Chapter 4
Smittestopp analytics: Analysis of position data

Vajira Thambawita, Steven A. Hicks, Ewan Jaouen, Pål Halvorsen, and Michael A. Riegler

Abstract Contact tracing applications generally rely on Bluetooth data. This type of data works well to determine whether a contact occurred (smartphones were close to each other) but cannot offer the contextual information GPS data can offer. Did the contact happen on a bus? In a building? And of which type? Are some places recurrent contact locations? By answering such questions, GPS data can help develop more accurate and better-informed contact tracing applications. This chapter describes the ideas and approaches implemented for GPS data within the Smittestopp contact tracing application. We will present the pipeline used and the contribution of GPS data for contextual information, using inferred transport modes and surrounding POIs, showcasing the opportunities in the use of GPS information. Finally, we discuss ethical and privacy considerations, as well as some lessons learned.

V. Thambawita
Department of Holistic Systems, Simula Metropolitan Center for Digital Engineering,
e-mail: vajira@simula.no

S. A. Hicks
Department of Holistic Systems, Simula Metropolitan Center for Digital Engineering,
e-mail: steven@simula.no

E. Jaouen
Department of Machine Intelligence, Simula Metropolitan Center for Digital Engineering,
e-mail: ewan@simula.no

P. Halvorsen
Department of Holistic Systems, Simula Metropolitan Center for Digital Engineering,
Department of Computer Science, Oslo Metropolitan University,
Department of Informatics, University of Oslo
e-mail: paalh@simula.no

M. Riegler
Department of Holistic Systems, Simula Metropolitan Center for Digital Engineering,
Department of Computer Science, UiT The Arctic University of Norway
e-mail: michael@simula.no

A. Elmokashfi et al. (eds.), *Smittestopp − A Case Study on Digital Contact Tracing*,
Simula SpringerBriefs on Computing 11, https://doi.org/10.1007/978-3-031-05466-2_4

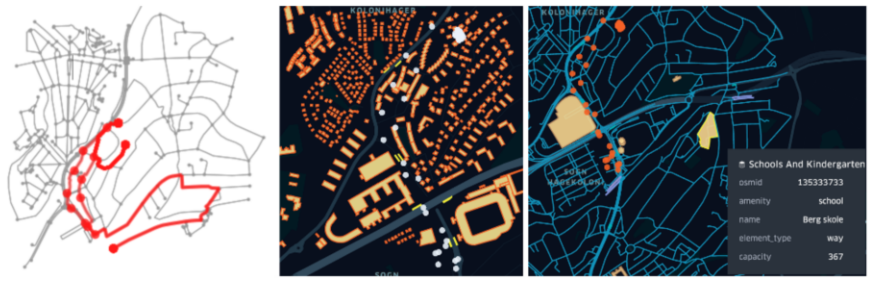


Fig. 4.1: Examples of map prototypes used during the development process. We can see different metadata plotted, such as pathways, buildings, and GPS data points. To create the maps, we used the Kepler library (kepler.gl). Maps created for this pipeline were only used for development and not provided in the final build, due to resource consumption and privacy concerns.

4.1 Introduction

One of the information sources that Smittestopp was supposed to rely on for contact tracing was the position data collected from people's smartphones. Such position data can help determine if people crossed paths and can provide information about the places they visited, such as if someone was in a store or used public transport. Position data, often also referred to as Global Positioning System (GPS) data, usually consist of the position as a longitude and a latitude, as well as a timestamp. Suppose the position data are collected via smartphones. In that case, one can also obtain additional information, such as speed (from one GPS data point to the next), altitude, and accuracy (an approximation of how accurate the obtained position data might be). On the other hand, some GPS data sets contain manually collected data, such as transport modes. For example, the GeoLife GPS Trajectories [11] data set contains annotated trajectories, with labels for the transportation mode (bus, train, walking, and car), latitude, longitude, altitude, and timestamp.

To render raw position data usable for tasks such as contact tracing, pre-processing and pre-analysis are required. First, one needs to extract trajectories from the raw GPS data. These trajectories can be used to find possible intersections and travel paths, among other things. Trajectories and GPS point data are essential for map matching, that is, to connect the GPS information to information obtained from a map. For example, was a person at a bus stop waiting for a bus, was a person in a store, did people walk together, and so on. In addition to map matching and trajectories, identifying a contact also requires analysing which mode of transportation was used (walking, driving, biking, public transportation, etc.) [7]. Moreover, necessary but straightforward pre-processing methods such as bounding box and polygon creation based on GPS points are required. One final challenge that needs to be addressed is that the amount of GPS data is usually massive, with huge numbers of data points (also depending on the granularity of time). Thus, one needs efficient methods to

