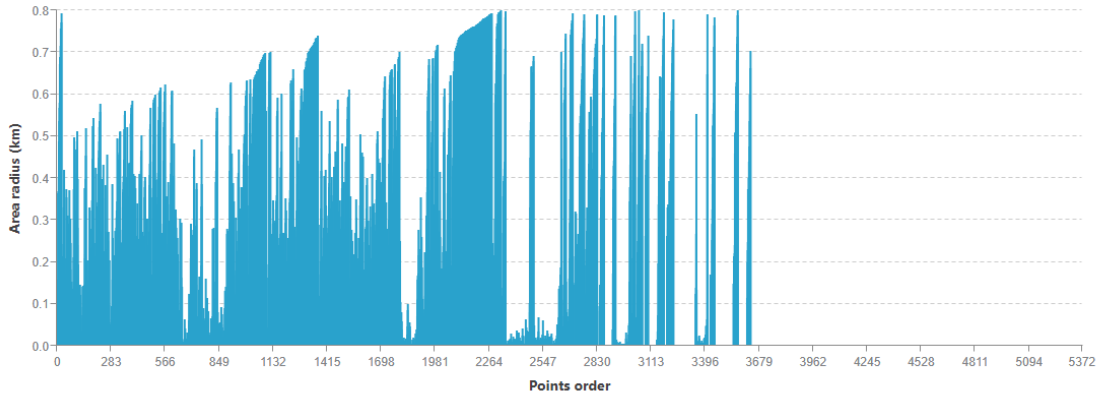
Figure 6.1: Temporal OPTICS algorithm results for the T-drive dataset

The second experiment was performed for the spatial dimension. As initially we are not interested in big areas, it is worth to choose a generating distance in bounds of 0.2 - 1 kilometres. The reachability plot for a smaller generating distance $\varepsilon = 500$ metres (6.2a) shows better separated clusters, while the reachability plot for a bigger generating distance (6.2b) $\varepsilon = 800$ metres does not indicate about a clear separation - there are some large clusters (≈ 6) and a lot of smaller ones. Based on these results we can conclude that the dataset is sensitive to changes of the spatial distance value as it did not change significantly in both experiments, while the number of points was reduced twice their initial value.

The last experiment was done for two dimensions - spatial and temporal. We attempted to combine small generating distances of ε from the previous results, but there was no such clustering, which would satisfy those conditions. Since the small spatial distance plays a more important role, we can increase the ε value for the temporal dimension and decrease the minimal number of points $MinPts$. Results plotted in graph 6.3 show an almost identical dataset distribution for both dimensions, which means that both parameter values can be used in the discovering clusters step.

