# Modelling and visualizing macroscopic city movements

by

**Ananya Bhanja Chowdhury**
Matriculation Number 385932

**Piotr Mrówczyński**
Matriculation Number 387521

**Gabriel Vilén**
Matriculation Number 387555

A project documentation submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

August 6, 2017

Supervised by:
Bianca Lüders
Friedhelm Victor
Prof. Dr. Axel Küpper

Service-centric NETWORKING

Technische Universität Berlin

# Contents


**1 Introduction** **1**

**2 Related Work** **3**
2.1 Human Mobility within the city . . . 3
2.2 Approaches to stop detection for location-based services . . . 3
2.3 Clustering . . . 4
2.3.1 DBSCAN on spark . . . 5
2.4 Graph Analytics: For analyzing human mobility data . . . 7

**3 Concept and Design** **9**
3.1 Type of data available . . . 9
3.2 Stop location detection . . . 10
3.2.1 Mobility index based approach . . . 10
3.2.2 Human Behavior based approach . . . 11
3.3 Clustering of stops . . . 12
3.4 City movements graph . . . 12
3.4.1 Basic concepts . . . 12
3.4.2 Graphical representation and analysis . . . 13

**4 Implementation** **15**
4.1 Scripts for data analysis in python and QGIS . . . 15
4.2 Architecture - batch processing in Apache Spark . . . 15
4.3 Determining stop location . . . 16
4.4 Clustering . . . 17
4.5 Analyzing the user movement graph . . . 18
4.6 User Interface . . . 18

**5 Evaluation** **21**
5.1 Location data characteristics for Berlin area . . . 21
5.1.1 Data analysis . . . 23
5.2 Pre-filtering of anomalies . . . 25
5.3 Stop detection algorithm . . . 25
5.3.1 Mobility Index approach . . . 25
5.3.2 Human Mobility approach for long/short distances . . . 28
5.4 Stop location clustering . . . 29
5.4.1 DBSCAN . . . 30
5.4.2 Evaluation of *eps* . . . 30
5.4.3 Evaluation of *minPts* . . . 31
5.4.4 Extending DBSCAN with areas . . . 31

5.5 User movement graph . . . . . 35

**6 Conclusions** **43**
6.1 Stop detection algorithm . . . . . 43
6.2 Clustering of detected stops . . . . . 44
6.3 Graph analysis . . . . . 45

**7 Future work** **47**

**Bibliography** **49**

# 1 Introduction

The last several years brought revolution in positioning and geospatial technologies, enabling enterprises to boost their core businesses with the potential these technologies are bringing. One of the adoptions are Location-Based Services (LBS), which take advantage of mobile devices, wireless, GSM and GPS technologies altogether. Due to the amount of data these positioning technologies generate, companies started looking at efficient ways of storing and processing the data - such as S3 storage, NoSQL databases, Apache Spark processing engines, and using microservices architecture.

The idea behind this project is to provide algorithms and architectural software solutions which will be able to process big amounts of data, visualize it and give macroscopic view on the collected data across many users. Particularly, we look at where the users stop and what are the significant places considering their movements in a macroscopic city view.

Location Based Services have an advantage that users do not have to enter their position manually, it is automatically collected (using all positioning technologies available in the mobile device), and used to generate personalized information. This service is of significant value for stores and shopping facilities, airports, tourism, car-sharing, payment systems, and more. Positioning technologies, being a core for LBS, are a trade off between accuracy of the positioning and its applicability indoor/outdoor. Bluetooth and Wifi technologies provide highest positioning accuracy, however they can only be used indoors. On the other hand, GPS that also providing relatively good accuracy can only be used outdoors. Cellular, GSM networks can be used indoors and outdoors, however they provide moderate to low accuracy (macroscopic picture) [7]. The are two types of LBS, proactive services which requests whenever user enters certain geographic area (approaches a shop or service he might be interested in or when he leaves the shared-car he has been renting), or reactive, in which user explicitly requests to use service (to find nearest point of interest). Thus, for most application, accuracy of GSM or WiFi/Bluetooth network is more then enough, and GSM is not playing the major role [13]. Application using LBS, at the given instance of time will try to obtain the location using the positioning technique which is currently available and having highest accuracy at that time. This implies, that the data collected cannot be considered as the one representing the exact location of the user at the specific moment in time, but only gives information, that user has entered/moved to a new geographical area in case of proactive LBS or that user is somewhere in the specified area in case of reactive LBS.

# 2 Related Work

This section introduces the reader to the state of the art of existing literature on human mobility patterns, approaches to stop detection and network graph analysis.

## 2.1 Human Mobility within the city

Understanding of human mobility patterns is essential for implementing various environmental changes [20], assessing and improving existing mobile network protocols [12] or to learn significant locations and build movement prediction models [10].

The study [12] points out that it is more human intentions instead of geographical factors which produce heavy-tailed distribution of human mobility (human tend to move by about similar distance from point to point, with heavy tailed exceptions). It also assumes, that this is caused by the power-law tendency of popularity of locations people visit or simply human interests. The study confirms, along with the other study [20], that both walking distance and walk duration follows heavy tailed distribution, where the average walking speed in the city for man aged 40 is 1.4 m/s (5 km/h), and slowest one is for elderly (with walking impairment) below 1 m/s (3.6 km/h) [4].

We find, that analyzing human mobility on short distances might be important for approaches to stop detection.

## 2.2 Approaches to stop detection for location-based services

The state of the art within works in the area of learning significant places for people usually implies usage of some sort of stop detection algorithm, which is usually bound to the way how the data is being acquired from mobile devices and the technology behind acquisition process.

As an example, study [5] proposed approach based on geo-grid of cells and continuous GSM trajectories matching certain cells. As a predicate for assuming a stay at certain cell, simple duration threshold has been used. Furthermore, post processing in the form of significance mining has been applied to extract only the most important places for the certain user and filter less significant ones.

Similar approach, however using continuous GPS trajectories, targets at finding places where user spends their time. Their assumption was that mobile device outside of building generates location data until the user enters a building, resulting in a time gap between recorded locations. To pre-filter short stays at location, duration threshold has been used. Using clustering concept, they find the significant places out of pre-filtered data.

In the paper [11], non-continuous position acquisition technique is used and movement of mobile devices between cells (handovers) is considered. To detect the periods of slow movements of stops at the specific location, parameter called Mobility Index is considered.

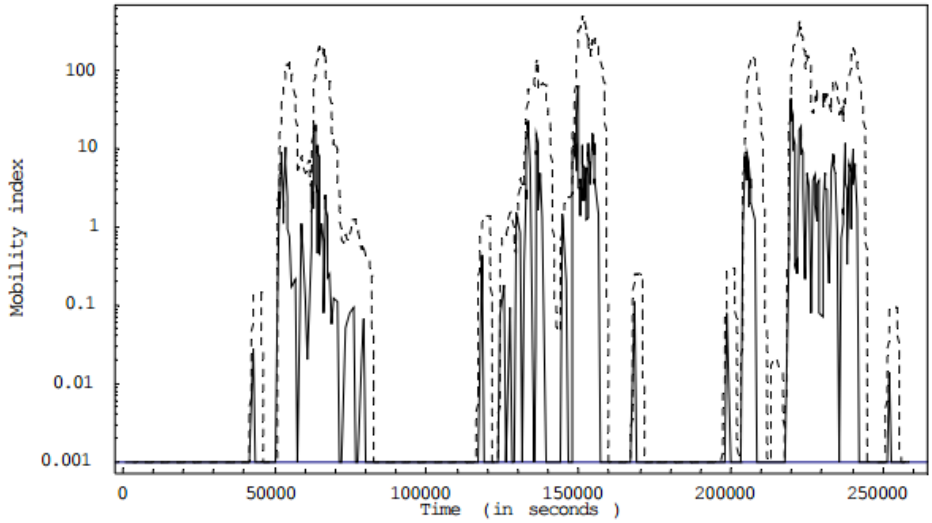


**Figure 2.1:** Mobility Index. Source: [11]

In the areas with many GSM cells (mainly areas with high population density), the mobile terminal may change cells frequently (which might identify movement) or stay at one cell for longer time. Given a set of consecutive records, the mobility of a user can be estimated by calculating the Mobility Index over a pre-defined time period (sliding window). The Mobility Index is defined as the sum of the "distances" between each record and the previous ones, where the "distance" is the inverse of the time spent on each cell. If the value of the index is below a certain threshold, it is assumed that the object stopped or moved slowly. In the publication, sliding window of 10 minutes and a mobility index threshold value of 6 has been used for certain area. The obtained results were in agreement with actual movements during more than 90% of the time.

Rapid drop in mobility index represents a set of points (restricted area) in which user has been moving slowly or stopped. Increase in mobility index means that user changed position from restricted area and is moving.

## 2.3 Clustering

Clustering can be described as "*the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)*" **[clustering]**.

There are different types of clustering (such as centroid, distribution, or density-based), and different algorithms with parameters that can be applied (such as K-means, DBSCAN, or OPTICS). However, they all work to achieve the same goal of structuring (clustering, grouping) similar objects and neglecting outliers from these groups. The algorithm and parameters are application sensitive and can produce highly different clusters. The difficulty of DBSCAN is

figuring out the two parameters, the minimum points per cluster (*minPts*) and the radius of the clusters (*eps*).

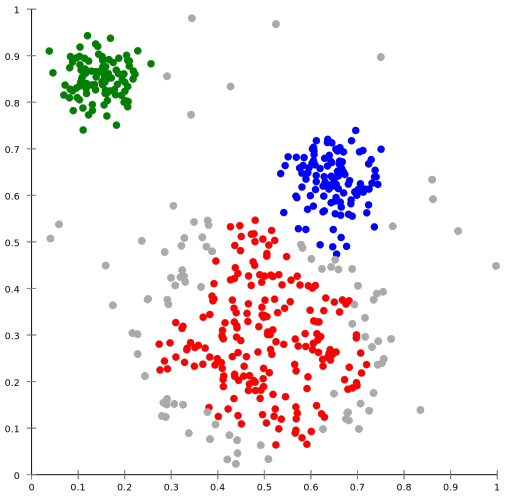


**Figure 2.2:** Clustering applied to a dataset.

### 2.3.1 DBSCAN on spark

In this project we used an implementation of the DBSCAN clustering algorithm on top of Apache Spark. The imp,enetation is called DBSCAN on Spark [8] and loosely based on the paper from *He, Yaobin, et al. "MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data"*.

This implementation of DBSCAN runs in parallel by splitting the data space into boxes, using the number of boxes as a cost estimator for the algorithm. Each box then grows to include one *eps* in it. After each box and its points has been determined the traditional DBSCAN algorithm is run on the points in each box. Finally it examines the intersection points betweeen boxes and merges the result together [9]. See figure set Figure 2.8 for a visual guide.

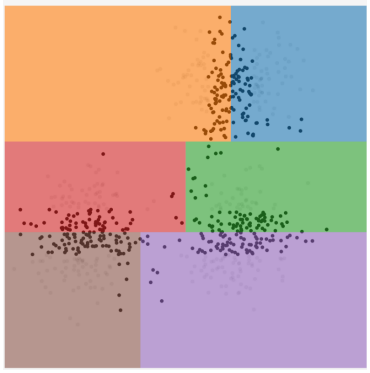

**Figure 2.3:** Step 1. DBSCAN on Spark assign the data space into boxes.

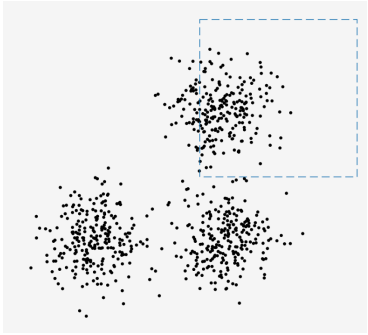

**Figure 2.4:** Step 2. Each box grows to include the points that are within one *eps* of it.

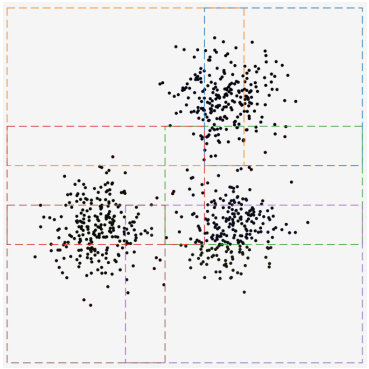

**Figure 2.5:** Step 3. Traditional DBSCAN algorithm is applied in parallel for each box. Each different color represents a different cluster.

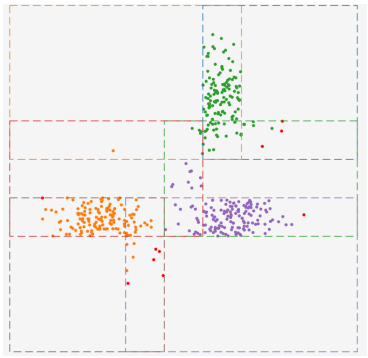

**Figure 2.6:** Step 4. After DBSCAN is done, all points within the borders of two clusters are examined. If they are part of a cluster within two boxes they are merged to one cluster.

**Figure 2.7:** Step 5. Finally all remaining points are labelled with the global identified cluster and we are done. (red points are noise points).

**Figure 2.8:** DBSCAN on Spark [9], maintained by Irving Cordova.

## 2.4 Graph Analytics: For analyzing human mobility data

Graph based models are very commonly used in scientific research related to mobility data. Graph analytics is used to study the relationships between nodes and edges in a network. It helps identify key players in a network, areas of interest, even differentiate between usual and unusual patterns in the data. In this section we present research papers that have used concepts of graph analytics and influenced our work.

One of the most important concepts for analyzing mobility data is an Origin-Destination Matrix(OD). OD matrix and hotzones have been used in the study to analyze urban population migration based on data from Beijing taxi service. The study was presented in the paper [21]. The authors identified hotzones in the city of Beijing by analyzing the most popular pickup and drop points from the trip data. They also analyzed the OD pairs to identify most frequent movement patterns of the residents. Their findings corroborate with their assumptions and some of the most important areas (key transport facilities/financial hubs) got identified as the hotzones.

In a similar study, based on taxi data from Berlin [6], the authors used OD Matrix to compute location-based taxi demand in Berlin. They analyzed time and location based taxi demand patterns, e.g. how the demand pattern varies for the two airports in Berlin. They computed the top 100 OD-relations and analyzed the attractiveness of each route. The study identified important places in the city as the most common destinations.

OD matrix is also used for travel demand estimation problems. In the paper [15], the authors have used mobile phone data of users, based on their daily activity and path choices of users, for their analysis. This data has been used to derive sample OD matrices and subsequently used for traffic assignment.