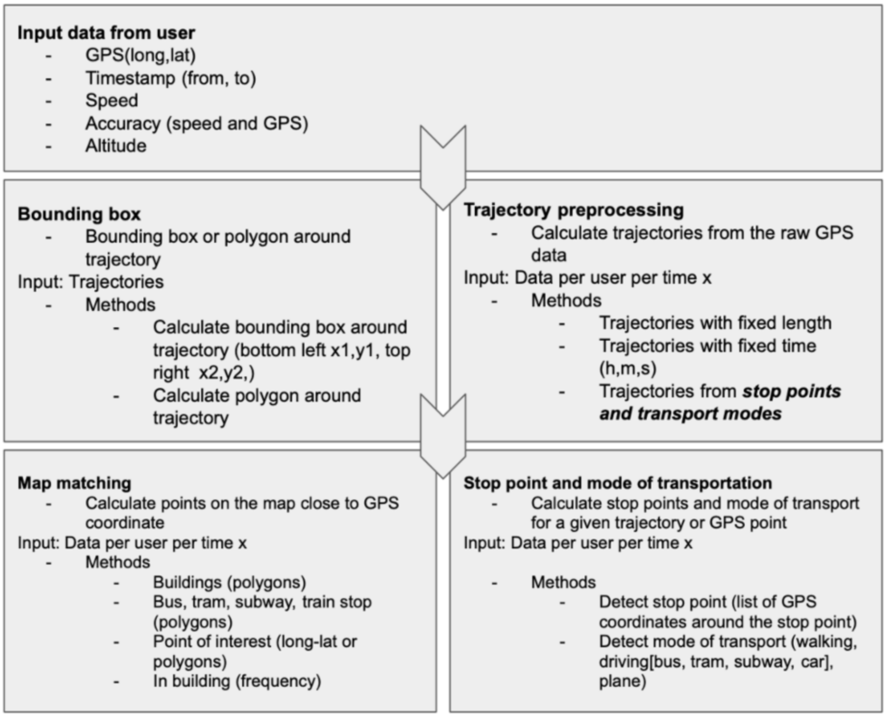

Input data from user
- GPS(long,lat)
- Timestamp (from, to)
- Speed
- Accuracy (speed and GPS)
- Altitude
Bounding box
- Bounding box or polygon around trajectory
Input: Trajectories
- Methods
- Calculate bounding box around trajectory (bottom left x1,y1, top right x2,y2,)
- Calculate polygon around trajectory
Trajectory preprocessing
- Calculate trajectories from the raw GPS data
Input: Data per user per time x
- Methods
- Trajectories with fixed length
- Trajectories with fixed time (h,m,s)
- Trajectories from stop points and transport modes
Map matching
- Calculate points on the map close to GPS coordinate
Input: Data per user per time x
- Methods
- Buildings (polygons)
- Bus, tram, subway, train stop (polygons)
- Point of interest (long-lat or polygons)
- In building (frequency)
Stop point and mode of transportation
- Calculate stop points and mode of transport for a given trajectory or GPS point
Input: Data per user per time x
- Methods
- Detect stop point (list of GPS coordinates around the stop point)
- Detect mode of transport (walking, driving[bus, tram, subway, car], plane)

Fig. 4.2: Overview of Smittestopp's GPS analysis pipeline. Raw GPS data are first transformed into trajectories and then combined with metadata from public maps and classified into different transport modes. The processed and enriched data are then used in the contact tracing algorithm.

process the data, especially if nationwide contact tracing is a final goal. Figure 4.1 shows some of the maps used during the development process. The maps contain different types of information, including possible walk paths (left), buildings and GPS data points, (middle), and a more detailed view of certain buildings that shows metadata information such as the name of the school (last).

Based on these general requirements for GPS data analysis, several main objectives arose that needed to be addressed to make the position data useful and usable for the Smittestopp application. These are as follows:

- Trajectory pre-processing.
- Stop point and mode of transport detection.
- Map matching and map visualization.

Figure 4.2 depicts the complete pipeline for the GPS analytics part of Smittestopp. The whole pipeline is implemented in Python and uses different libraries. Details about the libraries used are provided in the respective sections. The pipeline starts with the user data queried from the database. The data can be queried by user, groups of users, or geographical area. Once the raw GPS data are in the pipeline, they pass through the different building blocks.

Trajectory pre-processing: In this part of the pipeline, the raw GPS data are transformed into trajectories. The transformation depends on the selected method (fixed length, time interval, or stop and transport mode).

Stop point and mode of transport detection: The stop point and transport mode part is responsible for detecting whether a person has stopped (e.g. at a bus stop) and the mode of transport (walking, driving, etc.).

Map matching and map visualization: The map matching and visualization part connects the GPS trajectories and point data to meta-information obtained from public maps, including methods to create bounding boxes and polygons for the trajectories for GPS points.

In the following sections, we describe these different components in more detail.