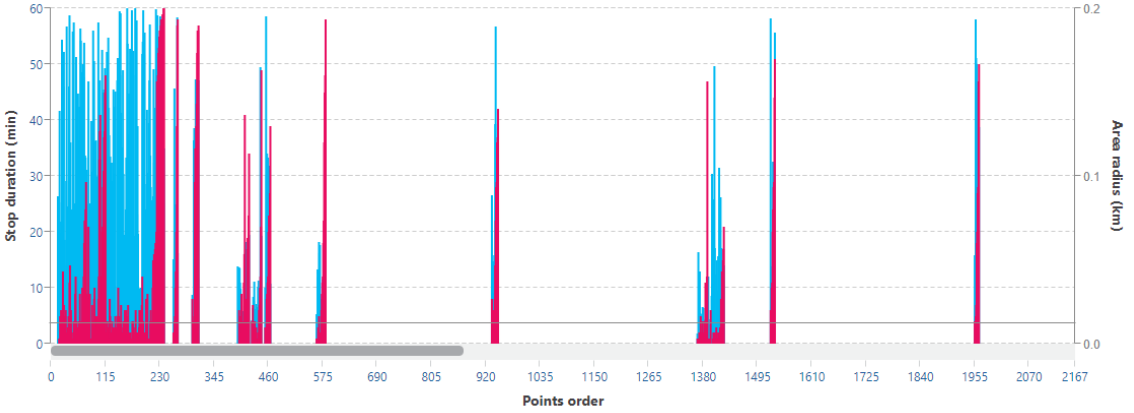


(a) smaller generating distance $\varepsilon = 500$ m, $MinPts = 30$

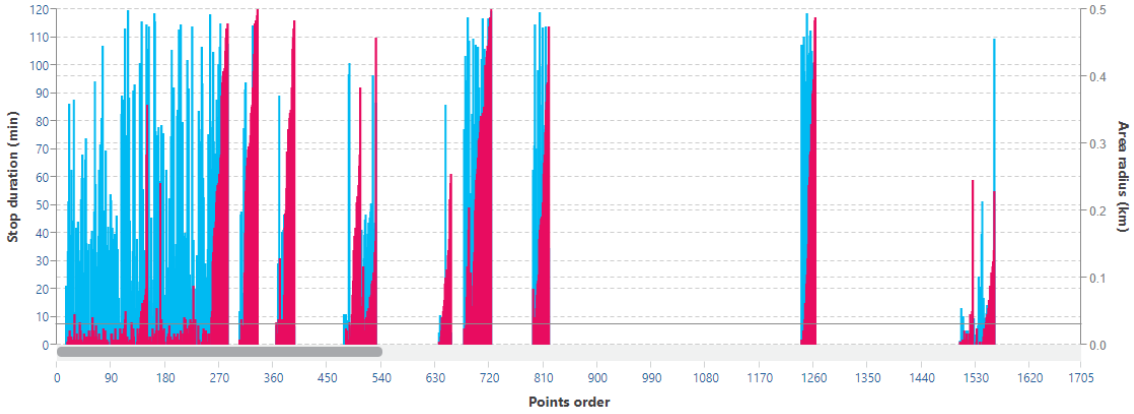


(b) bigger generating distance $\varepsilon = 800$ m, $MinPts = 15$

Figure 6.2: Spatial OPTICS algorithm results for the T-drive dataset



(a) smaller generating distance $\varepsilon - spatial$ (blue) = 200 m, $\varepsilon - temporal$ (pink) = 1 hour, $MinPts$ = 10



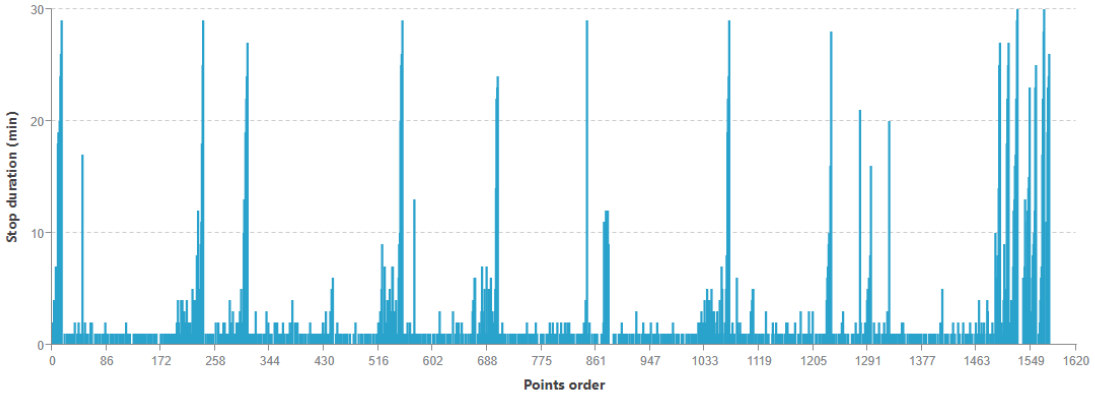
(b) bigger generating distance $\varepsilon - spatial$ (blue) = 500 m, $\varepsilon - temporal$ (pink) = 2 hours, $MinPts$ = 20

Figure 6.3: Spatial Temporal OPTICS algorithm results for the T-drive dataset

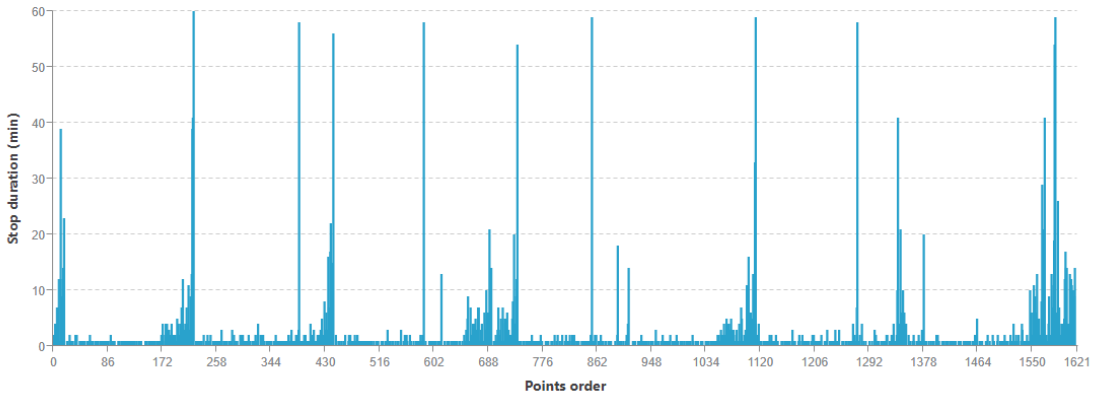
BerlinMOD

The dataset contains 99 302 stops in total, but for the purpose of this experiment the stops duration was limited to t_{min} = 20 minutes and t_{max} = 3 hours. This reduced the amount of stops to 1 620 and the number of trajectories to 434. As the dataset is not as big as the previous one, the bounds for $MinPts$ could be smaller. In this experiment a range of 4 - 8 points will be considered.