Apart from human mobility data, graph based models are also very useful for analyzing air traffic as discussed in the paper [18]. The authors discussed a two step based process for building the network graph based on air traffic. In their study, Marzuoli et al. created a network flow model of the airspace. They defined their network as a system of nodes and directed edges. Working with air traffic, they defined network nodes as areas where aircraft are entering or leaving a region and edges as flow corridors used by the aircrafts. This data was used to create origin-destination pairs to determine relative importance of different routes used by aircrafts.

The concepts of OD Matrix, hotspot identification, and modelling raw data into a graph, as are of primary importance for our work.

# 3 Concept and Design

This section is a theoretical introduction to the modules required for macroscopic movements modeling and visualization. There are three main modules in this project which can work independently from each other:

- Stop location detection
- Clustering of detected stops
- City Movements Graph and Analysis

## 3.1 Type of data available

The data supplied to the system is based on tracking and geofencing proactive location-based service, targeted at end users and retailing companies, as discussed in section 2.1. The accuracy of user location is biased not only on accuracy of position estimation used by mobile device, but also by the technological and commercial limitations of the data acquisition strategy of geofences. This means that data is spatially sparse, coarse and its resolution depends on density of geofences.

Thus, recorded data points are associated with unique user id, timestamp, latitute and longitude of the position at which user have been found to move from one area to another.

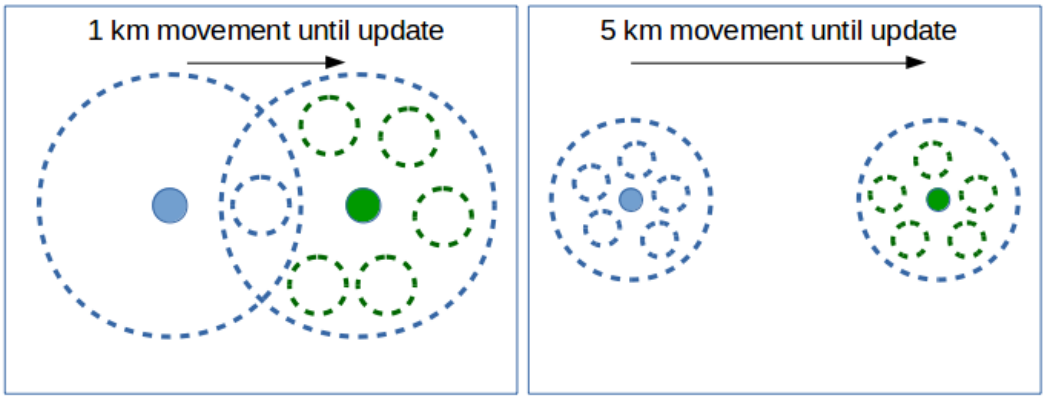


**Figure 3.1:** Continuous and Discontinuous location update after movement to section lacking area details

Because of the characteristics of GPS and other positioning methods, in so called urban street canyons and the like, signal loss with re–acquisition times of 45 seconds to five minutes can be

observed [10]. This could lead to discontinuous recording of the location - Figure 3.1.

Another aspect to consider is the fact that because of commercial application of the available data might be targeted to a specific user group, the macroscopic movements analysis might not be representative for the population in general (for example, position data collected from a running application would probably generate different points the one for finding restaurants).

## 3.2 Stop location detection

In the context of macroscopic movements of people, stop detection is required to identify, based on supplied data source, whether the person under observation has stopped, at what location and how long he stayed at that location. The challenge for stop detection, in the context of proactive location-based services, is the method of data collection. Firstly, each location is recorded due to movement from one area to another. This means, that person might have stopped somewhere in between these points, and stop detection should be able to identify both location and duration of stay. Furthermore, if a user enters an urban canyon, building or move underground, the signal might need to be re-obtained after a movement to new area. This loss of signal can take some time, where points possibly can be spanned over mid-range/long distances. On the other hand, points received over a short distance might mean that a person has been already been moving with a good signal reception .

### 3.2.1 Mobility index based approach

We decided to reuse the concept of mobility index, having artificially created areas in the city. In this case, the location point is recorded whenever moving into a new geographical area and requesting an update.

In a populated area, the user can move from one area to another, and the movement can take from seconds to several minutes. Thus, if within a certain period of time (called mobility index time window) the user stays in the same area or changes the areas with small pace (mobility index is below certain threshold) it can be assumed that the user is immobile or approaching stop location and having a stay somewhere within that time window (restricted area).

There are two approaches to calculate mobility index - having equally spaced windows in time, or sliding window.

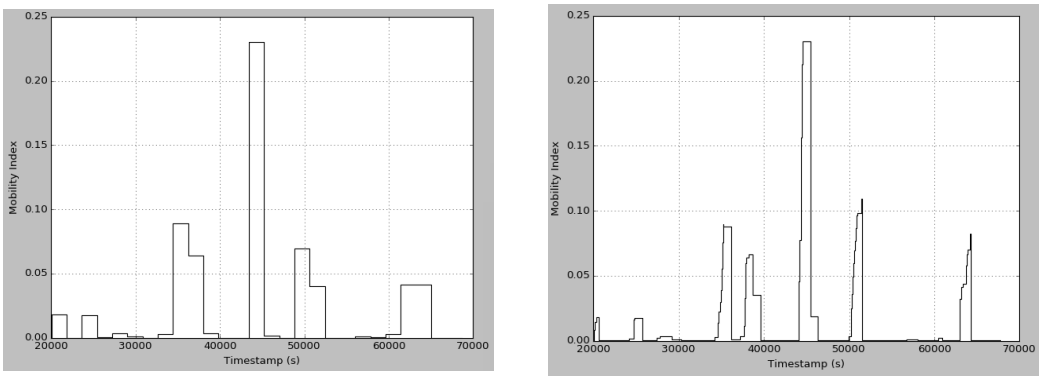


**Figure 3.2:** Mobility Index calculation using equally spaced windows in time (left) and using sliding window over time (right). Time Window of 30 minutes using the same data set for one unique user

In the first approach, for a single user, minimum and maximum timestamp of sorted sequence is found. The function is creating equally spaced bins of that value within the maximum and minimum timestamps. Each point is visited and assigned to the proper timestamp-bin according to its timestamp and duration to the previous point. The disadvantage of this approach is that points and their duration are randomly placed in bins, which could lead to representing mobility of fixed window, instead of mobility history for points.

In the second approach each point of a user is being visited and temporarily marked as "current position". For each current position, all points which are distant in time by the mobility index window in the past are assigned to the sliding window. The advantage of this approach is that it precisely represents the mobility history in the past duration window.

### 3.2.2 Human Behavior based approach

This section describes the approach and conditions for movement between two points that has to be satisfied in order for the point to be a considered a stay.

The section 2.1 discusses, that in average, a walking human is traveling with the speed of over 3km/h up to 5km/h. However for movement over long distances, a user will most likely use a city or private transport, thus her speed might vary drastically in that time/distance period.

This approach states, that if movement between two points has been recorded within short distance, and movement of speed is below a certain threshold, it might mean that user has stopped or has been moving very slowly within a restricted area. This is called a **possible stop** for that location between these points.

For movements between points over mid/long distances recorded, a user might have stopped at the first location, moved with different speeds between points and then continued moving (proactive localization is used). As an example, movement over 6km might require up to 10 minutes with transportation. If the user stopped at previous location for 20 minutes, the total movement between point took 30 minutes, which in average is about 12 km/h. Universal

speed threshold is not able to detect stops in these kind of examples, since travels over higher speeds with stop at previous location would result in relatively high average speed.

## 3.3 Clustering of stops

After our stop detection algorithm has detected the proper stops we need to analyze them. Our goal is to retrieve the overall movement patterns of a population within a certain area, which means we are interested in **macroscopic** movements and not microscopic (individual) movement. If we would consider every individual our analysis algorithms would get fed with a lot of noise (outliers) which would make data mining and prediction difficult and inaccurate. Meaning, *Macroscopic Movements* are not interesting in tracking individuals but rather seeing the big picture.

Once the clustering algorithm has been applied and we have the overall picture, we can see which stops are most popular, where people are moving from them, and how long they stay at a certain location. One could then perform some interesting graph analysis on these locations, answering questions such as "what is the probability that a person moves from point $A$ to $B$?", or "what is the average stop time at location $A$?".

To achieve these clusters, which represents our interesting stop locations, we need to apply some clustering algorithm. DBSCAN, described in chapter 2, was our approach of achieving these clusters.

## 3.4 City movements graph

This section introduces the concepts and terms related to graph analysis used in this project. Our target is to identify popular locations and routes in a city and determine movement patterns of people. The stop points identified by the stop algorithm correspond to actual physical locations in the city with individual latitude and longitude values. These stop points are clustered to identify places of interest in the city. A collection of such stop clusters can be used to chart the movement patterns of the residents.

### 3.4.1 Basic concepts

The first step of creating a user movement graph is to create vertices and edges.It is important to outline how the vertices and edges have been selected for creating the city graph.

- **Vertices** - Clusters generated by the DBSCAN algorithm are considered as vertices.
- **Edges** - For our analysis we are using an edge to represent the trajectory of movement. A trajectory or simple edge is created by joining successive points from individual user movement data.
- **Graph** - The edge set has been used to create a simple directed graph. Direction of movement is considered from the start point going towards the end point or next point in sequence.

An important aspect about our data is that it has been captured over a period of 24 hours.

In other words, our user data does not vary over days of a week. In case of such temporal variation of data, day of the week has to be considered while creating the edges.

### 3.4.2 Graphical representation and analysis

We are using various characteristics of a graph like in- and out-degrees, strongly connected components, betweenness and PageRank [19] for our analysis. We are also using the concept of Origin-Destination Matrix and using the trip counts for modeling the network graph.

As mentioned in section 2.4 , OD matrix is widely used in mobility data analysis and can be used to predict movement patters, important routes etc. From incidence relation between an edge and a vertex, we can calculate degrees(in and out degrees), and can identify hotspots in a city. For example: the central train station or airport of a city will have a high in and out-degree, due to high number of people traveling to and from that place.

We will be analyzing our directed graph for strongly connected components, betweenness and PageRank. Strongly connected component can give us the areas of the city that are highly interconnected. This connection or route can be roads or transportation networks. For our mobility data, PageRank can provide us with a set of vertices that are most influential or highly accessed. Similarly, betweenness can provide us with nodes that are part of many shortest routes and hence influence the routing decisions by users. For a city, these vertices can correspond to transportation hubs, central business or administrative districts, junction or crossover points.

# 4 Implementation

This chapter provides the reader understanding of the project organisation, architecture and algorithms used.

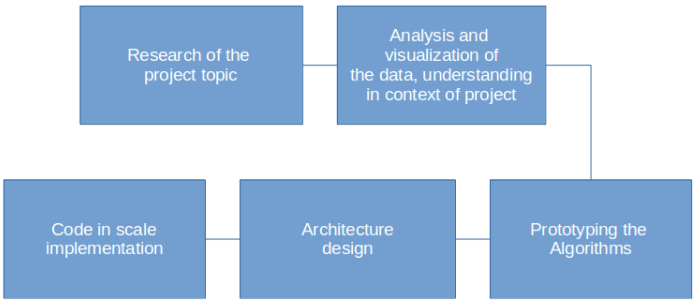


**Figure 4.1:** Project workflow

At first, we researched the topic of LBS, stop detection, clustering and graph theory. Second step was to understand the data, start prototyping the algorithm for stop detection and visualize. Third step was to build four independent components using Apache Spark API and Scala - stop detection algorithm, clustering, graph analysis and web server. Fourth step was to build a scalable architecture and mock additional 3rdparty components (Minio S3 and Apache Spark containers).

## 4.1 Scripts for data analysis in python and QGIS

To understand the data we have used a set of tools in python [17]. These enabled us to quickly understand the data in general (nubmer of users, avarage number of points, average duration, distance, speed etc) and per user (plotting distances and speeds between points in time). We also extensively used a tool called QGIS in order to visualize data on the map.

## 4.2 Architecture - batch processing in Apache Spark

The architecture of the project follows an elegant microservice concept, using independent deployable components. The project primarily consists of three independent components; S3 storage (a protocol for data storage, in this case open source *Minio* in Docker container), Apache Spark cluster (distributed compute engine over nodes), and a web server with spark jobs and spark-submit. See figure Figure 4.2.

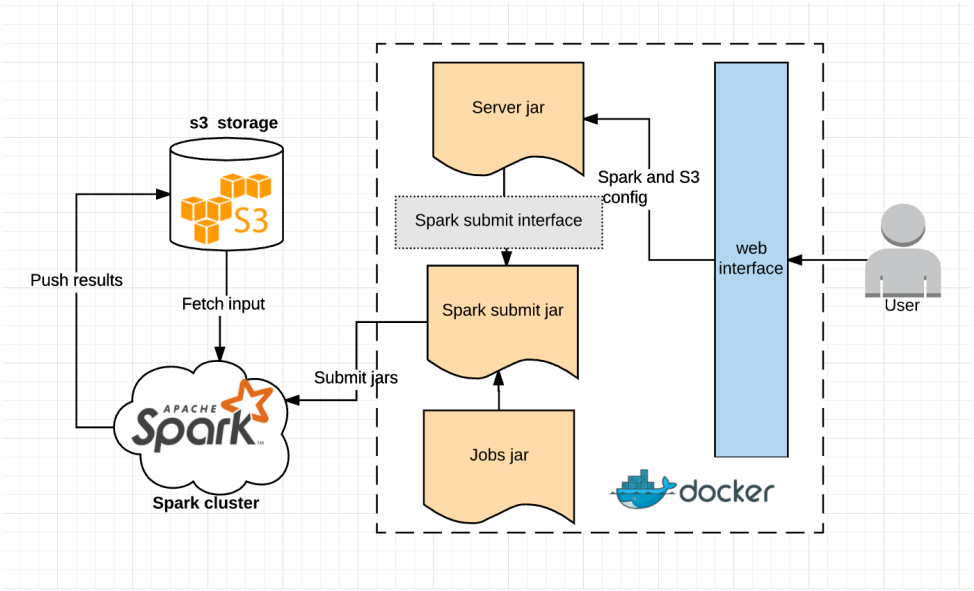


**Figure 4.2:** Architecture of implemented project, using s3, spark, and docker

The stop detection jobs are bundled in jars and submitted through the **Spark submit interface** to the **Apache Spark cluster**. They will there be load balanced over the slaves by the Spark master which fetches the input file from the **s3 storage**. This storage can be local or deployed on a shared cloud service. After the computation is finished the master can push data back into the **s3 storage**.

To easily deploy the application and make it environmental agnostic we bundled everything into the container platform Docker.

Let us describe a basic scenario:

1. User starts the docker containers which will start spark cluster and s3 storage.
2. User uploads the input file on to the S3 storage.
3. User runs docker container containing the project code (macromovements jobs jar, spark submit jar and web server jar)
4. User accesses the web interface, provide configuration such as the url of the input file (in S3) and the url where the Spark master resides.
5. User runs the stop detection job from the web interface by pressing the "setup" and "run" buttons. Output will be displayed using Leaflet Javascript library on the map

## 4.3 Determining stop location

This section discusses implementation details of the algorithm used to identify stops. Figure 4.3 shows block diagram of stop detection algorithm. The input to the algorithm is list of vectors containing at its first 5 elements day of the week, time of day, user id, latitude and longitude of the recorded location.