4.2 Trajectory pre-processing

A GPS trajectory is a collection of GPS points describing an object's movement along a specific path. No clear definition is available on how long a trajectory is and what it should contain, and these often depend on the use case. For example, in an application used to track running, different trajectories could be segments of 1 kilometre or specific times, such as a new trajectory every 10 minutes. Another approach could involve a new trajectory for every trip, resulting in a collection of runs. For Smittestopp, we experimented with different ways of creating trajectories from GPS data. The experiments were inspired by related work by [9, 8, 12, 4]. Unlike the related work, we did not have labelled data for our trajectories, so we decided to develop methods that could work unsupervised. The main goal was to keep the methods simple, understandable, and explainable.

From the raw GPS data, we extracted GPS trajectories that could then be used in other analysis steps, such as to find intersections, points of interest (POIs), visited buildings, travel paths, and so forth. Looking at existing systems and the literature, we saw no clear way to extract trajectories that worked the best, but there were different suggestions. For our pipeline, we decided on three different methods:

1. Trajectories with a fixed length of GPS data points.
2. Trajectories with fixed time intervals (hours, minutes, seconds).
3. Trajectories obtained from stop points and transport modes.

The input data for the trajectory pre-processing are GPS data obtained from smartphones per user per time or region. The outputs are the trajectories per user, as a collection of GPS data points.

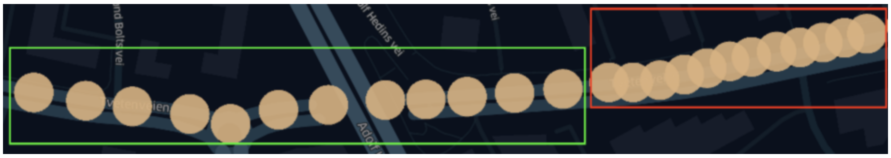


Fig. 4.3: Comparison of changes in a fixed number of GPS points. The green box shows 12 fast-moving GPS points and the red box shows 12 slow-moving GPS points.

4.2.1 Trajectories with a fixed length of GPS data points

In this approach, we divide a long series of GPS data points into trajectories based on a given fixed length. Here, the fixed length is the number of GPS points to be extracted as a single trajectory. Simplicity in handling and processing the trajectories are the main advantages of this method. However, this method's main drawback is that some subtrajectories cover small geographical areas, while others cover large ones. For example, a number N of GPS points for a walking trajectory can be within 1 kilometre, while the same number of GPS points for the trajectory of a fast car can cover around 10 kilometres. To understand this phenomenon, see the illustrations in Figure 4.3. Therefore, when analysing POIs around a trajectory, a large geographical area causes problems in extracting a large amount of additional metadata.

4.2.2 Trajectories with fixed time intervals

In this method, we use predefined time intervals, such as two hours, one hour, half an hour, a minute, and a second to create trajectories. These time intervals are calculated using timestamps, which are part of the metadata of each GPS point collected by the mobile application. The main advantage of this method is its ability to find trajectories for a given time interval. However, similar to the method of extracting a fixed number of points, the resulting trajectories can have an area so large that extracting metadata from the map servers would be too computationally expensive. This function works as a support function to extract POIs from long trajectories, because extracting POIs around a long trajectory is time and resource intensive. The pseudocode of this function is presented in Algorithm 1.

Algorithm 1: Extracting trajectories with fixed time intervals

```
Data: GPSData – A series of GPS points of a trajectory
Result: A list of subtrajectories

timeMode ← SelectTimeMode from (2H, H, HAH, M, S) ;
timeModeInSeconds ← convert timeMode to seconds ;
dividedTimestamp ← GPSdata[timestamp] / timeModeInSeconds ;
uniqueTimestamp ← FindUniqueValues(dividedTimestamp) ;

for $timestamp_i \in uniqueTimestamp$ do
    subTrajectory ← GPSdata[$timestamp == timestamp_i$] ;
    Append $subTrajectory$ to $Result$ ;
end
```

4.2.3 Trajectories based on trips and stop points

A natural way of thinking about a trajectory is in terms of different modes of transport. This approach splits a long GPS trajectory into smaller chunk-based trips and stop points. In this case, a trip is defined as moving some distance over some time, such as a person driving to work or walking a dog. Stop points are points where a person is still for a longer period within a predefined radius. The approach is based on the work of Cich et al. [2]. The advantage of this method is that we obtain a clean set of trajectories pertaining to either stop points or trips that can be further classified into a mode of transport. A disadvantage of this method is that we must ensure that the trajectories obtained are stable enough to detect stop points properly and to distinguish between different trips. Furthermore, this method produces trajectories of various lengths. For example, the trajectory of a person going to the post box would be much shorter than that of a person walking to work.

4.3 Predicting the mode of transport

Even if the physical distance between two people is close enough for infection, it does not necessarily mean they were in contact. For example, one person could be driving a car, while the other one is running on the sidewalk. Furthermore, two people can be a safe distance apart but sitting on the same bus, raising the likelihood of infection. Therefore, it was important to assign a mode of transport to the GPS trajectories generated by the users. For this purpose, we defined seven different types of transport modes: being still, walking, running, on a bus, in a car, on a train, and on a plane (see Table 4.1). These categories were selected based on the most common forms of transportation in Oslo, Norway.