The first experiment was conducted with the temporal dimension and, as illustrated on the reachability plots Fig. 6.4 (a, b), the temporal dimension is not sensitive to ε and $MinPts$ parameter's change. Next, the dataset distribution by the spatial dimension was analysed. For a smaller generating distance of ε = 400 metres and a high number of $MinPts$ = 7 only few clusters can be found (Fig. 6.5a). Therefore either the generating distance should be bigger or the number of $MinPts$ has to be decreased. The reachability plot configured with the parameter of a bigger generating distance ε = 800 metres (Fig. 6.5b) does not have defined silhouettes of valleys and it is difficult to figure out the approximate number of clusters as no large clusters were detected.

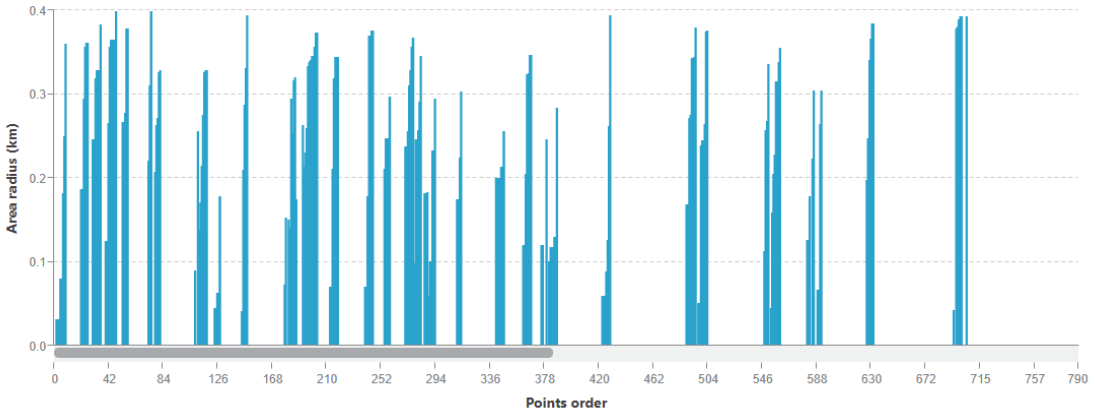


(a) smaller generating distance $\varepsilon = 30$ minutes, $MinPts = 8$

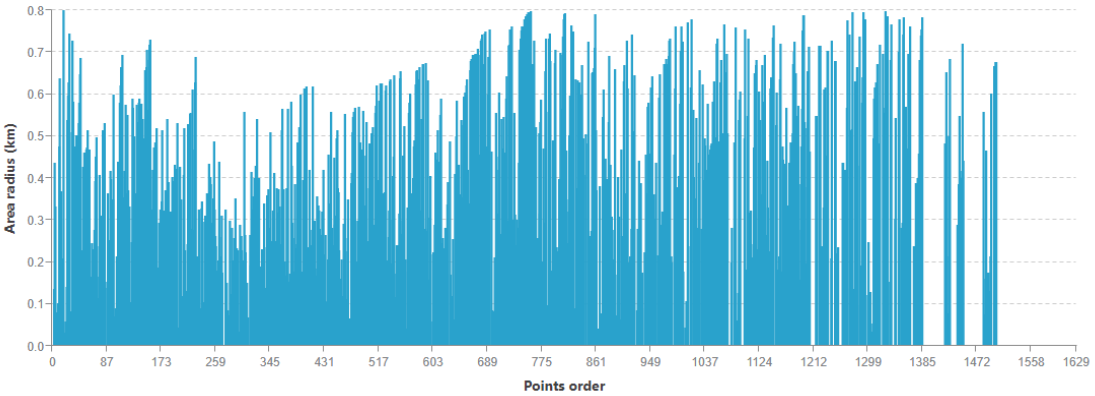


(b) bigger generating distance $\varepsilon = 1$ hour, $MinPts = 4$

Figure 6.4: Temporal OPTICS algorithm results for the BerlinMOD dataset



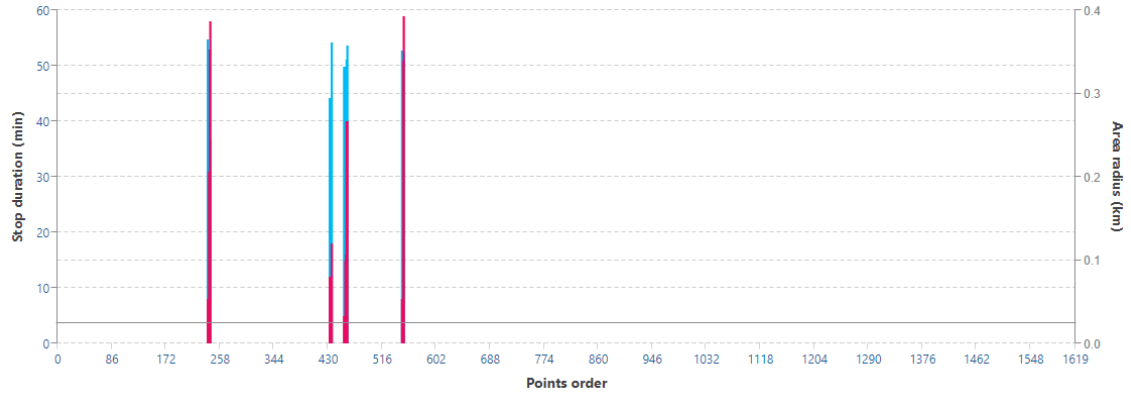
(a) smaller generating distance $\varepsilon = 400$ meters, $MinPts = 7$