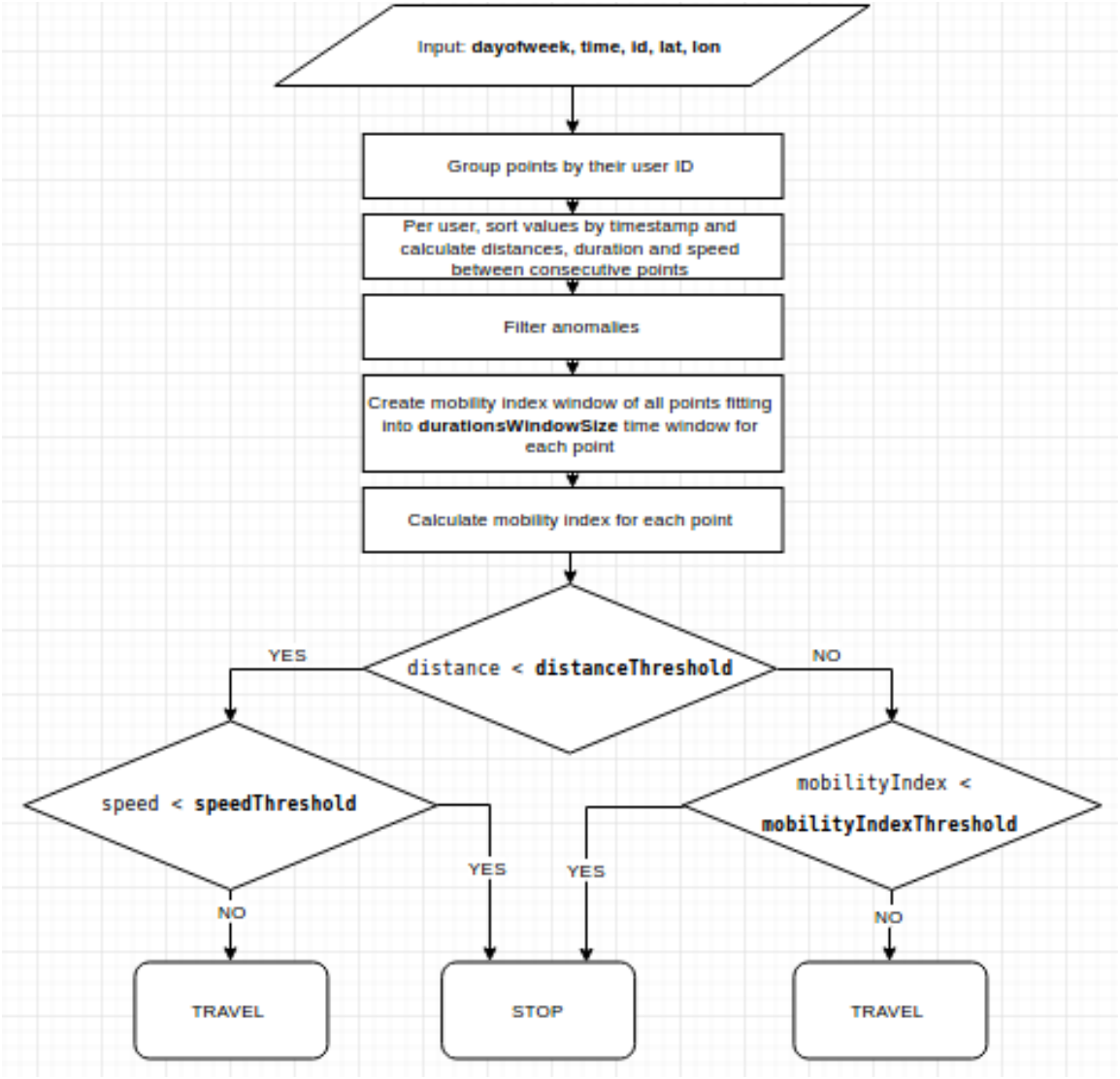


**Figure 4.3:** Stop detection algorithm for proactive localization services.

The first phase of stop detection algorithm is to group points according to user id. As algorithm is implemented in Spark, each user is processed in parallel. Per user, values are sorted according to their timestamp. Distance, speed and duration is calculated for each consecutive pair of points. Values are being filtered for anomalies in the next step. Mobility Index for each point is calculated, as discussed in subsection 3.2.1. Further, decision based on thresholds is made whether the point is stop or not.

## 4.4 Clustering

The implementations of clustering were primarily done by using open source implementations and extending it to our needs. For clustering we used the algorithm DBSCAN.

We wanted a scalable implementation of DBSCAN that could handle large data sets, preferable in parallel. We decided to use Apache Spark, a scalable engine for large-scale data processing [2]. For this we used an implementation by Irving Cordova, explained in Related work, chapter 2.

Along as using this implementation we extended it with an area version, which divided the data set into predefined boxes. Each box were assign different parameters (discussed in Evaluation, chapter 5 and ran in parallel. After each box was finished the results were merged and written to a file along with important output (such as latitude, longitude, user id, cluster id).

Other versions of DBSCAN were also implemented that ran the jobs in batches and stored it in different files, used for evaluating the results and plotting them in the graph visualization tool QGIS.

## 4.5 Analyzing the user movement graph

We have used GraphX and Python for creating and analyzing the user movement graph. GraphX was used for creating the data sets and the graph. Before creating the graph, we filtered the data and structured it in a suitable format.

Filtering was essential at this step as the clustering algorithm identified many stop points as outliers with cluster id=0. We filtered out these outliers from the main data set. With the filtered data, our task was to determine the vertex and edge sets. We used trip count values for each source destination pair as the edge weight. Next task was to create the graph.

GraphX provides built-in functions to extract many different properties of a graph. We used GraphX for the indegrees, outdegrees and PageRank values and also NetworkX library of Python to obtain the strongly connected components and betweenness values of our graph. We plotted the results using Python so that we could form a better understanding of the data. It also helped us determine thresholds while calculating top indegree, outdegree, pagerank and betweenness values. We also used the tool QGIS to visualize the property values with respect to physical locations in cities.

## 4.6 User Interface

To bundle our three parts; stop detection, clustering of detected stop, and graph analysis we developed a web based user interface where job files could be submitted to a running Spark cluster from a specified Amazon S3 storage. The UI was build upon leaflet[14], a interactive map library for JavaScript, and featured functionality for running the jobs, displaying the clusters and visualizing graph information such as connected component with out and in degree. To minimize executing time we implemented functionality for displaying previously ran job, instead of re executing jobs it could now display previously ran jobs. When a stop is clicked statistics about the stop appears and lines are drawn on the map according to the out and in degrees (connected travellers to/from this stop to another captured one) See figure Figure 4.4

The current UI provides following features:

- Interactive world **map** for zooming and panning

- **S3 input file for ETL**: S3 bucket and file name of the input job (currently csv file)
- **Spark master address**: URL of the spark master that should process the job
- **S3 storage URL**: The URL of the running S3 instance (could be local or public)
- **Test**: Test if job is OK and we can connect to Spark and S3
- **Run**: Run the job and display it on map (in case of a result file, just display the result)
- **Export to CSV**: Download the result output as a csv file for further use, such as not having to re-run the job again.
- **Statistics about stop**: Panel displaying information graph information about the clicked cluster, such as id, size, in/out degrees, PageRank etc.

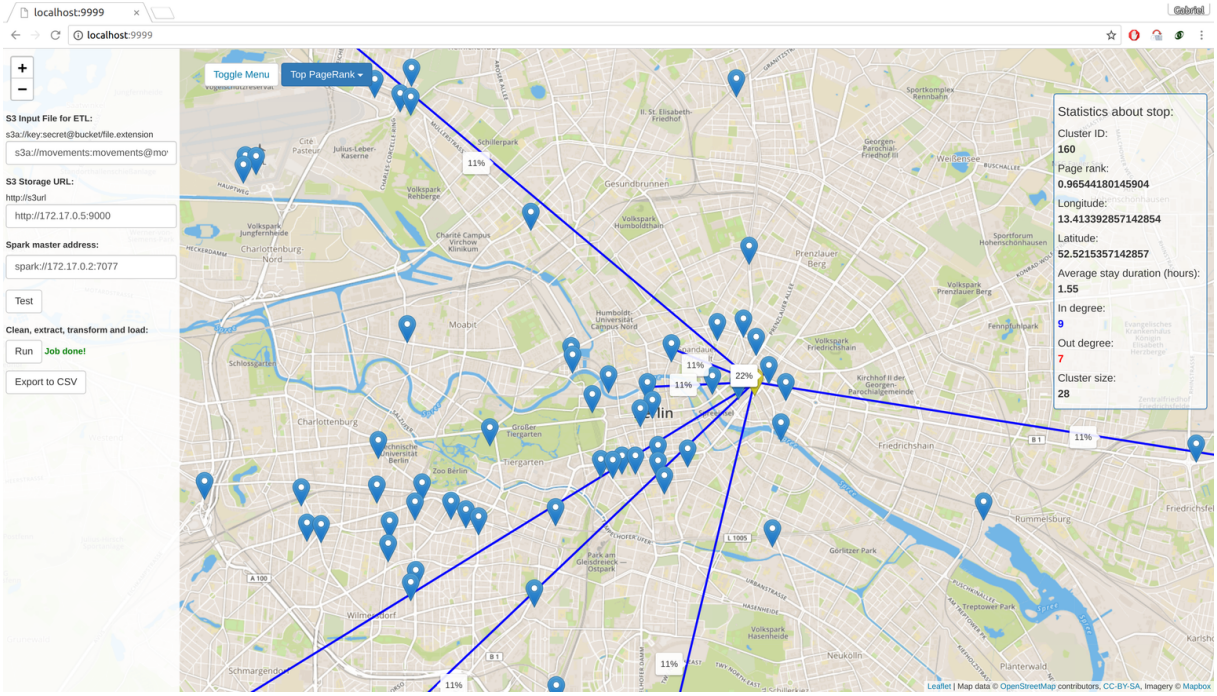


**Figure 4.4:** Screenshot of the user interface

# 5 Evaluation

In this chapter we evaluate and analyze our implementations of the Stop Detections, Clusterings and Graph processing.

## 5.1 Location data characteristics for Berlin area

We have worked with movement triggered data from mobile devices are spanned across Germany. Plotting the data gave us the picture shown in Figure 5.1.

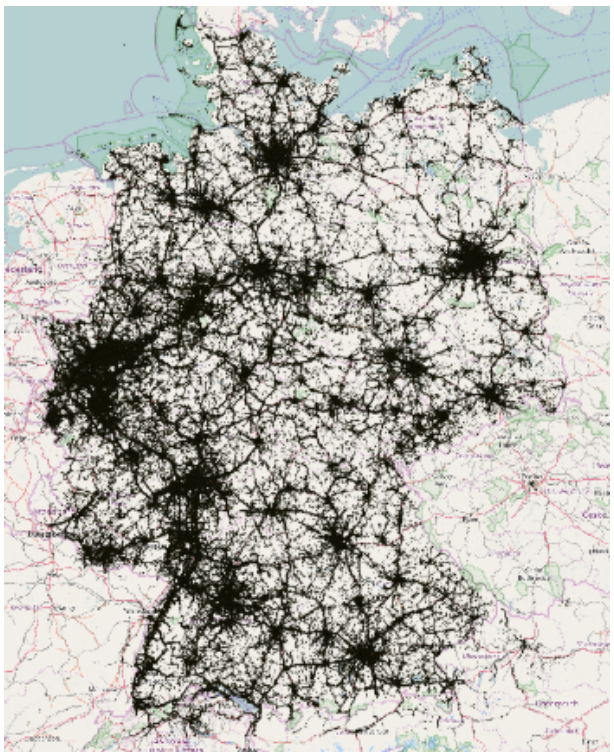

**Figure 5.1:** Visualisation of data points location in the geographical map of Germany

It turns out that most of the detected movements from one point to another are mostly gathered on the highways and inside the cities. Since our given points are updated on movement this is reasonable.

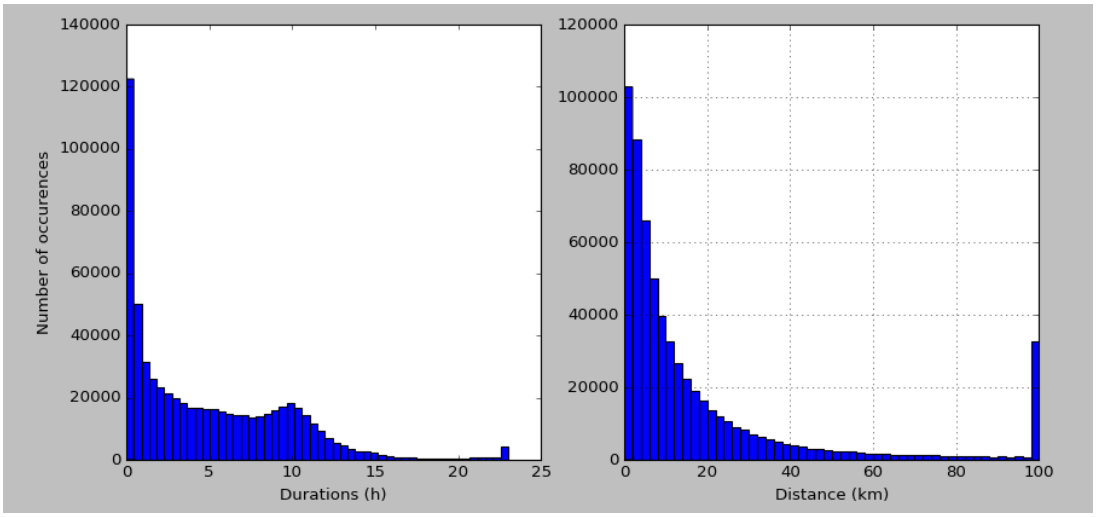


**Figure 5.2:** Histograms of duration and distances between consecutive points under and cumulatively over 100 km for the area of Germany

The given data batch has been gathered in the interval of 24h starting from 3am in the morning. It consists of over 675000 data points, belonging to 47300 unique mobile devices. Figure 5.2 shows, that for unique mobile devices, the duration between consecutive points in time are most frequent within 30 minutes and most of the data is in the interval of 1-10 h. Regarding distances, consecutive points are most frequent below 2 km and most data is in the interval 0-10 km. Big chunk of distances is also over 100 km.


**Figure 5.3:** Heatmap of data point densities within the area of Berlin

Figure 5.3 shows the heatmap of data points for Berlin area. We can see that most of the movements detected are on major train/tram/metro stations, highways and major living areas.