Since we were, for the most part, interested in determining whether or not two people were in direct contact with each other, we generalized these groups further to only include *still*, *on foot*, and *in a vehicle* The group *still* covers the case in which a person is standing still, or, in other words, has no speed; *on foot* encompasses all instances of a person moving on foot, such as walking, jogging, or running; and *in a vehicle* contains all cases in which a person is in a vehicle, including seated inside a car, bus, tram, train, or airplane. Despite being a less precise estimation of the transport mode than in using the seven categories above, these categories still achieved our target of being able to more accurately differentiate between in-person contacts. For example, if contact between individuals is determined, but one is *on foot* while the other is *in a vehicle*, we can rule out the possibility of infection. As development continued, we planned to gradually increase the level of precision of the predicted modes of transport, but this was not achieved before the project was shut down.

Research on deriving transport modes from GPS data relies mostly on supervised learning methods that use a large training data set of labelled trajectories to learn the nuances between the different modes of transport automatically (eg, [10, 13, 6, 3]). Since we did not have access to an extensive training data set or time to collect one, we had to settle for a simpler approach using heuristics. We decided to use the speed of a given GPS point to predict its transport mode. This means that we assigned a mode of transport for each point in a given trajectory. Initially, we planned to assign one transport mode per GPS trajectory, but we found that giving each point a transport mode would provide greater flexibility further down the pipeline.

There were two advantages to this method. First, speed was calculated on the smartphones, such that we obtained this information for free, without the need for any additional computations. Second, assigning a transport mode for each point gave us more flexibility in comparing contacts at specific periods of time. The lookup table used to determine transport modes is shown in Table 4.1, where we also show the conjectured speed of the more precise modes of transport. These speed values were selected by taking the average speed of the transport mode in question and adding a small upper buffer to allow for some variability. Since the accuracy of the collected GPS points varied, the assigned transport modes underwent cleaning and post-processing. This involved looking at the individual points within a given time frame and assigning all points within that frame to the majority transport mode.

The evaluation was carried out by comparing the predicted transport modes against a set of manually annotated points collected by members of the development team. Overall, we had approximately 35 test cases for evaluation, all from Simula employees who gave explicit permission to use their data for testing purposes. Each entry consisted of a start time, an end time, and a mode of transport for that duration. The results were evaluated against this test data set continuously through a series of unit tests that ran every time the analytics pipeline was started. Furthermore, we performed a qualitative evaluation of the predicted transport modes using interactive maps generated by kepler.gl.[1] Based on this visual analysis, we made specific changes

[1] See https://kepler.gl/.

to the transport mode prediction algorithm, such as the speed thresholds used for the individual transport modes and fixing bugs that were not caught by the unit tests. Despite being very useful in creating a more robust and stable transport mode prediction approach, we had to be careful when making changes, since we were using a small subset of data that might not have represented the overall population. It is important to note that these interactive maps were only used in the development phase of Smittestopp, and not in production.

Mode of Transport	Minimum Speed (km/h)	Maximum Speed (km/h)
Standing Still	-	1
Walking	1	6
Running	6	14
Car | Bus | Tram | Train	14	80
Car | Train	80	150
Train	150	220
Plane	220	-
Standing Still	-	1
Running	1	14
In a Vehicle | Public Transportation	14	-

Table 4.1: The lookup table used to determine the transport modes for a given GPS point. The upper half of the table shows the more precise modes of transport, while the lower half shows the modes of transport used in the final version of Smittestopp.

4.4 Map matching and map visualization

Map matching involves combining GPS data with metadata from publicly available maps. For the Smittestopp project, we tested Google Maps, Azure Maps, and OpenStreetMap[2]. Based on the performance we needed and the amount of metadata available for Norway, we decided to use OpenStreetMap. The methods implemented were partially inspired by other work (a good overview can be found in [1]). Nevertheless, we basically started from scratch, to develop methods specifically designed for contact tracing that could provide the information needed and, at the same time, respect the privacy of users. More details on the OpenStreetMap backend we used to query the information are available in the part describing the backend.

Our goal for the map matching was to understand whether a contact had occurred in a closed environment (e.g. a building or a bus), since this could impact the risk level. Contacts are split into three categories: *inside*, which can be inside a given building or a vehicle or public transportation, *outside*, and *uncertain*, when we lack

[2] See https://www.openstreetmap.org.