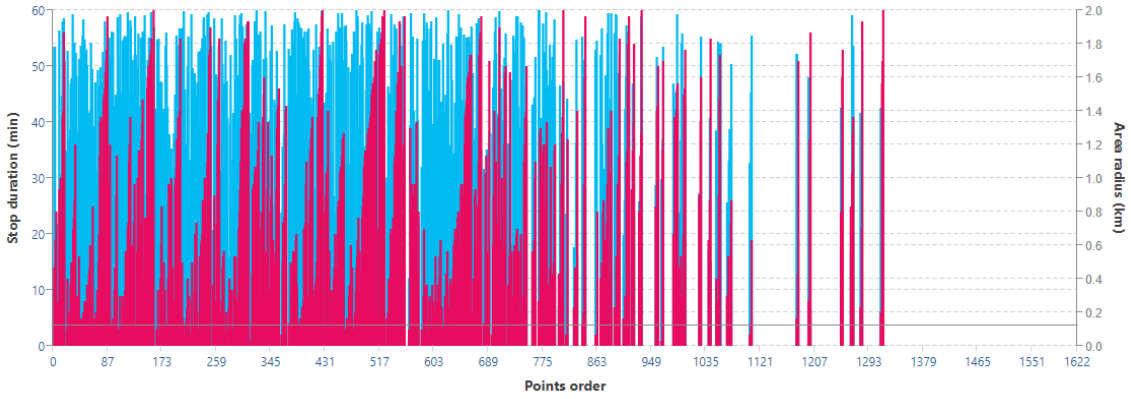
(b) bigger generating distance $\varepsilon = 800$ meters, $MinPts = 4$

Figure 6.5: Spatial OPTICS algorithm results for the BerlinMOD dataset

The last experiment with two dimensional data was conducted for a smaller generating distance $\varepsilon = 400$ metres and $MinPts = 4$. The reachability plot in Fig. 6.6a shows only a very few clusters. The followed up test with increasing just one parameter of ε to 2 kilometres gives better results. In general, based on the OPTICS reachability plot outputs, we can come to a conclusion that this dataset is not appropriate for clustering in order to discover semantic locations as it requires a high ε parameter value for the spatial dimension to discover clusters.



(a) smaller generating distance $\varepsilon - spatial = 400$ m (blue), $\varepsilon - temporal = 1$ hour(pink), $MinPts = 4$



(b) bigger generating distance $\varepsilon - spatial = 2$ km (blue), $\varepsilon - temporal = 1$ hour (pink), $MinPts = 4$

Figure 6.6: Spatial Temporal OPTICS algorithm results for the BerlinMOD dataset

6.2.3 Clustering and semantic locations discovery

In this step of the analysis, the clustering algorithm is applied to datasets with parameters evaluated in the previous chapter. Followed up semantic locations discovery was performed to provide clusters with semantic context.

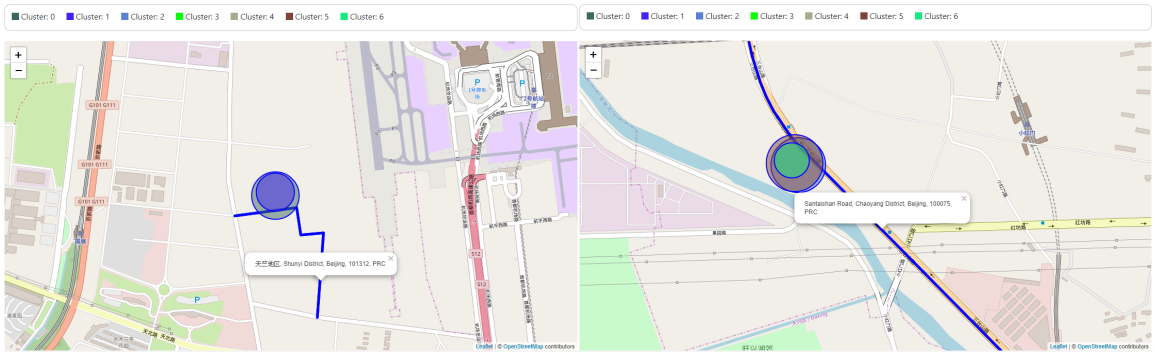
T-drive dataset

For the T-drive dataset the following clustering parameters were used: ε-temporal = 1 hour, ε-spatial = 200 metres and $MinPts = 10$ points. The clustering algorithm applied for Feb. 2 - Feb. 4, 2008 finds various number of clusters per day (on average 5 - 11) in different districts of Beijing, but some clusters are daily detected in the same areas. These clusters are represented in Fig. 6.7 - Fig. 6.8.