### 5.1.1 Data analysis

Data analysis had shown, that the users at least once visiting Berlin, had in average 17 points gathered in the period of 24h, harmonic mean of 4 points and maximum 100 points.

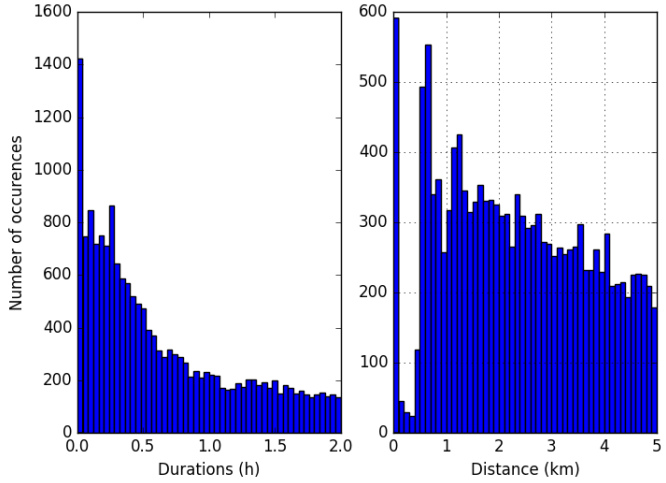


**Figure 5.4:** Histograms of durations and distances between consecutive points under 5 km and under durations of 2h for the area of Berlin

Furthermore, majority of duration between points are within intervals of 30 minutes. For distances between points, there are significant "jumps" at 500-1000m and 1100-1500m, which might mean that at these intervals, continuous points been gathered, and distances over that values might be discontinuous (ref. Figure 3.1 ). It is due to the fact that these distance intervals are most frequent and that might suggest that this is an average "update" distance for the moving mobile devices. Less frequent values might suggest that updates were obtained with lower accuracy (e.g. by presence inside the building, metro line or simply mobile device lag in obtaining its location) and are result of discontinuity between consecutive updates.

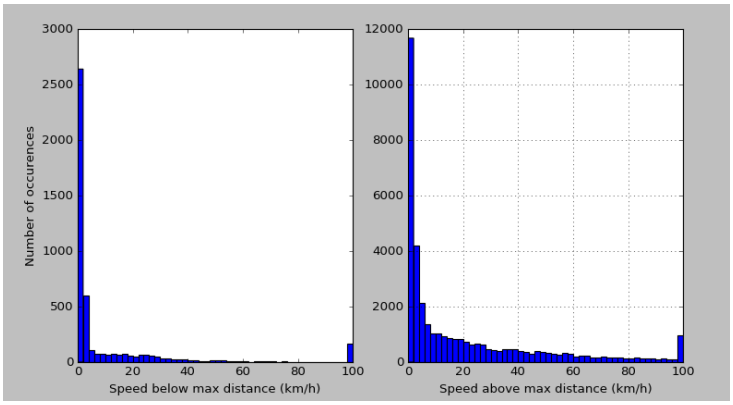


**Figure 5.5:** Histogram of speed between points with corresponding distance above or below 1.5 km

Figure 5.5 shows that considering the registered distances within range of 1500m, the wast majority of points are between 0-2 km/h, and also significant interval at 2-6 km/h. The rest of the values is sparse distributed in interval 6-100+ km/h. In case of registered point above range of 1500m, we observe that indeed most frequent occurrence is at interval of 0-4 km/h, however most are sparse distributed above 4 km/h, with most points being in interval of 4-20 km/h.

## 5.2 Pre-filtering of anomalies

Figure 5.4 shows that there is a fraction of points which duration or distance rapidly changed, thus resulting in very high speeds between the points. Points having speeds above 300 km/h and distances above 100km can be considered as flights, however the ones below 100km are anomalies. Figure 5.6 shows that number of flights in Germany has not been significant (around 100) compared to number of speed anomalies (over 25000).

Furthermore, due to the methodology of obtaining points (movement triggered), there are some minimum distances and durations at which points can be collected, and values not matching stop or travel expectation according to gathered statistics, have to be filtered and considered as duplicates.

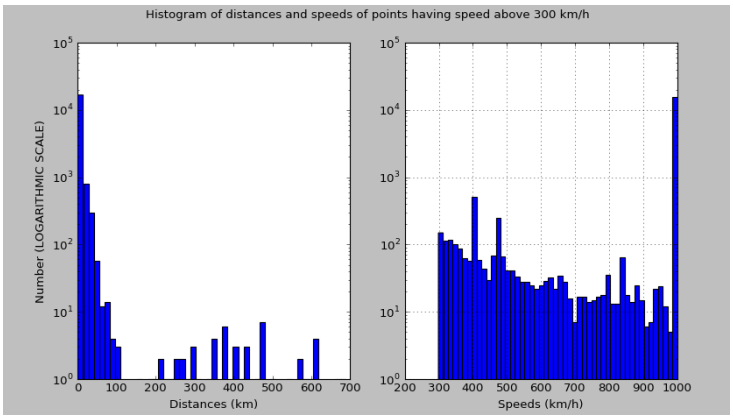


**Figure 5.6:** Speeds and distances histogram of points having speeds above certain threshold

Thus, the following predicates for filtered values have been set:

***Jumps*** Points with speed over 83 m/s [300 km/h] and at the same time distance below 100 km/h

***Duplicates*** Points with distance below 100m and at the same time duration below 100s

## 5.3 Stop detection algorithm

### 5.3.1 Mobility Index approach

Mobility Index analysis is performed in order to detect single movements or series of movements to be classified as stop or slow movement over the restricted area.

By the nature of mobility index, more values in the window, higher the mobility index is. Thus, if only one value is in the window, one can apply "stop duration" threshold, however if windows contains more values, we require higher threshold to be able to fit sum of inverses of the movements.

As an example, having window of 20 minutes and threshold of 1/(10 minutes), if we fit into window single point with duration of 15 minutes, this value will be classified as a stop. However, if we obtain 2 values, 5 and 10 minutes respectively (1/5 + 1/10), our threshold requires to be higher in order to include slow movements in the window and classify this 2

values as stop in restricted area.

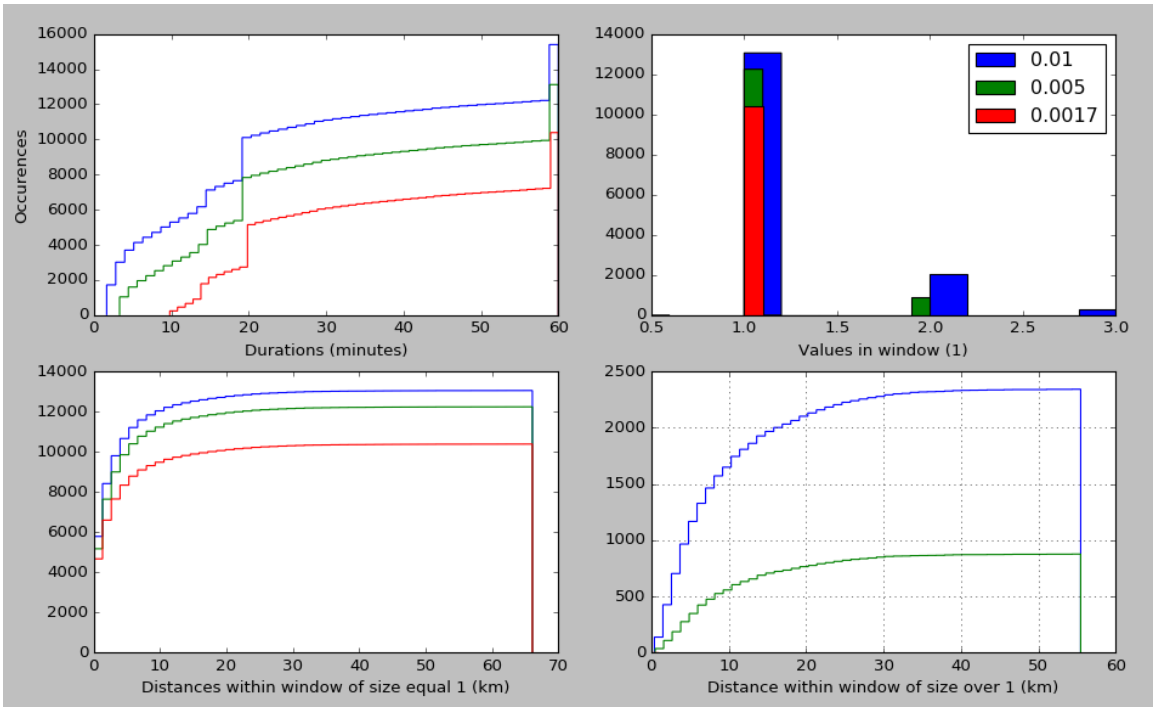


**Figure 5.7:** Analysis of detected stops for varying MobilityIndexThreshold for time window of 20 minutes. Cumulative distribution of inverted mobility index (top left). Histogram of window sizes (top right). Cumulative distributions of total distances between points in the window of size 1 (bottom left) and above 1 (bottom right).

Figure 5.7 shows the analysis of the stops detected using mobility index, using sliding time window approach.

Cumulative distribution of inverted mobility index (top left) shows that for window of 20 minutes, increasing MobilityIndexThreshold from 0.0017 (1/10 minutes) to 0.01 (1/2minutes), sum of inversions of durations (MobilityIndex) for points within the window, increases total number of stops from 10000 to 15000, where the increase (5000 points) is caused by points in the range 2-10 minutes, which is expected result increasing the threshold. The rapid jump at 20 minutes is caused by the fact that at that threshold some inverted mobility indexes were result of summing window of more then one value.

Histogram of sizes of the window (number of points within a 20 minutes window - top right) shows, that with parameter 0.0017, all the points are contained within the windows having size of 1 point. Increasing the threshold to 0.005 (1/4 minutes), it resulted in 95 percent of points being in windows of size 1 and 5 percent in windows of size 2 (2 points in the window - restricted area of stop). Doubling the threshold to 0.01 resulted in 75 percent being in single value window, and 25 percent in over 2 points per window.

The graphs in the bottom of Figure 5.7 show what are the distances covered in the windows having 1 or more points. It occurs, that in case of 1 point in the window (which means that this point is a stop), distances between points range from 0 to 60km, and this is expected. However, in case of 2 or more points per window (thus stops within a restricted area bound by points),

it occurs that only small fraction of values has total distance covered less then 2km, and major fraction of points represent long distance covered.

The above analysis shows, that mobility index cannot be effectively used to detect group of points which might be considered as a stop and only very small fraction of windows having more points then 1 is considered as a stop within a small restricted area. The rest is spanned across bigger distances. Thus, small values of mobility index window and threshold might start detecting trips over longer distance as a stop and providing erroneous results.

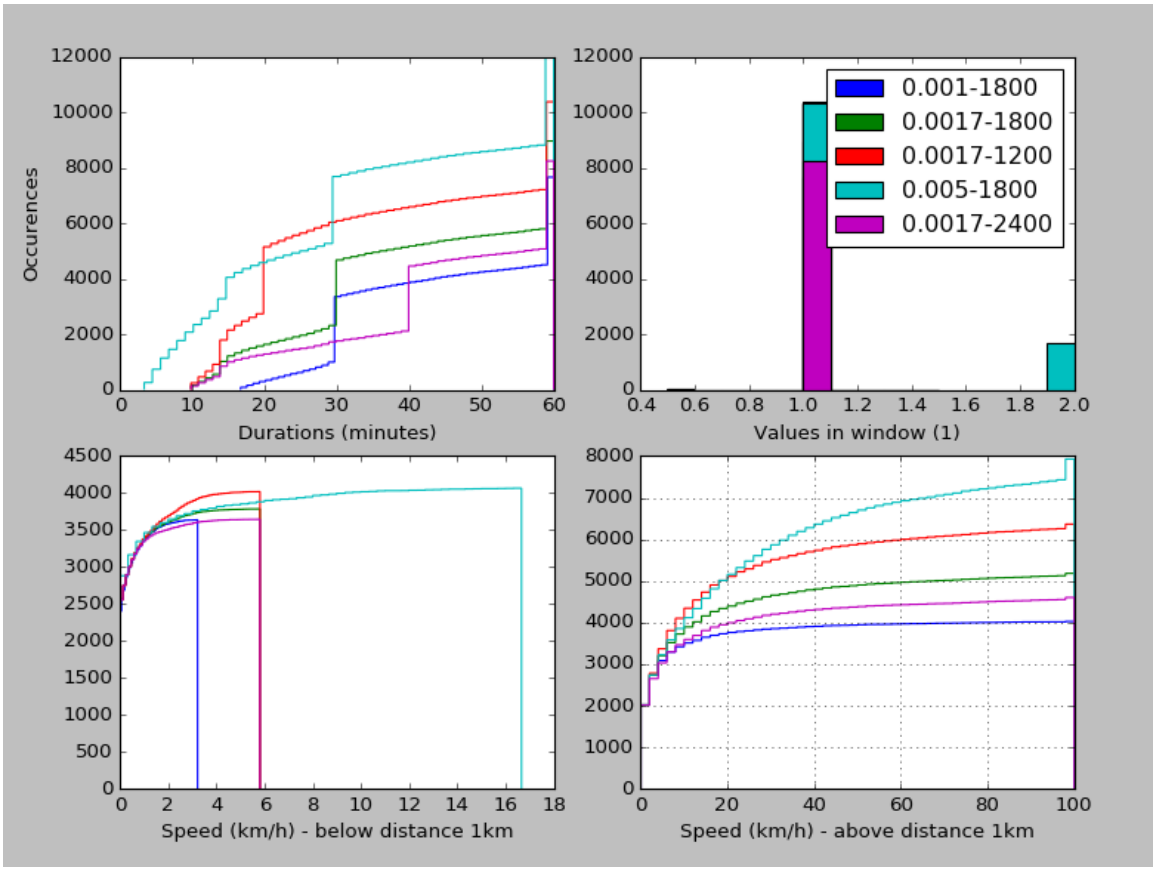


**Figure 5.8:** Analysis of detected stops for varying MobilityIndexThreshold-MobilityIndexWindow(seconds), correspondingly as in the label of measurement. Cumulative distribution of duration for detected stops (top left). Histogram of window sizes (top right). Cumulative distributions of speed between point for distance below 1km (bottom left) and above 1km (bottom right).

Figure 5.8 shows the analysis of the stops detected using mobility index threshold, using sliding time window approach, however in this case it is used to identify mobility of a user considering only last recorded point in the window to decide if the movement can be a stop considering its past mobility.

This approach tries to tackle the problem discussed in subsection 3.2.2, where it has been said that over longer distances e.g. 2-10 km between recorded points, detection using speed is difficult since stop for a short time and movement with high speed will produce very high average speed in that time period.