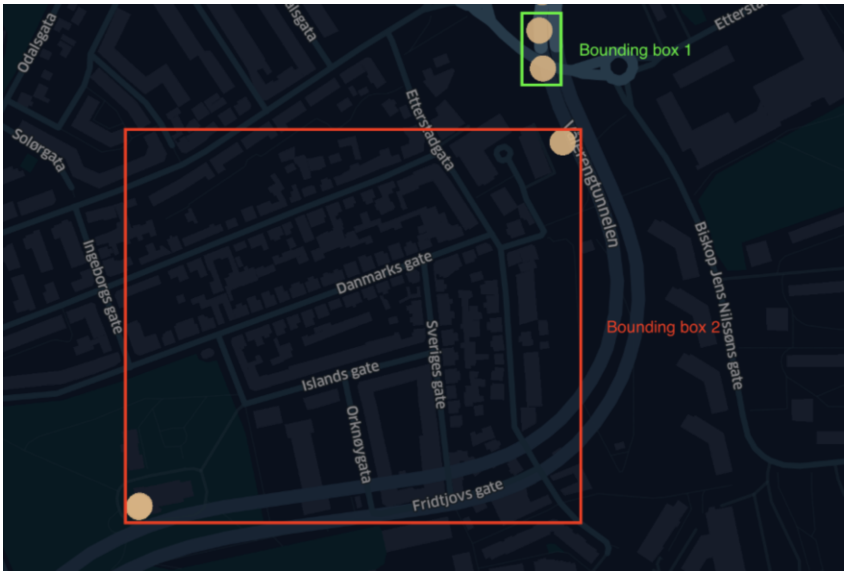


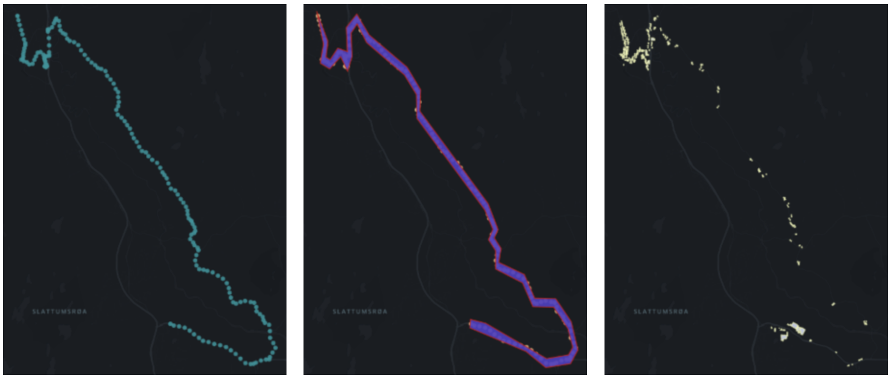
Fig. 4.4: A comparison of two bounding boxes and the areas they cover. Note the differences between the green and red boxes, which both cover the area corresponding to two GPS points.

relevant information or when the data are inconsistent (e.g. one person is detected as being in a car while the other is walking).

4.4.1 Extract POIs from dilated areas

Our main goal is to find POIs around the given trajectories that are associated with positive cases or identified as a trajectory to be analysed. Initially, we considered bounding boxes around trajectories as an area around which to extract POIs. However, the bounding boxes cover a large geographical space than we wanted to analyse to find POIs. The maximum number of unnecessary POI extractions arises when two GPS coordinates are located at two ends of a bounding box's diagonal. Figure 4.4 illustrates this phenomenon. Therefore, we were looking for alternatives, such as extracting POIs using dilated areas.

The main purpose of extracting POIs using a dilated area is to minimize the number of POIs returned for a given trajectory. In this method, we replace the bounding box with a narrowed polygon. The polygon's width (the maximum distance to the polygon boundary from a given point) is defined as an input parameter (a given fixed value). A sample trajectory, a corresponding dilated area, and the extracted POIs within the area are illustrated in Figures 4.5a, 4.5b, and 4.5c, respectively.



(a) Trajectory. (b) Dilated area. (c) Extracted POIs.

Fig. 4.5: An example of a trajectory and its POI extractions using a dilated area.

4.4.2 Obtaining contacted POIs with points

Identifying POIs around a specific GPS coordinate is essential when determining the contacted POIs of a given trajectory. To extract POIs from each GPS point, we consider a circular area around every GPS point of a trajectory, because we did not have access to indoor positioning systems [5]. Practically, when we extract POIs around a point using the radius of a circle, we observe several POIs per point. However, we know that this point could only be in contact with one POI. Therefore, there should be a reliable method to identify the most appropriate POI with a high chance of having contacted a given point.

In this functionality, we can select two options for the value of the radius of the circle, which will be considered the area for extracting POIs. The first option is a fixed radius passed to the algorithm. In this case, we can carry out a heuristic analysis to find the most suitable radius. The second option is to use the accuracy radius, which is part of every GPS coordinate collected by our mobile application. Considering the accuracy value (radius of the accuracy circle) as the radius to extract POIs is logically more related to our goal than the fixed radius, because, when a mobile phone has a poor GPS signal, the radius of the accuracy circle is bigger to extract POIs around it. On the other hand, a large error (when the GPS signal is poor) can lead to a large radius. Then, the POIs cannot be extracted because of there being too much data. To overcome this problem, we defined a threshold for the maximum radius.