Clusters in Fig. 6.7a are located in approximately 2 kilometres to the Crowne Plaza Beijing International Airport. Since the area is the same, the points are grouped into two clusters by the time dimension - taxis from the one cluster were parked there during the afternoon hours (13:00 - 15:00), while another group of taxis was parked during the evening hours (17:00 - 22:00) on Feb. 2. On Feb. 3 evening hour and afternoon hour clusters are merged together and an additional cluster in the morning hours (10:00 - 11:00) is discovered in this area. On Feb. 4 only one large cluster is detected containing taxis' stops in time

interval 13:00 - 23:00. Analysing the time intervals in clusters, it is possible to make an assumption that the detected area is actively used by taxis for parking, but as this area does not adjoin any significant place, the discovered semantic location is the street name with the name of the district.

Clusters in Fig. 6.7b discovered on Santaishan Road contain taxis' stops, which on Feb. 2 occurred during the afternoon hours (13:00 - 14:00, 16:00 - 16:30) and during the evening hours (18:00 - 19:00). On Feb. 3 taxis' stops are grouped by morning hours 9:00 - 10:00 and afternoon hours within intervals 13:00 - 13:30, 15:00 - 16:00. On Feb. 4 there are three groups of taxis' stops in time intervals 05:30 - 06:30, 09:00 - 11:00 and 13:00 - 14:00. Short time intervals can signalize traffic jams in this area.



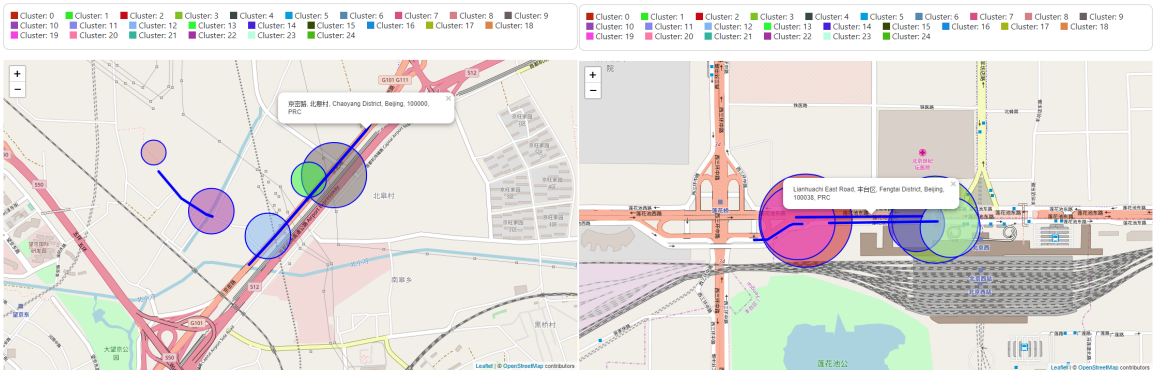
(a) (b)

Figure 6.7: Clusters discovered with parameter $MinPts = 10$

Another group of clusters is shown in Fig. 6.8a for the Feb. 3. With the clustering parameter $MinPts = 10$ only two overlapping clusters are discovered, but with decreasing $MinPts$ to 7, three more clusters are detected. Overlapping clusters differ by time, within one cluster taxis' stops during the morning hours 09:00 - 10:30 are detected, within another cluster taxis' stops are found at night hours in the interval 01:00 - 03:00. On Feb. 4 again two overlapping clusters are discovered in the time intervals 02:30 - 04:00 and 09:00 - 10:00.

The last one group of clusters is detected on Lianhuachi East Road and is shown in Fig. 6.8b. Two clusters on the left differ by the time intervals of 18:30 - 19:30 and 21:30 - 22:30. Another group on the right is clustered by the time intervals 07:00 - 08:00, 21:00 - 22:00 and 23:30 - 00:30. On Feb. 4 three clusters are detected in this area in the time intervals 16:30 - 17:00, 20:40 - 21:40 and 23:00 - 00:50.

For all described clusters the discovered semantic locations are names of roads or streets with districts, which are represented on a map as line segments geometry. There are no significant locations discovered within bounding boxes of clusters by the Overpass provider as taxis were mostly detected either on the road standing in a traffic jam or parked in some residential area. Therefore the reverse geocoding technique by Nominatim was applied to obtain the semantic locations nearby the points of the clusters' centroids.