Figure 5.8 gives cumulative histograms considering different pair of parameters (MobilityIndexThreshold-MobilityIndexWindow) considered in producing stop locations. Cumulative distribution of duration at detected stops (top left) shows that considering the same MobilityIndexThreshold (0.0017-1200s, 0.0017-1800s, 0.0017-2400s) and increasing MobilityIndexWindow, causes less stops to be detected and higher average duration between points detected. Considering the same MobilityIndexWindow (0.001-1800s, 0.0017-1800s, 0.005-1800s) and increasing MobilityIndexThreshold, number of stops increases, and this increase is bound only to values below MobilityIndexWindow threshold. High MobilityIndexThreshold produces more stops with shorter duration of stop.

Cumulative distribution of speed for detected points show that algorithm with all tested parameters has been able to identify correctly stops over distance less then 1km and with very small speeds (long stay duration at the point). However, MobilityIndexThreshold or decreasing MobilityIndexWindow cause that average speed of detected points had higher speed over distances > 1km and these were not able to filter trips over long distances correctly. For thresholds 0.001-1800s, 0.0017-1800s and 0.0017-2400s higher speeds are being filtered and only longer stays are left, and the difference between performance of these parameters is a trade-off between an accuracy and number of detected stops.

Histogram of values in the window (top right) also confirms, that for 0.001-1800s, 0.0017-1800s and 0.0017-2400s thresholds, detected stops had window of 1 value, thus none of the windows having 2 or more values were not qualified for being a stop and thus being filtered out as high mobility trip. This behavior of algorithm allows to identify stays with simple duration threshold and at the same time filter out all multi-point trips using mobility index window.

### 5.3.2 Human Mobility approach for long/short distances

Speed/Distance analysis is performed in order to detect single movements as stop or slow movement. If the average speed between the points is below the certain threshold, it is assumed to be a stop.

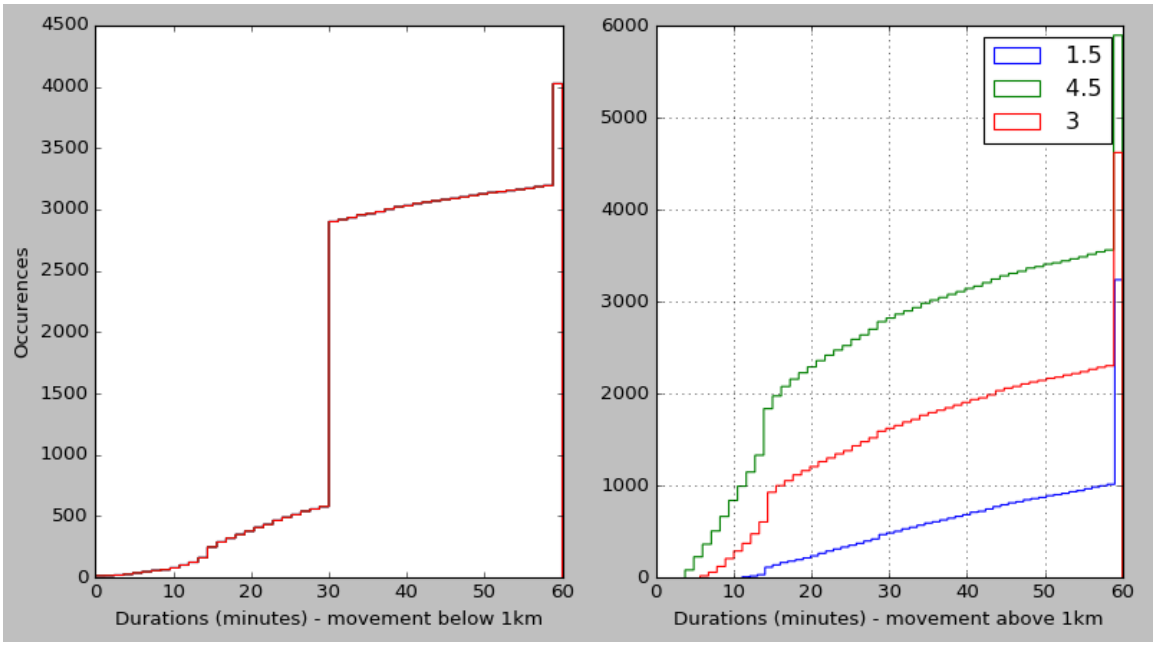


**Figure 5.9:** Speed/Distance detection with StopCertaintyMaxDistance, StopCertaintyMaxSpeed and TravelCertaintyMinSpeed thresholds. Cumulative distribution of duration for detected stops below and above distance StopCertaintyMaxDistance of 1km for different TravelCertaintyMinSpeed thresholds.

Figure 5.9 shows analysis of detected stops having TravelCertaintyMinSpeed varying and StopCertaintyMaxSpeed fixed. Increasing the speed threshold for points above 1km, we observe that number of points in interval 15 to 60+ minutes stays fixed for all parameters, however increasing threshold increases number of stops detected in interval 5-15 minutes diracticaly. It means, that with these thresholds, much more movements which has very low duration over the distance are classified as a stop and identification might be erroneous to mislead long distance trip with a stop.

Furthermore, having TravelCertaintyMinSpeed threshold of 1.5 m/s, we can see that speed threshold correctly identifies the stops, however the number of detected stops in the movements above 1km compared to mobility index approach is low (Figure 5.8).

On the other hand, for distances below 1km, speed detection has not only been able to identify longer stays at location, but also a short distance, very slow movements over the area, successfully.

## 5.4 Stop location clustering

This section describes how we evaluated our DBSCAN algorithm and the choice of the *epsilon*, and *minimum points per cluster* parameter.

### 5.4.1 DBSCAN

Running the DBSCAN with different parameters resulted in different cluster sizes and number of clusters. We focused on the target area of greater Berlin and surrounding towns. Plotting histograms and visualizing the result on a QGIS [1] map helped us in our choice of parameters. Since we used the dense target area of inner Berlin the parameter choice was easier than for example targeting whole of Germany or a larger area. However, we saw that small tweaks of the parameters had big impacts of the resulting cluster sizes and number of clusters. Below we evaluate the two parameters

### 5.4.2 Evaluation of *eps*

The *eps* parameter (epsilon, radius of the cluster) was initially chosen by looking at the graininess (accuracy) of our given data set - points only occur about each **110 meter** (longitude and latitude with five decimals accuracy, approximated to meters) therefore 110 meter is called the *minimum distance* between two points.

We took Alexanderplatz in Berlin as an example cluster (a popular train stop area in Berlin). We interpret one cluster as being one stop point of interest, therefor for this zoom level we expect Alexanderplatz to be represented as one cluster.

As noted previously, DBSCAN is very sensitive to the choice of its parameters. In the figure Figure 6.2 we run DBSCAN with *minimum distance* and *minPts* = 5, which causes expected looking clusters of popular stop areas. When choosing a greater *eps* of 220 meters we instantly get huge clusters which correspond to an abnormal amount of expected stop areas, seen in figure Figure 5.10.

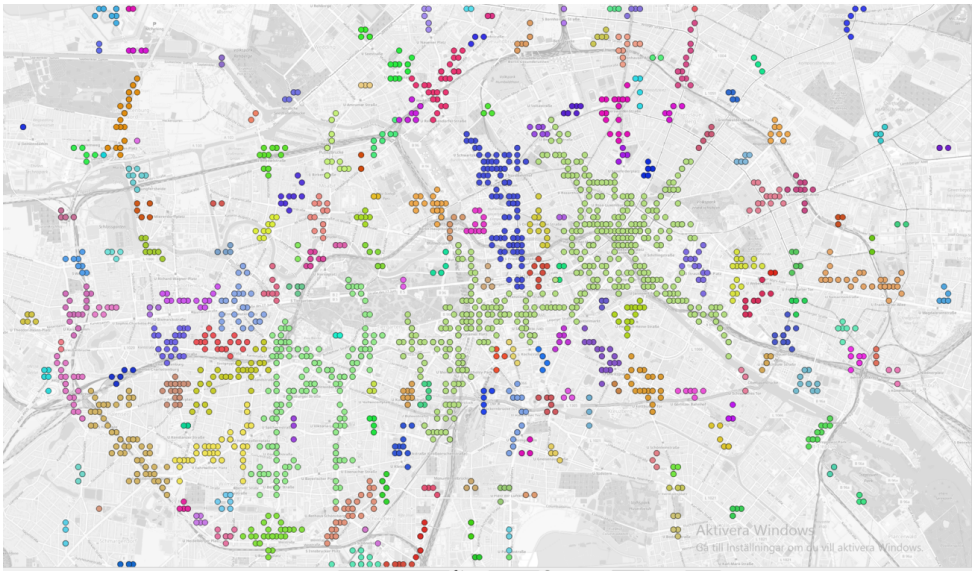


**Figure 5.10:** Example of a clusters with bad parameters, here *epsilon* = 220 meter is too large, *minPts* = 5. "Alexanderplatz" (light green) has 748 data points in it and stretches over the whole centre (Mitte) of Berlin.

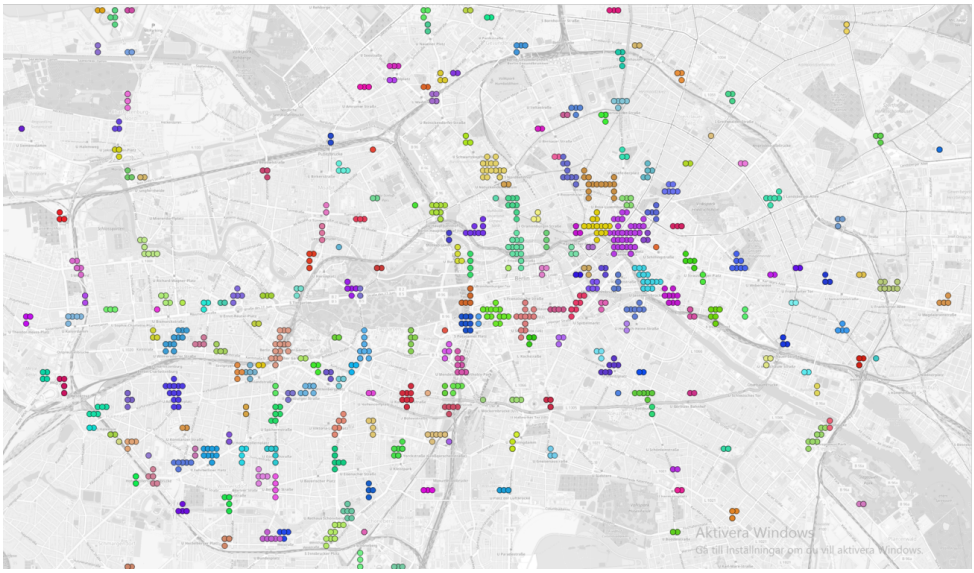

**Figure 5.11:** Clusters with good parameters *epsilon* = 110 meter, *minPts* = 5. Alexanderplatz (pink) has 85 data points in it.

We concluded on the *eps* parameter of 110 meters, the *minimum distance*, since this gave us satisfying looking clusters for inner Berlin, whilst higher value of the parameter resulted in clusters being unnaturally large and *eps* below the *accuracy distance* resulted in only single-point clusters (points are stacked at the same location).

### 5.4.3 Evaluation of *minPts*

So far we have not discussed the *minPts* parameter, minimum points per cluster. The parameter does not determine the size of the clusters but how many of them that has to be in one for it to be recognized as a cluster - which results in the number of clusters we get. Lower *minPts* result in more found clusters, while higher result in less found clusters. The choice of *minPts* was set by running DBSCAN multiple times with different values of the parameter and plotting them in histograms and on the map. We used the determined *eps* of . In Berlin there are currently 173 U-bahn stops [3]. We would expect each stop to be represented as a stop point if our stop detection algorithm works well enough and we have enough data points. Since U-bahn stations usually are in areas of interest we could approximate the number of sought after clusters to a round this number. A number of detected stops between 100 and 200 hundred seems reasonable. As seen in the histogram in figure Figure 5.12 setting *minPts* to 5 gives us a cluster size of 140. Plotted on the map of our area, figure Figure 6.2, gave us good looking result for inner Berlin.

### 5.4.4 Extending DBSCAN with areas

Our approach with using the same two fixed parameters, *eps* and *minPts*, worked well for inner Berlin. However, when zooming out and running our algorithm on a less dense area, such as greater Berlin, we encounter problem with few detected clusters. To fix this and detect more

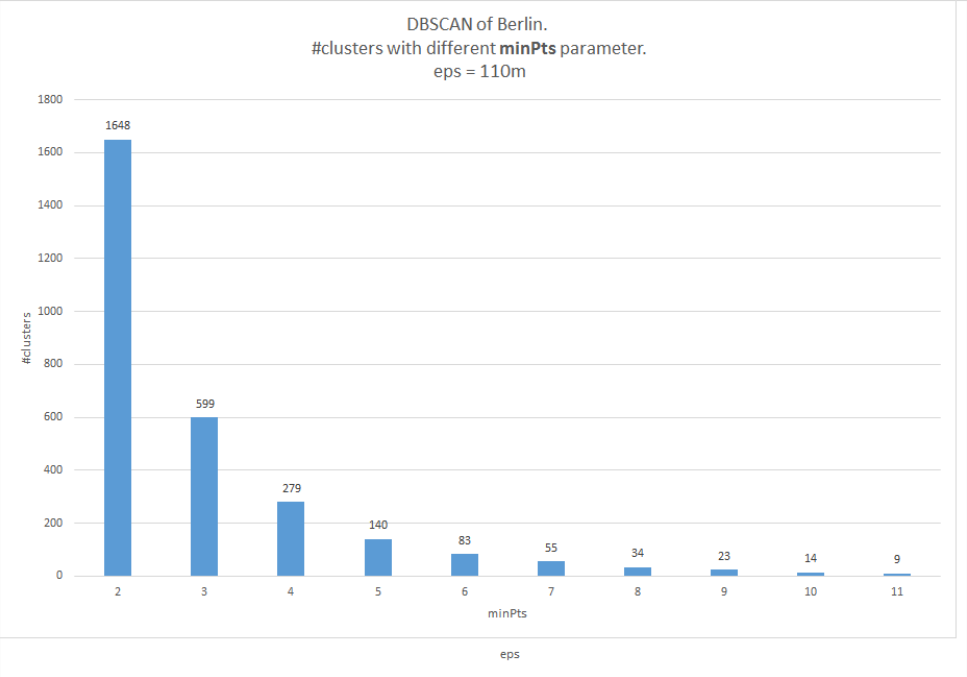


**Figure 5.12:** Histogram of number of clusters found from DBSCAN with different minPts. Higher *minPts* captures significant fewer clusters .

clusters we needed to increase the *eps* parameter, however this caused problems with **huge** clusters in central Berlin as seen in figure Figure 5.13. To solve the problem of not detecting