Within a single circle, we can sometimes observe several POIs, as mentioned earlier. However, we have to select one of them. We do so by maintaining a ranking score for every POI extracted for a given trajectory. In this scoring method, we increment a counter (starting from zero) when a POI is detected as a contacted POI for a given GPS point. For example, assume that R GPS points intersect with a building B. Then, the rank of building B is R. Ultimately, each and every POI around a given trajectory is assigned a rank based on the number of intersected GPS points. If we detect a single POI for a given point, we consider that POI as contacted. If we have more than one contacted POI, we use the POI with the highest rank. However, if we have numbers of equal rank, we select the first POI detected as the point contacted. The pseudocode for this algorithm is given in Algorithm 2.

Algorithm 2: Obtain the POIs of contacted points.

```
Data: GPSData, amenities, paddings, maxPadding, columnName,
      OverpassAPI
Result: OutputPerPoint, POIsInfo, POIsCount

GPSOutput ← [ ] ;
POIsInfo ← [ ] ;
POIsCount ← [ ] ;

if padding is not given then
    paddings ← List[Minimum(accuracy, maxPadding) for all GPS records];
end

Output ← CallingToOverPassServer(GPSData, amenities, paddings);
OutputPerPoint ← split Output to length of GPSData;
for POIs in OutputPerPoint do
    if POIs is not Null then
        append POIs to POIsInfo;
    end
end

if POIsInfo is not empty then
    POIsCount ← CountUniquePOIs(POIsInfo);
end
for ContactedPOI in POIInfo do
    ranks ← calculate ranks of ContactedPOI;
    POIMaxRank ← selectMax(ranks);
    append POIMaxRank to POIsInfo;
end
```

4.4.3 Relation between POIs and transport modes

Transport modes and POIs are closely related, since both indicate whether the contact could have happened in a closed environment. We describe this interaction in a few points below.

First, POIs are only queried at instants when the trajectories in contact are associated with transport modes indicating that the contact could indeed have happened. In a nutshell, we will not check whether people considered to be in a car were inside a building. More precisely, we will query POIs for trajectories satisfying the condition that either (i) both are considered to be *on foot* (which covers walking and running) or *still* or (ii) one of the trajectories is considered so and we do not have data for the other (transport mode *N/A*). The result of querying the POIs will lead to their classification as either *outside* or *inside*.

Second, parts of trajectories for which either (i) both are considered inside transport or (ii) one of them is and we lack data for the other will be considered as inside a transport, which means the contact will be classified as *inside*. Table 4.2 summarizes how we first infer the *contact transport modes* from the transport modes of each trajectory. A contact inside transport is considered to occur inside, and looking for POIs will lead to contacts occurring either outside or inside.

Trajectory	Still	On Foot	Public Transport	Vehicle	N/A
Still	Out/Inside	Out/Inside	Uncertain	Uncertain	Out/Inside
On Foot	Out/Inside	Out/Inside	Uncertain	Uncertain	Out/Inside
Public Transport	Uncertain	Uncertain	Inside	Inside	Inside
Vehicle	Uncertain	Uncertain	Inside	Inside	Inside
N/A	Out/Inside	Out/Inside	Inside	Inside	Uncertain

Table 4.2: This table describes how the different contact modes (inside, outside, or uncertain) are informed by the transport modes of the trajectories. For example, if two users who meet have *vehicle* as the transport mode in the same location, the contact is defined as happening inside.

We also apply a smoothing function to the trajectories' transport modes, to detect suspicious transport modes, which involve data we assume to be spurious and due to incorrect transport mode attributions. More precisely, we initially compute whether the different points of the trajectories are those for which we should query POIs or ones judged to be *inside transport*. Afterwards, we smooth this prediction by re-attributing the predicted value for points of contact whose prediction differs from those both before and after. More specifically, we look for points of contact t_i such that contact at t_i is considered inside a vehicle, but contacts at $t_{i-2}, t_{i-1}, t_{i+1}$, and t_{i+2} (where the length of this interval is a parameter) are considered to be walking/still. Such points, if below a certain duration threshold, are transformed into the opposite prediction, which is, in this case, not *inside transport*.

Once this smoothing has been applied to the contact's transport modes, the points are classified as (i) inside a transport (e.g. contact happening inside a car); (ii) points for which people are walking/still and we would like to know if they met outside or, for example, inside a school or a shop; and (iii) points with inconsistent transport modes, whose contact location/context will be considered *uncertain*.

Lastly, the radius of search of POIs depends on the transport mode detected. The reason we generally use a nonzero radius – that is, a point need not strictly be within a polygon, but only very close to it – is to handle GPS imprecision, which can typically be high inside a building. On the other hand, keeping a high radius can lead to trajectories on the pavement being perceived as happening in nearby shops alongside that pavement. A compromise is to first compute a search radius based on the GPS inaccuracy and to apply a coefficient based on the transport mode detected (e.g. equal to 1.0 if both trajectories are *still*, but reduced to half if both are *on foot*).