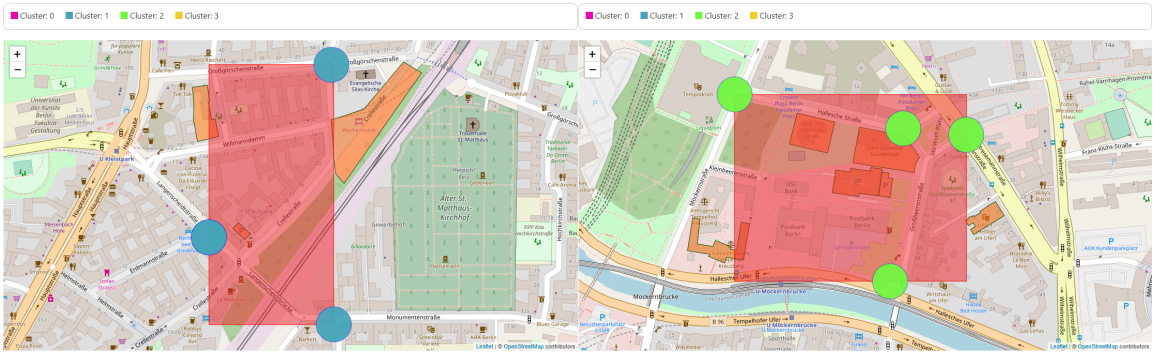
(a) (b)

Figure 6.8: Clusters discovered with parameter $MinPts = 7$

BerlinMOD dataset

The clustering for the Berlin MOD dataset was done with the following parameters: ε-temporal = 1 hour, ε-spatial = 400 metres and $MinPts = 4$ points. The evaluation of these parameters in the previous chapter produces only a few discovered clusters. Increasing the ε-spatial would significantly change the situation, but it contradicts our goal to discover semantic locations in small areas.

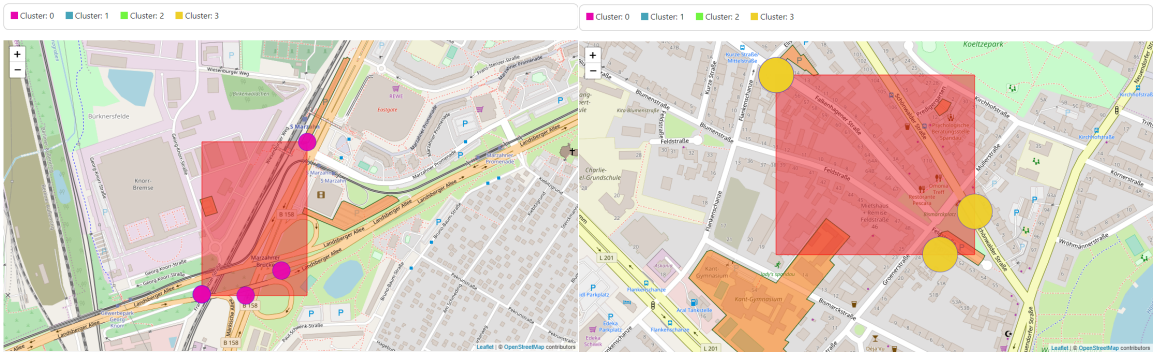
The total amount of discovered clusters is four. These clusters are shown in Fig. 6.9 - Fig. 6.10 with their bounding boxes, points and discovered semantic locations.



(a) (b)

Figure 6.9: Clusters' bounding boxes with points discovered with parameter $MinPts = 4$

It is obvious, that ε-spatial = 400 metres is already quite a big distance to discover semantic locations, which adjoin the particular cluster. The dataset consisting of 434 trajectories appears to be too small for the density-clustering analysis and discovery of semantic locations. Nevertheless, we can see a different pattern in discovered locations in comparison to the T-drive dataset. As clusters were detected in the city, it was possible to obtain the geometry and names of significant locations nearby. The cluster in Fig. 6.9a contains a marketplace and a parking area, in Fig. 6.9b are two different parking areas, a theatre, a school and a courthouse, in Fig. 6.10a two large parking areas and a smaller one were detected and in Fig. 6.10b are three parking areas and a school.



(a) (b)

Figure 6.10: Clusters' bounding boxes with points discovered with parameter $MinPts = 4$

Chapter 7

Conclusion

Thesis conclusion

In this work, we proposed a framework for trajectory data preprocessing to discover semantic locations in raw trajectories. The framework consists of two central modules: discovering interesting physical locations by utilizing a density-based algorithm and semantic enrichment of trajectories by querying the online GIS database OpenStreetMap via Overpass API and Nominatim API.

The framework utilizes the idea of semantic-enhanced clustering [10], but is based on the spatio-temporal clustering algorithm ST-DBSCAN. It allows to eliminate scenarios of discovering clusters in regions which have high throughput (e.g. gas stations), but are not densely filled. Another benefit results from using the bounding box of discovered clusters to query semantic locations nearby. This approach reduces the amount of queries sent to online databases which are usually generated by the reverse-geocoding method described in 3.2.3. Besides, the response returned from the online database will contain accurate data on buildings' street addresses and does not depend on availability of yellow pages web resource.