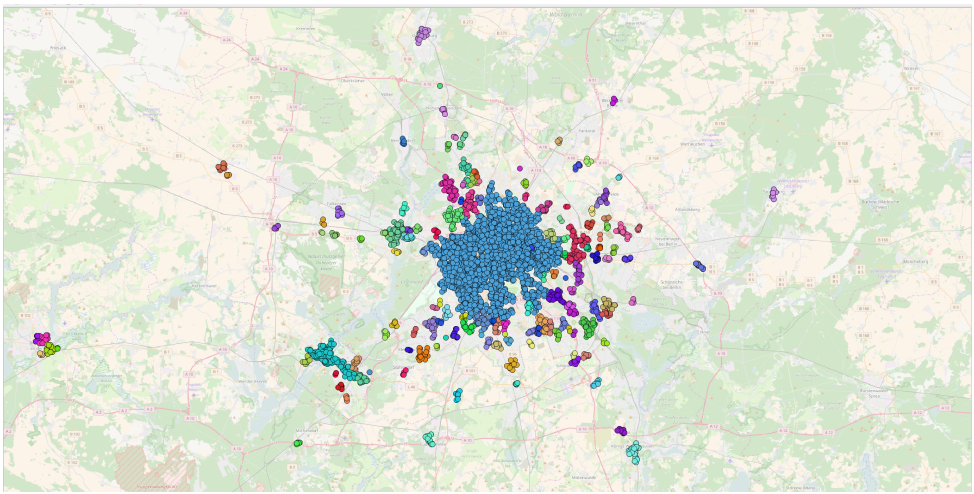

**Figure 5.13:** Huge cluster problem in central Berlin problem when trying to detect clusters outside the center.

clusters in less dense areas we implemented an area version of DBSCAN. This version ran DBSCAN on different areas of varying density with different *eps*. To demonstrate this we divided Berlin and surrounding area into three different boxes of increasing density, as seen in figure Figure 5.14. Each point was assign to its specific box and a *DBSCAN on Spark* was executed on each box and finally the result was merged. The goal was to capture more clusters in less dense

regions. The *eps* of each outer area was increased from the previous one and the histogram in

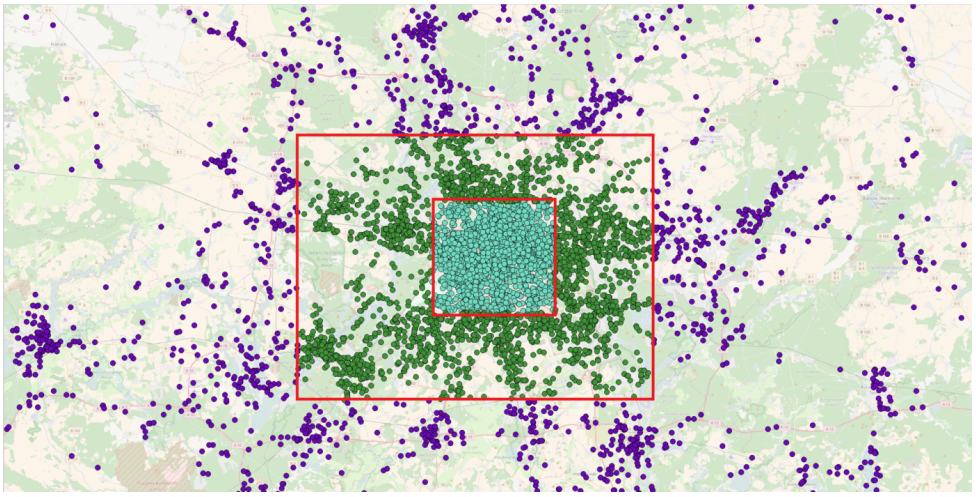

**Figure 5.14:** Berlin and surrounding area divided into boxes. On each box a separate *DBSCAN on spark* is ran with different parameters.

figure Figure 5.15 proves the feasibility of the area implementation, we capture more clusters outside of inner Berlin with the new area version. The first version runs the original DBSCAN version without using boxes, the second version divides greater Berlin into two boxes, inner and outer Berlin. Finally, the third version uses three boxes; inner, middle and outer Berlin.

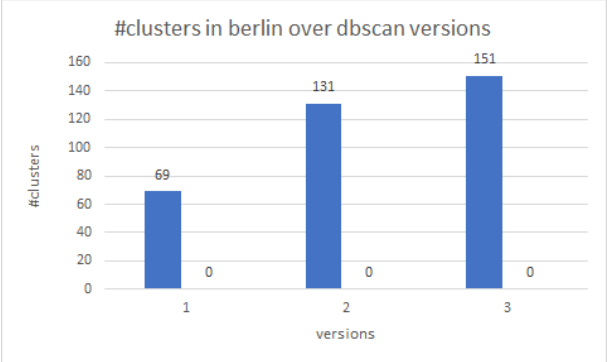


**Figure 5.15:** Histogram over found clusters running the different area DBSCAN versions.

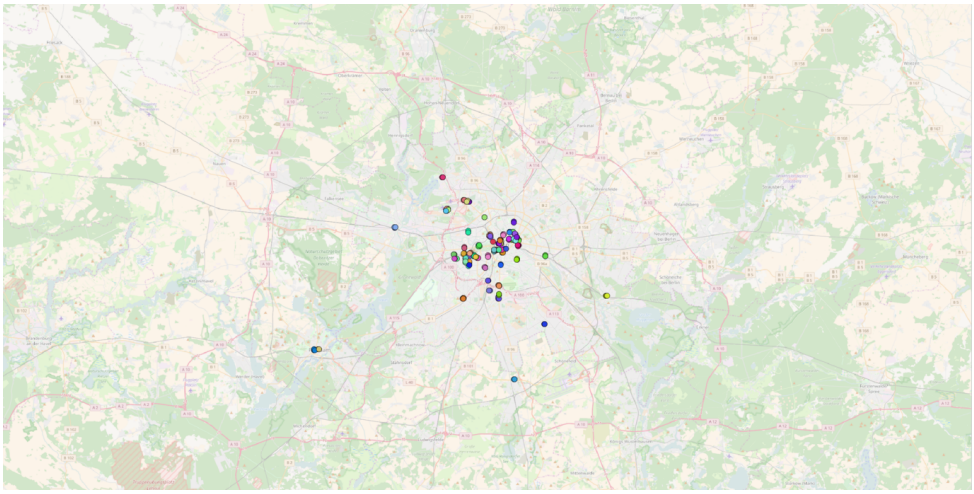

**Figure 5.16:** Version 1. Result of running original DBSCAN without boxes. *eps* = *minimum distance* on dataset

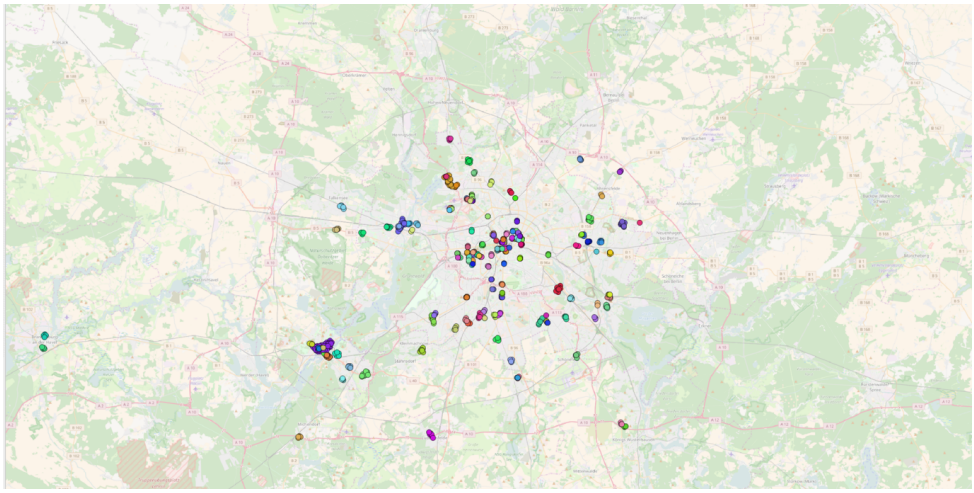

**Figure 5.17:** Version 2. Result of running DBSCAN with two boxes. *eps* = *2 * minimum distance* on dataset

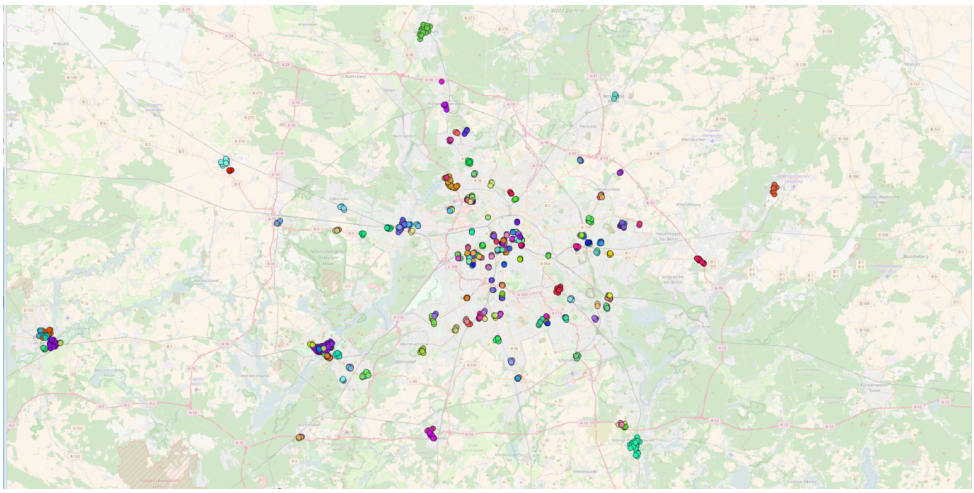

**Figure 5.18:** Version 3. Result of running the area extension of DBSCAN with three boxes. Parameter *eps* = *minimum distance* for inner area, *eps* = 3 * *minimum distance* for middle area, and *eps* = 5 * *minimum distance* for outer area. We find much more found clusters than original DBSCAN version

## 5.5 User movement graph

This section describes the evaluation of the results obtained from the user movement graph.

We choose the clusters in and around Berlin for our evaluation and worked with the data from the same. Different properties were extracted from the graph like trip count, indegree and outdegree of the vertices, PageRank and Betweenness centralities. It was our assumption that these values should help us identify areas of interest in the city as well as provide an insight into user movement patterns. We plotted the distribution of values in python to get a better idea of the range and characteristics of the values. Plotting the data also helped us identify suitable thresholds for the properties. These thresholds were needed in order to obtain clusters of importance i.e. vertices with the highest values, as well as to examine validity of the results.

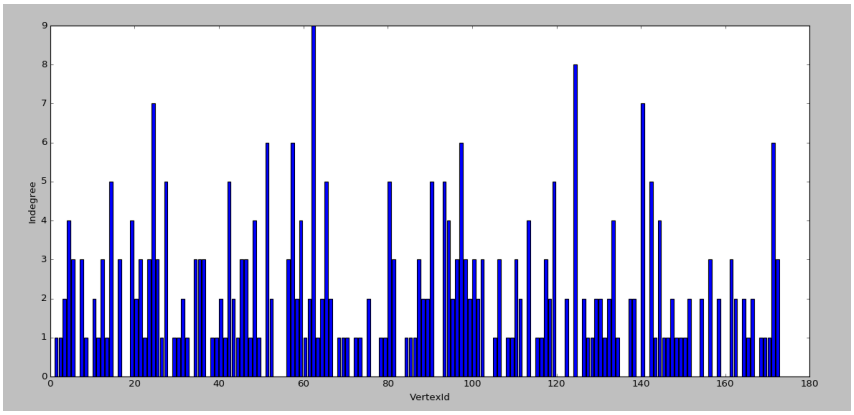


**Figure 5.19:** Plot of indegree values of vertices

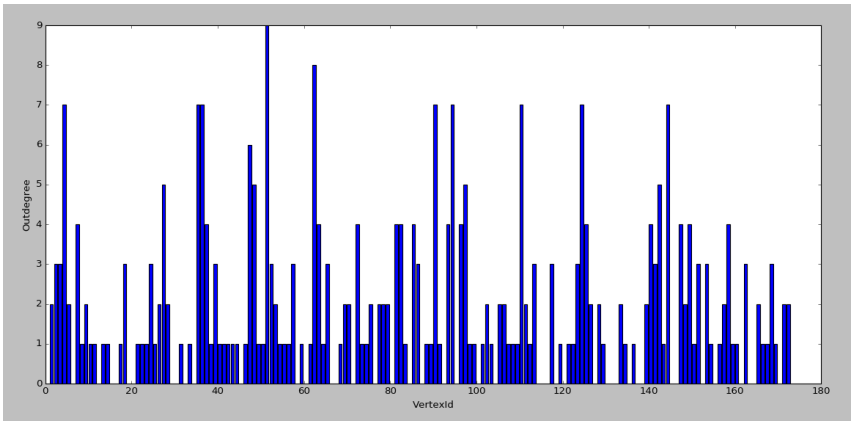


**Figure 5.20:** Plot of outdegree values of vertices

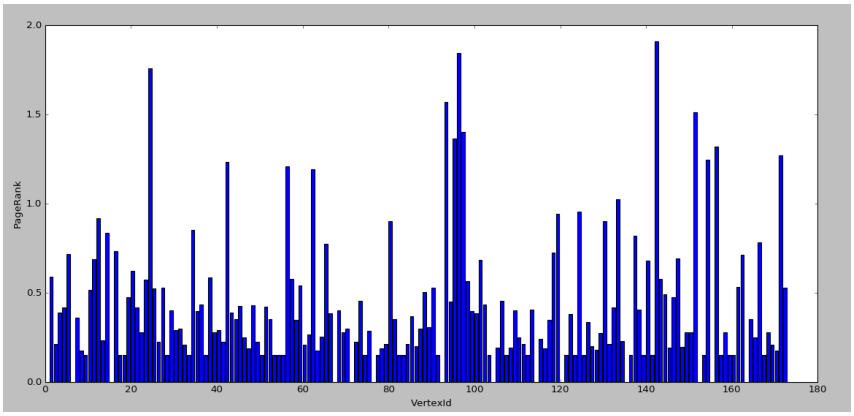


**Figure 5.21:** Plot of PageRank of the vertices

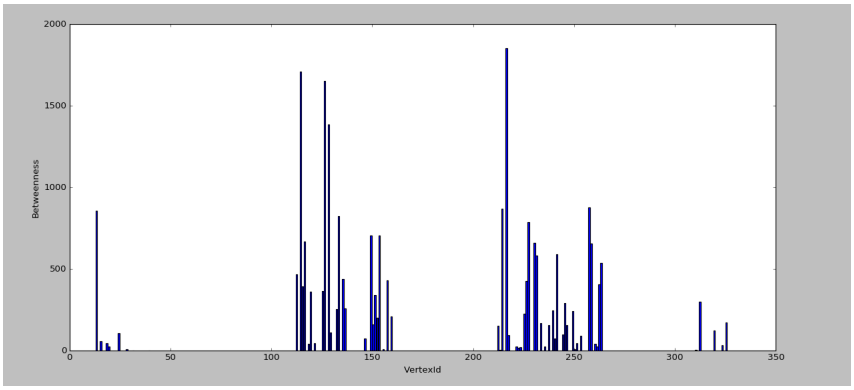


**Figure 5.22:** Plot of Betweenness of the vertices

As seen in the above plots, the values range from 0 ( or default value of 0.15 in case of PageRank) to significantly high values. For example: highest betweenness was 1854.38. Hence we decided to discard the zero values (in case of In,Out degrees and Betweenness) or default values (in case of PageRank) in order to obtain a realistic and meaningful threshold.