4.4.4 Querying POIs

Once the transport modes are analysed, the last step of our method to obtain the context of a contact (see *Algorithm 3*) is to query points of interest for parts of the trajectories. First, we feed chunks[3] of these trajectories to the functions mentioned above that query contacted POIs. Second, we apply smoothing to these detected points, very similarly to smoothing we applied to the transport modes. Since we use a nonzero radius to search for POIs, the role of this filtering is to eliminate contact points that actually might not have happened in a closed environment. Such scenarios – isolated trajectory points being attributed to a POI for a short duration – could involve, for example, nearby shops when walking on the pavement.

When querying POIs from OpenStreetMap Overpass, we receive specific information with very precise names[4]. Therefore, we design a dictionary converting the information detected into simple categories names (e.g. we extract the value *dormitory* and convert it into the general term *residential*). We used the following list of (generalized) POIs: hospitals, nursing homes, schools, kindergartens, universities, bars and restaurants, sports facilities, culture and entertainment facilities, residential, religious buildings, shops, other education facilities, other healthcare facilities, and others.

Information returned to reports include the duration of contact within each of the three categories (*inside*, *outside*, and *uncertain*), as well as the types of inside contact (inside a mode of transport or one of the POIs following the aforementioned list) and their respective durations. Moreover, we returned a filtered version of the POIs, keeping only locations above a certain duration and/or relative duration threshold.

The data obtained from the POIs are then part of the information of a given contact. More specifically, the POIs are an attribute of the contacts, and the aforementioned information (duration inside and outside, locations of contacts, transport modes) are stored within dictionaries describing a contact's properties. These dictionaries are used to display results when executing specific code or are sent as JSON objects to the frontend. When producing the reports, one can then retrieve the particular

[3] It became progressively clear that, to handle the data we received, we had to compute the nonzero duration of single timestamps. In other words, we interpreted the duration of the segment $[t_i, t_j]$ as roughly $(t_{j+1/2} - t_{i-1/2})$, with $t_{j+1/2}, t_{i-1/2}$ defined using previous and subsequent timestamps. This allows single timestamps to have a nonzero duration. To ensure that we are not capturing excessively long durations (e.g. where the phone stops recording so that $t_{j+1} \gg t_j$), we cut up the trajectories into chunks of typically two hours.

[4] See https://wiki.openstreetmap.org/wiki/Points_of_interest.

locations if contacts between two users occurred, as well as obtain a summarized description of the most important of all the users' contact locations.

4.4.5 Accuracy of the identified POIs

The quality evaluation of the predicted POIs was carried out using data for which we had ground truth generated by members of the Smittestopp development team to test different scenarios and edge cases. Thus, different parts of the code were changed based on these tests (e.g. varying the accuracy with transport modes, asking for one or both individuals to share a specific type of transport mode).

In their definitive form, the POIs seemed accurate and coherent with the ground truth available. However, we have no quantitative figures to more precisely judge their accuracy. Besides, one should bear in mind that several steps filter out potential errors (if they account for a negligible part of the predictions). Such cleaning steps include the generalization of the detected points of entities (a step related to privacy concerns) into more general categories – for example, the algorithm can detect a 'apartment' instead of a 'cabin' and, since both are displayed as being 'residential', the error would be invisible in the reports – as well as the filtering of POIs accounting for a short duration and the relative duration of the contact – for example, if one identifies a 'shop' for a few seconds that was not in the ground truth data, it could be filtered out and again become invisible in the reports.

Algorithm 3: Computing POIs for a contact.

```
Data: Contact details, incl. locations, timestamps, accuracy. ;
Transport modes of each trajectory.
Result: POIs, duration of contact inside, outside, uncertain.
Get types of transport modes (inside transport, on foot or still, uncertain) for the contact
  from each trajectory's transport modes. ;
Get the radius of search for POIs based on transport modes and accuracy of the contact. ;
Split the contact into chunks of shorter duration (2 hours). ;
for chunk of trajectory in full trajectory do
    Call 'Get POIs of contacted points' to attribute a POI or none to each trajectory point. ;
    Eliminate 'suspicious PoIs' i.e., indices whose labels are isolated (surrounding
      instants not related to the same POI) and of short duration are filtered out. ;
    Keep only PoIs for which the contact's transport modes are consistent (on foot/still). ;
    Extract building/PoI type and convert it into more general categories.
    for label type, i.e., some PoI, inside transport, outside, uncertain do
        Get the duration of contact related to that label (e.g. inside transport) by
          computing the duration of consecutive indices of identical label. ;
        Update the total duration of contact inside, outside, uncertain. ;
    end
end
```

4.5 Challenges, experiences, and lessons learned

The treatment of GPS data led to multiple challenges, as well as being time-consuming and resource intensive. GPS signals often being very imprecise, we had to reconsider early plans to adapt them to typical levels of inaccuracy or issues in the real data gathered. These issues include difficulties in determining transport modes (since successive inaccurate positions can lead to incorrect speed computations) or POIs (forcing us to relax the strict requirement that a point must be within a building's OpenStreetMap polygon to be considered inside it). In addition, we considered leveraging these very difficulties to our advantage, such as attempting to deduce being inside from more inaccurate measurements, which did not perform satisfactorily. Great opportunities are associated with the use of GPS data in combination with external resources such as public maps and metadata, which constitute a powerful tool to infer much information. Many functionalities were eventually left out, not used, or not finished before Smittestopp was shut down, including street map analysis using pathways and distinguishing public transportation use from other personal transportation, among many others. Proper followup studies and more development time would be needed to observe the effects and value of such features for contact tracing.