The proposed approach was implemented as a standalone Java application designed in REST (Representational State Transfer) architectural style and provides following features:

- Discover places with vehicle's stops
- Evaluate the stops distribution in the dataset and discover dense regions in raw trajectories by clustering vehicles' stops
- Discover semantic places nearby the dense regions with the ability to filter them
- Project clustering results with discovered semantic locations on a map

The implemented solution was tested on two different datasets - a database of taxis' trips in Beijing and synthetically generated data of vehicles' trips in Berlin. The clustering results are dependent on input parameters and therefore the implemented heuristics for evaluating the dataset distribution appeared to be very helpful in choosing the optimal parameters for the clustering. Since the setting of appropriate inputs is also dependent on the data analysis goals, it was easier to detect the smallest spatial $\varepsilon - neighbourhood$ value for each dataset, where clustering is possible. Evaluation of the semantic locations discovery module shows that discovered locations are dependent on the nature of objects in the dataset. Semantic locations discovered for taxis mostly contain names of roads, streets

and districts, while the second dataset contains the names and addresses of buildings, where vehicles' stops were detected.

Potential extensions and profitability in future work

The framework was developed in a way that it can become a part of data mining process, which requires trajectories integrated with semantic context. For example, studying behaviour patterns in street traffic or scene partitioning depending on frequently visited areas in the city. The application allows to export prepared data in JSON (JavaScript Object Notation) format, which is a lightweight data-interchange format. Although initially the developed framework was not intended to use as a traffic jams discovery tool, during experiments on real dataset with taxis, some possible traffic problems on roads were discovered.

The framework can be improved in terms of implementation of automatic techniques of the cluster-order analysis, OPTICS-ξ, which is able to identify start-of-cluster and end-of-cluster values and then combines matching regions into clusters. It will significantly improve the dataset distribution analysis by the reachability plot. Another improvement can be made in the semantic locations discovery by putting more strict conditions on queries to the OpenStreetMap services or by defining a dictionary containing special categories of required semantics. Sufficient improvement can be made in terms of performance by running the OPTICS and the ST-DBSCAN algorithms in parallel.

Bibliography

[1] Alon, J.; Sclaroff, S.; Kollios, G.; et al.: Discovering clusters in motion time-series data. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE. 2003. pp. I–I.

[2] Alvares, L. O.; Bogorny, V.; Kuijpers, B.; et al.: Towards semantic trajectory knowledge discovery. *Data Mining and Knowledge Discovery*. vol. 12. 2007.

[3] Alvares, L. O.; Fernandes, J. A.; Macedo, D.; et al.: A model for enriching trajectories with semantic geographical information. In *in 'ACM-GIS', ACM*. Press. 2007.

[4] Alvares, L. O.; Loy, A. M.; Renso, C.; et al.: An algorithm to identify avoidance behavior in moving object trajectories. *Journal of the Brazilian Computer Society*. vol. 17, no. 3. 2011: pp. 193–203.

[5] Andrienko, G.; Andrienko, N.; Rinzivillo, S.; et al.: Interactive visual clustering of large collections of trajectories. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. IEEE. 2009. pp. 3–10.

[6] Ankerst, M.; Breunig, M. M.; Kriegel, H.-P.; et al.: OPTICS: ordering points to identify the clustering structure. In *ACM Sigmod record*, vol. 28. ACM. 1999. pp. 49–60.

[7] Benkert, M.; Gudmundsson, J.; Hübner, F.; et al.: Reporting flock patterns. *Computational Geometry*. vol. 41, no. 3. 2008: pp. 111–125.

[8] Birant, D.; Kut, A.: ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*. vol. 60, no. 1. 2007: pp. 208–221.

[9] Bogorny, V.; Kuijpers, B.; Alvares, L. O.: ST-DMQL: a semantic trajectory data mining query language. *International Journal of Geographical Information Science*. vol. 23, no. 10. 2009: pp. 1245–1276.

[10] Cao, X.; Cong, G.; Jensen, C. S.: Mining significant semantic locations from GPS data. *Proceedings of the VLDB Endowment*. vol. 3, no. 1-2. 2010: pp. 1009–1020.

[11] Ester, M.; Kriegel, H.-P.; Sander, J.; et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, vol. 96. 1996. pp. 226–231.

[12] Gaffney, S.; Smyth, P.: Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 1999. pp. 63–72.

[13] Kalnis, P.; Mamoulis, N.; Bakiras, S.: On discovering moving clusters in spatio-temporal data. In *SSTD*, vol. 3633. Springer. 2005. pp. 364–381.

[14] Krumm, J.: Trajectory Analysis for Driving. In *Computing with Spatial Trajectories*. 2011.

[15] Lee, J.-G.; Han, J.; Li, X.; et al.: TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*. vol. 1, no. 1. 2008: pp. 1081–1094.

[16] Lee, J.-G.; Han, J.; Whang, K.-Y.: Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM. 2007. pp. 593–604.

[17] Li, Y.; Han, J.; Yang, J.: Clustering moving objects. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004. pp. 617–622.

[18] Liu, J.; Wolfson, O.; Yin, H.: Extracting semantic location from outdoor positioning systems. In *Mobile Data Management, 2006. MDM 2006. 7th International Conference on*. IEEE. 2006. pp. 73–73.

[19] Mazimpaka, J. D.; Timpf, S.: Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*. vol. 2016, no. 13. 2016: pp. 61–99.