Table 5.1 provides the range and threshold (calculated after discarding the zero or default values) in each case. The mean of the distributions were selected as threshold.

**Table 5.1:** Range and thresholds

| Property | Range | Mean |
|---|---|---|
| Indegree | [0-9] | 2.46 |
| Outdegree | [0-9] | 2.44 |
| PageRank | [0.15 -1.909] | 0.522 |
| Betweenness | [0-1854.38] | 342.33 |

For each of the properties, vertices with values above the threshold were plotted on the map of Berlin using QGIS. Once the values were mapped to actual physical location in a city, we could understand and evaluate the results. The results are presented in the images from Figure 5.23 to Figure 5.26

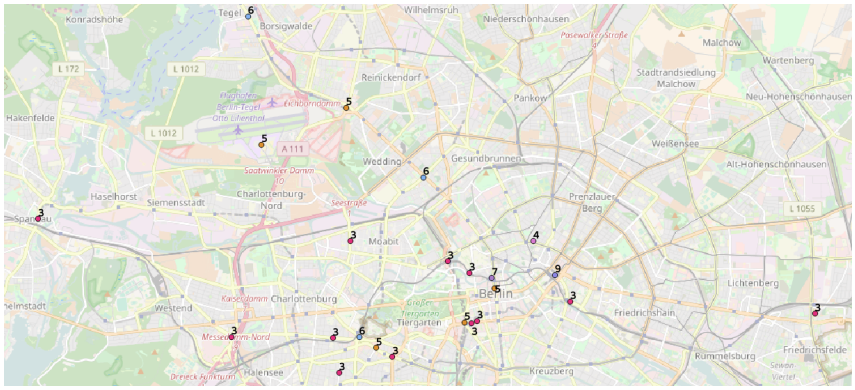


**Figure 5.23:** Indegree values of places in Berlin

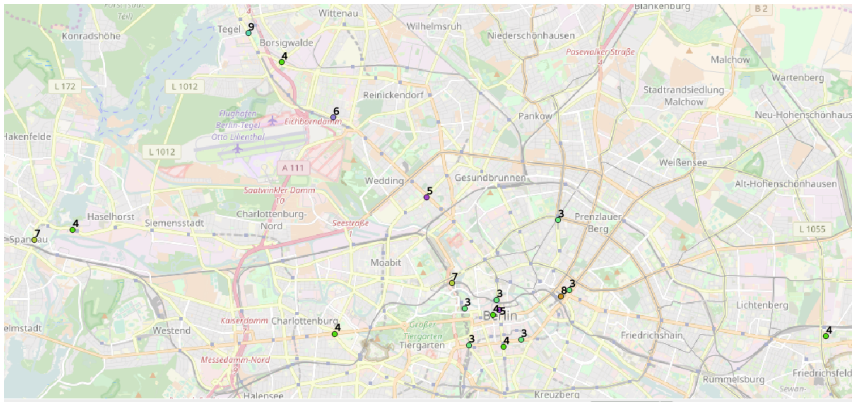


**Figure 5.24:** Outdegree values of places in Berlin

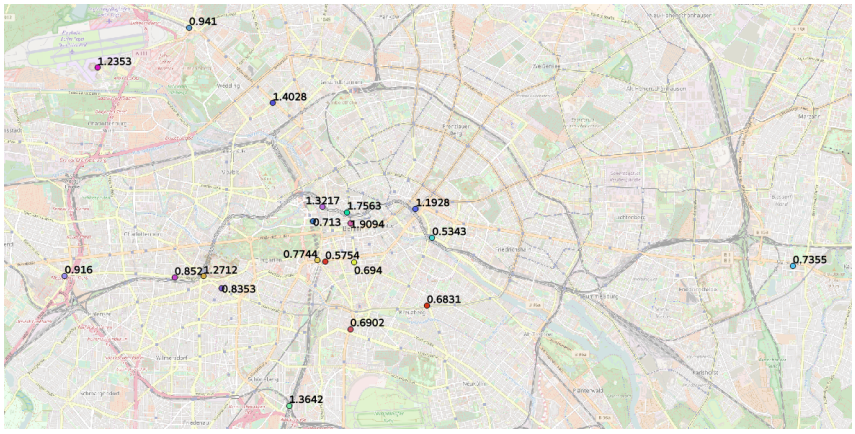


**Figure 5.25:** PageRank values of places in Berlin

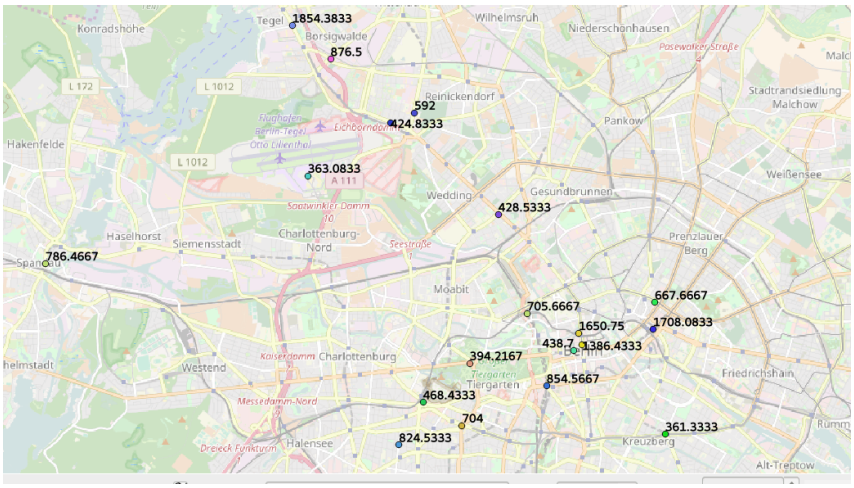


**Figure 5.26:** Betweenness values of places in Berlin

From the above images we could identify that the areas of high indegrees and outdegrees were residential areas or areas of special importance in the city like business or shopping destinations or major transport hubs. For example: Alexanderplatz in Berlin has both highest indegree (=9) and second highest outdegree (=8). The values made sense as Alexanderplatz is not only a popular spot for shopping and dining, it is also a major transport hub and is equally popular among locals residents and tourists.

In the following section, we present in-and out-degree values of a few chosen well-known places in Berlin and our explanations for the same.

**Table 5.2:** Locations and in-& out-degree values

| Index | Place | In-& Outdegree |
|---|---|---|
| 1. | Tegel | 6,9 |
| 2. | Friedrichstraße | 7,5 |
| 3. | Potsdamer Platz | 5,3 |
| 4. | Zoologischer Garten | 6,2 |
| 5. | Berlin Hauptbahnhof | 3,7 |
| 6. | Alexanderplatz | 9,8 |
| 7. | Spandau | 3,7 |
| 8. | Potsdam | 8,7 |

As per our analysis, the locations presented in Table 5.2 are important locations in Berlin for several reasons such as: transportation hub (Indexes 2,4,5,6,7), popular tourist areas(Indexes 2,3,4,5,6,7), popular shopping and dining areas(Indexes 2,3), proximity to airport(Index 1) or university( Indexes 2,4) and residential areas(Indexes 1,6,7).

Our next target was to learn about movement patterns and connectivity in the city. To achieve this we analyzed the locations with high pagerank and betweenness centralities (refer to Figure 5.25 and Figure 5.26). A list of locations chosen for our study is presented in Table 5.3 and Table 5.4

**Table 5.3:** Locations with PageRank values

| Index | Place | Betweenness |
|---|---|---|
| 1. | Tegel Airport | 1.2353 |
| 2. | S + U Friedrichstraße | 1.7563 |
| 3. | Friedrichstraße | 1.9094 |
| 4. | Berlin Südkreuz | 1.3642 |
| 5. | Zoologischer Garten | 1.2712 |
| 6. | Alexanderplatz | 1.1928 |
| 7. | Messedamm Nord | 0.916 |
| 8. | Leopoldplatz | 1.4028 |
| 9. | Potsdam | 0.9556 |

**Table 5.4:** Locations with betweenness values

| Index | Place | Betweenness |
|---|---|---|
| 1. | Tegel | 1854.383 |
| 2. | S + U Friedrichstraße | 1650.75 |
| 3. | Friedrichstraße | 1386.43 |
| 4. | Potsdamer Platz | 854.567 |
| 5. | Zoologischer Garten | 468.43 |
| 6. | Alexanderplatz | 1.1928 |
| 7. | Spandau | 786.467 |
| 8. | Potsdam | 869.05 |

We compared the locations in the Table 5.2 and Table 5.3 and noticed that some of the locations in these two tables are different. That is, despite moderate to high in- and out-degree values, the PageRank values of some nodes were not significant. We attributed this to the connectivity among the nodes and assumed that locations mentioned in theTable 5.3 were important because of they were connected with other significant locations in the city. We found that indeed these locations were important transport hubs apart from being shopping/tourist attractions or residential localities. For example: Messedamm Nord is close to central bus stop of Berlin, Berlin Südkreuz is a major train station and both these places have good connectivity with the city and other towns nearby. Thus we the results can be inferred as valid.

Next, we analyzed the locations with high betweenness values with the assumption that this result set should provide us with a list of nodes that enjoy a place of influence in the network due to their relative position with respect to other nodes. On assessing the results in Table 5.4, we found that our assumptions were true as these nodes are part of many important transport routes and serve as change-over points between different modes of transport. We have already sighted examples of Alexanderplatz and Zoologischer Garten in this respect. Equally important examples were that of the cities of Spandau and Potsdam. From the results we can conclude that we have been able to identify key locations in the city that denote the choice of routes by users.

To further validate the inferences drawn from the centrality values, we checked the strongly connected components of the graph. Not surprisingly the nodes appearing in the above tables featured in those as well. As seen in Figure 5.27, the strongly connected component comprises of the key locations were already identified by us.

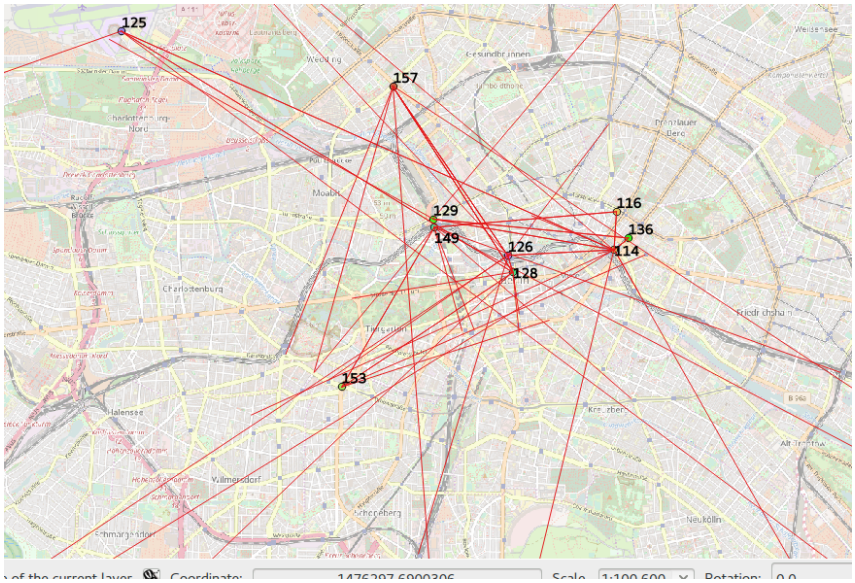


**Figure 5.27:** Largest strongly connected component in Berlin

After identifying popular locations and connectivity choices of users, our next focus was to identify key routes in the city. As already described in subsection 3.4.2, trip count values indicate the number of edges between each pair of nodes. We used trip counts of node-pairs to identify the nodes with highest traffic.

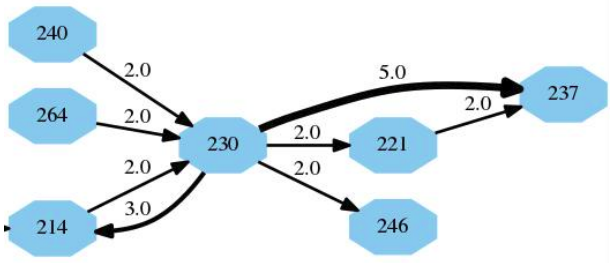


**Figure 5.28:** Edges with the maximum trip counts

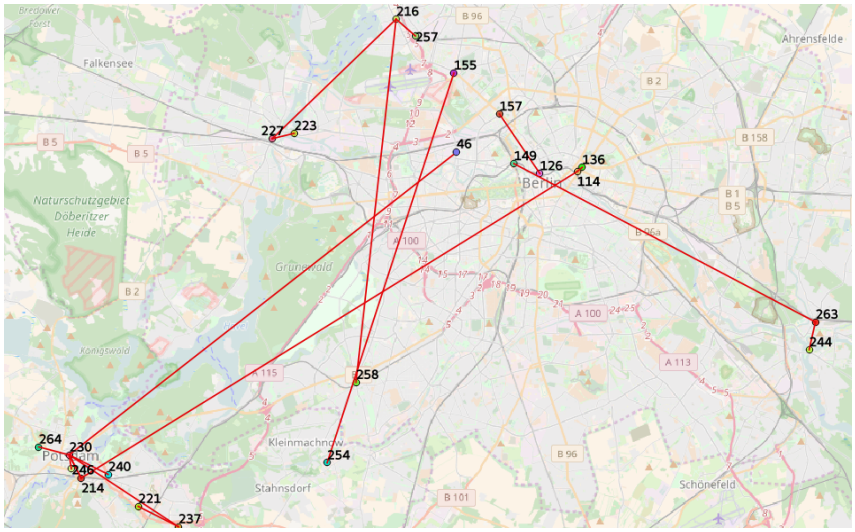


**Figure 5.29:** Popular routes in the city

From Figure 5.29 we can observe that the edges with high traffic correspond to routes between key locations in Berlin and neighboring cities. This result matches the results we already obtained from degree and centrality values and confirms our initial assumptions.

Thus we can conclude that by representing user movement data in the form of a graph and using its properties, we could determine key locations, key routes and connectivity of the city.