[20] Nanni, M.; Pedreschi, D.: Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*. vol. 27, no. 3. 2006: pp. 267–289.

[21] Palma, A. T.; Bogorny, V.; Kuijpers, B.; et al.: A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM symposium on Applied computing*. ACM. 2008. pp. 863–868.

[22] Parent, C.; Spaccapietra, S.; Renso, C.; et al.: Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*. vol. 45, no. 4. 2013: page 42.

[23] Rama, B.; Jayashree, P.; Jiwani, S.: A survey on clustering current status and challenging issues. *International Journal on computer science and engineering*. vol. 2, no. 9. 2010: pp. 2976–2980.

[24] Zheng, Y.: Trajectory Data Mining: An Overview. *ACM Transaction on Intelligent Systems and Technology*. September 2015.
Retrieved from: `https://www.microsoft.com/en-us/research/publication/trajectory-data-mining-an-overview/`

Appendix A

DVD content

The attached CD contains:

- `sminer_source` - source files, as can be obtained from the GitHub repository[1]
- `sminer_executable` - folder with executable application, contains readme.txt manual
- `datasets` - datasets for experiments
- `thesis.pdf` - text of the thesis in .pdf format
- `thesis_latex` - LaTex source files

[1] https://github.com/annaostroukh/SMiner

Appendix B

Manual

B.1 Dataset configuration

1. Click on the **Configuration** tab. There are two options available:
   (a) Choose one option for preconfigured test datasets.
   (b) Input order of columns for custom dataset in corresponding input fields. Note, that ordering starts from 0.
2. Click on **Save** button.

B.2 Extraction of stops

1. Click on the **Extract stops** tab.
2. Click **Browse** to select the dataset from your computer.
3. Click **Upload**.
4. Input required minimum duration of stops in minutes to the corresponding input field. Optionally, input maximum duration of stops in minutes.
5. Click **Extract stops**.
6. Optionally select interval in days for which analysis is going to be conducted. Click **Submit**.

B.3 OPTICS reachability plot

1. Click on the **OPTICS reachability plot** tab. There are input fields for three types of reachability plots available:
   (a) Temporal plot.
   (b) Spatial plot.
   (c) Spatial Temporal plot.
2. Input values into corresponding input fields. The description of input fields is provided below:

(a) **Epsilon (maximum temporal distance in minutes)** - sets the neighbourhood value for the temporal dimension. For example, for the value set to 20 minutes, the reachability plot will display the dataset distribution, where 20 is the maximum reachability distance on the Y axis.

(b) **Epsilon (maximum distance in km)** - sets the neighbourhood value for the spatial dimension. For example, for the value set to 0.4 km, the reachability plot will display the dataset distribution, where 0.4 is the maximum reachability distance on the Y axis.

(c) **Minimum amount of data points** - sets the required minimum amount of stops in a cluster.

3. Click on the corresponding button near the plot settings.

B.4 ST-DBSCAN clustering

1. Click on the **Extract semantic locations** tab.

2. Input values into the corresponding input fields:

   (a) **Epsilon of temporal distance in minutes** - sets the neighbourhood value for the temporal dimension.

   (b) **Epsilon of spatial distance in km** - sets the neighbourhood value for the spatial dimension.

   (c) **Minimum amount of data points** - sets the required minimum amount of stops in a cluster.

   For example, settings of 20 minutes, 0.4 km and 5 points will discover clusters, where the distance between adjoining vehicles is equal to or less than 0.4 km and the time difference between vehicles' stop points is equal to or less than 20 minutes. A group of points will be considered as a cluster when this group contains an amount of points equal to or more than 5.

3. Click the **Run ST-DBSCAN** button.

B.5 Semantic locations discovery

Semantic locations discovery is available on the **Extract semantic locations** tab after discovery of clusters.

1. Optionally put required distance in kilometres to extend clusters' bounding boxes boundaries. If no value is set, the semantic locations which are inside and intersect the bounding box are discovered.

2. Click the **Extract all locations** button to extract locations for all clusters in a list or click the **Extract semantic locations** link near the corresponding cluster to extract locations of a particular cluster.

3. Click the **View locations** link to see the discovered locations.

   In the appeared modal window select a location or multiple locations (holding **Shift** button). Move selected locations to the **Chosen locations** column of the modal window. Click on the **Save changes** button to save selected locations.

4. Click on the **Show map** button to display a map. There are options available for the data visualization on a map:

   (a) **Show clusters' area** - shows circle areas of clusters where the centre of the circle is a cluster's centroid.

   (b) **Show clusters' points** - shows points in a cluster.

   (c) **Show bounding boxes of clusters** - shows rectangular bounding boxes of clusters.

   (d) **Show stops info** - shows information about stops in clusters containing the location, moving object ID, stop ID and duration of the stop.

   (e) **Show locations' geometry** - shows geometry of semantic locations.

5. Click on the **Export clusters' data to JSON** button to export the clustering result integrated with semantics to a JSON file.