# 6 Conclusions

This chapter will summarize the main findings of the project implementation [16].

## 6.1 Stop detection algorithm

In the final implementation (section 4.3), the following parameters were used:

- durationsSlidingWindowSize - 1800 s
- mobilityIndexThreshold - 0.0017 1/s
- distanceThreshold - 1000 m
- speedThreshold - 1.4 m/s

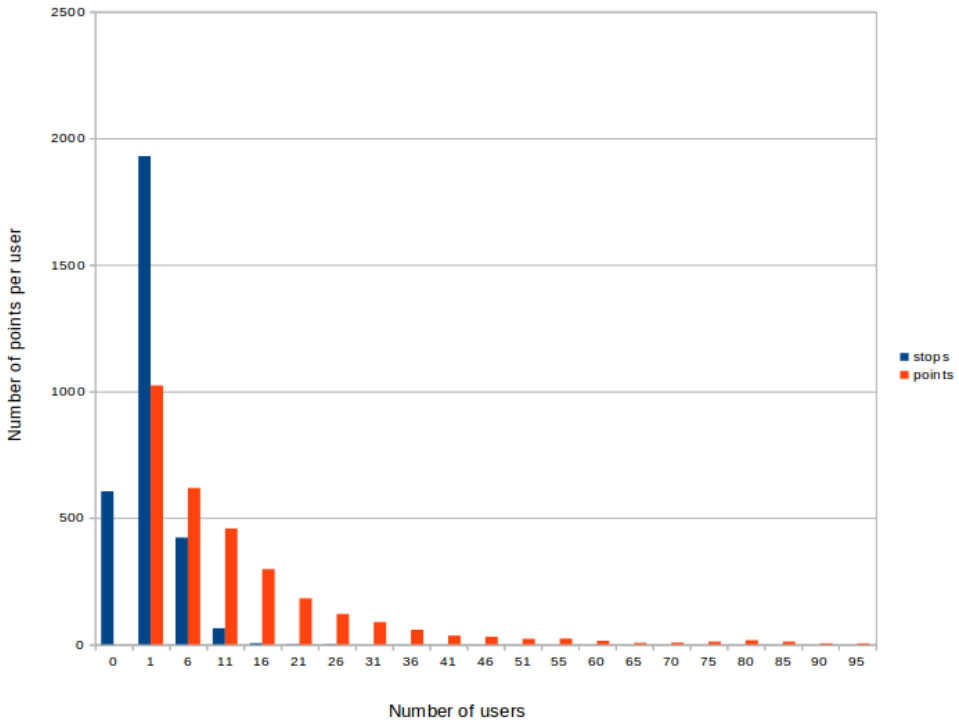


**Figure 6.1:** Histogram of number of users having corresponding number of detected stops in the area of Berlin

Figure 6.1 shows, that distribution of recorded points per user follows heavy tailed distribution, and most of the users produced between 1-10 points during the observation period of 24h, within the Berlin area (52.0102, 11.2830, 53.0214, 13.9389). The stop detection algorithm imple-

mentation (section 4.3) has been able to identify 9022 stops for 2422 unique users, whereas 605 users had no stop at all. Figure 6.1 additionally shows, that most of the users had between 1-6 stops, and characteristic is also heavy-tailed.

**Table 6.1:** Comparison of two analyzed stop detection algorithms for proactive localization

| Case | Mobility Index | Speed/Distance |
|---|---|---|
| Multiple, slow movements over small area or distance | Possible, but adjusted parameters will cause results to be erroneous for different detections | Very good |
| Single record over small area | Good, but only for longer stays | Very good |
| Single record over big area | Very good, but only in case of adjusted parameters which might be erroneous at short distances | Possible, but might be erroneous and stops might be mislead with travel due to high speed averages in these cases |

Table 6.1 shows comparison of two approaches to stop detection evaluated in section 5.3. It occurs, that for short distances it is more efficient to use speed/distance algorithm to be able not only capture longer stays at short distance, but also very slow movements in the limited area. For longer distances it is preferable to use mobility index to be able distinguishing travels from stops better using mobility index, which preserves mobility history in the time/duration context for the certain past time window. Using duration based mobility index approach for longer distances allows better stop detection than possibly erroneous speed averaging approach.

## 6.2 Clustering of detected stops

In this project we used the technique of clustering for detecting popular stop areas. We found that using the density-based clustering algorithm DBSCAN worked well. To be able to run the algorithm for big data, we extended a distributed and parallel DBSCAN implementation on Spark based on MapReduce, called *DBSCAN on spark*.

Our data was generated from all over Germany, but we started off focusing on the Berlin area. Running the clustering of stop detection gave us clusters in expected places, such as metro stations and plazas. More points where captured at popular "stop points", such as Alexanderplatz (popular plaza and train stop), and Hauphbahnhof (main train station).

However, we found out that since DBSCAN is highly parameter sensitive, a configuration for a specific area might not give satisfying results for another less, or more, dense area. Running with the parameters on inner Berlin as outer Berlin and rest of Germany yielded few or no clusters (due to less points in close proximity to each other).

To address this we implemented and analyzed an "area version" of DBSCAN on spark. The implementation ran multiple parallel iterations of *DBSCAN on Spark*, each on a specified area of various density with different *epsilon* and *minPts* parameters, and finally merging the results. This addressed our problems of not finding sufficient amount of clusters in less dense areas,

which was proven by visualizing our findings in the tool QGIS, along with plotting graphs and histograms.

**Table 6.2:** Final DBScan parameters used

| DBSCAN area version | eps | minPts |
|---|---|---|
| 1 | 0.001 | 5 |
| 2 | 0.003 | 5 |
| 3 | 0.005 | 5 |

We conclude that the best parameters we got for our DBSCAN implementation were the one shown at Table 6.2. The reason for running with the same *minPts* parameters with bigger *eps* were lack of data points in lesser dense areas, resulting in the need of higher *eps* but not *minPts* for larger areas. Important to remember is that the parameter choice depends on the granuality and application, if we are interested in capturing smaller areas, such as U-bahn/S-bahn stops, we need different parameters than clustering bigger areas, such as big plazas or towns.

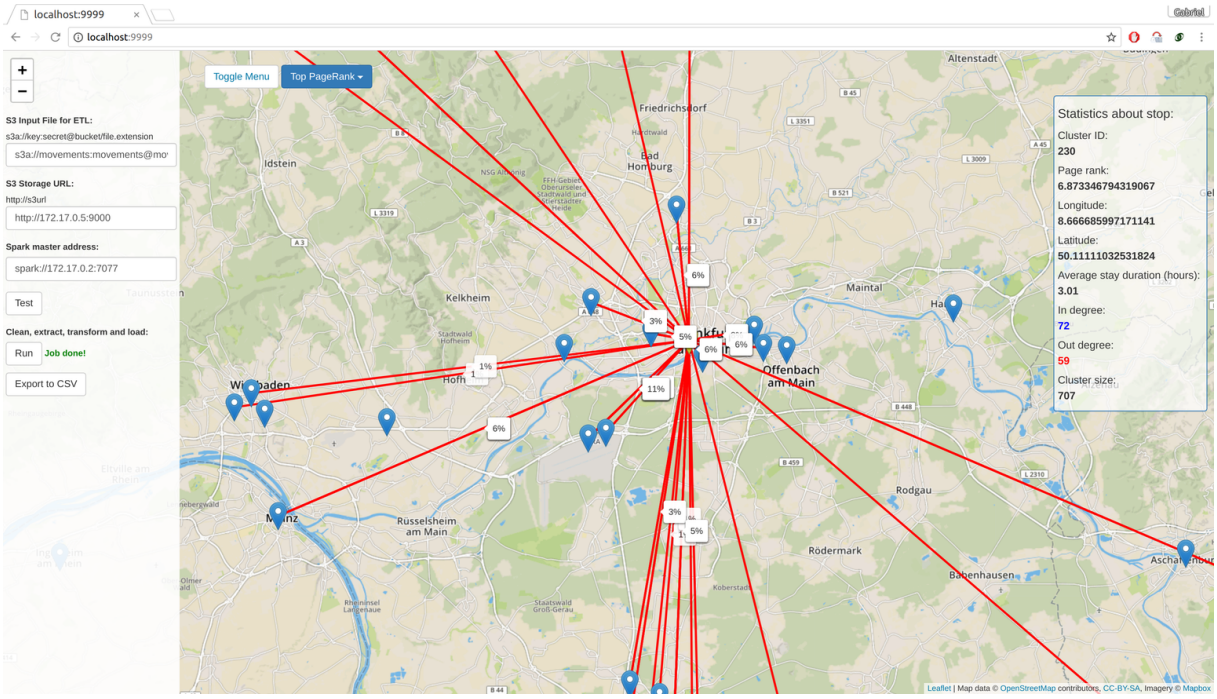


**Figure 6.2:** User interface showing clusters and connection from central Frankfurt.

## 6.3 Graph analysis

The graph analysis phase was based on the output of clustering phase. Our assumptions were that properties of the graph should help us identify important areas and routes in a city. We chose to analyze clusters in and around Berlin in order to verify our assumptions. Once the graph was created and compared with map of Berlin, we could conclude that the stop algorithm and clustering algorithm worked correctly, as nodes of the graph matched with many

well-known areas in the city.

Now, based on properties extracted from the graph, we tried to understand nature of movements of users and locations of interest. Using properties like in- and out-degrees we could identify key locations that were important as shopping or business areas, residential areas, transport hubs as well due to proximity with airport or Universities.

Furthermore, by analyzing the centrality values like betweenness and pagerank we could identify influential nodes that represented key junction in the city transport network. Due to their connectivity and position in the network, they strongly influenced routing decision of users. It was least surprising that some of the most well traveled routes passed through these nodes.

We could also correctly identify location with high inter-connectivity and verify the results using strongly connected components of the graph. Last but not the least, using trip count measures of the OD matrix, we were able to identify popular routes in Berlin and nearby cities of Spandau and Potsdam.

The results extracted from each of the properties were in sync with each other and also corroborated our assumptions. In this way we can conclude that the results of graph analysis phase could successfully identify key locations and routes in the city and provide a better understanding of user movement patterns.

# 7 Future work

We identified few bullet points which might be considered in the future implementation

- The user interface in the current version just runs job including stop detection, clustering and graph analysis together, with predefined parameters (hard-coded). However, it might be interesting to provide more sub-tabs which will allow to test each of algorithms with changing parameters.
- DBScan implementation currently uses hard-coded areas to run separate algorithms on. It might be valuable to dynamically allocate area (e.g. from database) according to the density of the population and adjust eps and minPoints parameters accordingly.
- It might valuable to provide additional section which will work on ETL basis. Thus, each new chunk of data extracts data from file and db, augments the results and loads back to the database, making the results and movements graph more accurate. This could result in e.g. merger of 2 nearby clusters (e.g. Tegel airport in Berlin) or underlying shapes of two neighboring clusters (e.g. Alexanderplatz in Berlin). In this case, user accessing interface triggers low-latency Spark job or queries the database.
- In our current implementation we focused on direct neighbors of nodes, i.e. nodes that were connected by a single hop. However, this can be extended to check second degree neighbors or more. Such results could provide further insight into the connectivity and movement patterns.
- Since we worked with data collected for a duration of 24 hours, we did not analyze our graph for time based variation in movement patterns. It would be interesting to see how the values and patterns vary based on time of day, between weekdays and weekends, over seasons or during any particular events.

# Bibliography


[1] *A Free and Open Source Geographic Information System*. URL: http://www.qgis.org/en/site/ (visited on 07/08/2017).

[2] *Apache Spark*. URL: https://spark.apache.org/ (visited on 06/18/2017).

[3] *Architecture*. URL: https://www.bvg.de/en/Company/Profile/Architecture (visited on 07/08/2017).

[4] LAURA ASHER. "Most older pedestrians are unable to cross the road in time: a cross-sectional study". In: *Oxford University Press* (2012). DOI: 10.1093/ageing/afs076. URL: https://academic.oup.com/ageing/article/41/5/690/47318/Most-older-pedestrians-are-unable-to-cross-the.

[5] Tengfei Bao. "Mining Significant Places from Cell ID Trajectories: A Geo-grid based Approach". In: *IEEE 13th International Conference on Mobile Data Management* (2012).

[6] Joschka Bischoff, Michal Maciejewski, and Alexander Sohr. *Analysis of Berlin's taxi services by exploring GPS traces*. IEEE, 2015.

[7] Aleksander Buczkowski. "Location-Based Services". In: *Geoawesomeness* (2012). URL: http://geoawesomeness.com/knowledge-base/location-based-services/.

[8] Irving Cordova. *DBSCAN on Spark*. URL: https://github.com/irvingc/dbscan-on-spark (visited on 06/18/2017).

[9] Irving Cordova. *Visualizing DBSCAN on Spark*. URL: http://www.irvingc.com/visualizing-dbscan (visited on 06/18/2017).

[10] Thad Starner Daniel Ashbrookm. "Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users". In: *Personal and Ubiquitous Computing* (2003). URL: https://link.springer.com/article/10.1007/s00779-003-0240-0.

[11] Adriano Moreira Filipe Meneses. "Using GSM CellID Positioning for Place Discovering". In: *Pervasive Health Conference and Workshops* (2006). DOI: 10.1109/PCTHEALTH.2006.361692. URL: http://ieeexplore.ieee.org/abstract/document/4205183/.

[12] Minsu Shin Injong Rhee. "On the Levy-walk Nature of Human Mobility". In: *IEEE/ACM Transactions on Networking (TON)* (2011). DOI: 10.1109/TNET.2011.2120618. URL: http://dl.acm.org/citation.cfm?id=2042974.

[13] Julia F. Kost. "GPS Everywhere – Location-Based Services". In: *Move-Forward* (2015). URL: https://www.move-forward.com/gps-everywhere-location-based-services/.

[14] *Leaflet - an open-source JavaScript library for mobile-friendly interactive maps*. URL: http://leafletjs.com/.

[15] Jingtao Ma et al. "Deriving operational origin-destination matrices from large scale mobile phone data". In: *International Journal of Transportation Science and Technology* 2.3 (2013), pp. 183–204.

[16] *Macromovements repository*. URL: https://gitlab.tubit.tu-berlin.de/gavil/MacroMovements.

[17] *Macroscopic Movements Algorithms Prototypes*. URL: https://github.com/mrow4a/macroscopic-movements-algorithm-prototypes.

[18] Aude Marzuoli, Vlad Popescu, and Eric Feron. "Two perspectives on graph-based traffic flow management". In: *First SESAR Innovation Days* (2011).

[19] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[20] Yong Yang. "Walking Distance by Trip Purpose and Population Subgroups". In: *PMC* (2013). DOI: 10.1016/j.amepre.2012.03.015. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3377942/pdf/nihms369465.pdf.

[21] Bing Zhu et al. *Urban population migration pattern mining based on taxi trajectories*. 2013.