4.6 Ethical considerations

The use of GPS data raises many questions about data privacy. Since our analysis can trace a person's locations, it allows us to access much personal information. For example, using the collected data, we can determine where a person went to work, when the person left for the gym, and what restaurant the person went to for dinner. Had one kept the raw information from OpenStreetMap, even more sensitive information would have been displayed. We intended to use the minimal amount of informative data, leaving out all that seems unnecessary or ethically questionable. Following that principle, we returned rather general terms (e.g. the residential label instead of an address), protecting more sensitive information (e.g. addresses, names, and the type of apartment a person was visiting). Moreover, one should keep in mind that the personal information mentioned is intrinsic to GPS data, in the sense that we did not add any information sources (e.g. public telephone book entries) that would have allowed for the easy identification of the person, simply using OpenStreetMap, which can be accessed as soon as one acquires GPS data.

To protect privacy, we also exclusively used the data of Simula employees who were part of the Smittestopp application development team and who agreed to being part of the testing. Information from these test subjects and the resulting maps were not made public or shared with anyone besides project management, and then only during the development stage. The data collection and analysis were kept separate. Additionally, later on, we decided to hide sensitive information by not providing the interactive maps created for the development stage in the presentation of the results (since visualizing the movements of users further raised privacy issues), as well as combining relevant information in summary form (e.g. communicating the most common transport modes of a contact, rather than the transport modes for all instances of that contact). No interactive maps were used in the production system, but static maps were implemented within the analysis pipeline.

4.7 Summary and conclusions

We presented and discussed the GPS data analysis pipeline and methods developed for Smittestopp. In addition, we discussed lessons learned and ethical considerations. Overall, we can summarize that GPS data hold a great deal of potential for contact tracing, especially if combined with metadata from public maps and related databases. Nevertheless, this information can be seen as optional and might not be relevant in determining close contacts. This information is also associated with substantial ethical and privacy-related concerns, and one needs to weigh the usefulness and added value against these. For Smittestopp, we implemented the features that we found most useful with the smallest privacy impact, and omitted functionalities such as geomapping/tracking and person identification (which is possible to a certain degree of accuracy with the data at hand, public maps, and the person's registration information).

References

[1] P. Chao, W. Hua, R. Mao, J. Xu, and X. Zhou. A survey and quantitative study on map inference algorithms from gps trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[2] G. Cich, L. Knapen, T. Bellemans, D. Janssens, and G. Wets. TRIP/STOP Detection in GPS Traces to Feed Prompted Recall Survey. *Procedia Computer Science*, 52:262 – 269, 2015.

[3] M. Etemad, A. S. Júnior, and S. Matwin. Predicting transportation modes of GPS trajectories using feature engineering and noise removal. *CoRR*, abs/1802.10164, 2018.

[4] R. Mariescu-Istodor, A. Tabarcea, R. Saeidi, and P. Fränti. Low complexity spatial similarity measure of gps trajectories. In *WEBIST (1)*, pages 62–69, 2014.

[5] R. Mautz. Overview of current indoor positioning systems. *Geodezija ir Kartografija*, 35(1):18–22, 2009.

[6] H. Omrani. Predicting travel mode of individuals by machine learning. *Transportation Research Procedia*, 10:840 – 849, 2015. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands.

[7] A. C. Prelipcean, G. Gidófalvi, and Y. O. Susilo. Transportation mode detection – an in-depth review of applicability and reliability. *Transport Reviews*, 37(4):442–464, 2017.

[8] X. Yang, K. Stewart, L. Tang, Z. Xie, and Q. Li. A review of gps trajectories classification based on transportation mode. *Sensors*, 18(11):3741, 2018.

[9] Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):1–41, 2015.

[10] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding Transportation Based on GPS Data for Web Applications. *ACM Trans. Web*, 4(1), 2010.

[11] Y. Zheng, H. Fu, X. Xie, W.-Y. Ma, and Q. Li. *Geolife GPS trajectory dataset - User Guide*, geolife gps trajectories 1.1 edition, July 2011. Geolife GPS trajectories 1.1.

[12] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800, 2009.

[13] F. Zong, Y. Bai, X. Wang, Y. Yuan, and Y. He. Identifying travel mode with GPS data using support vector machines and genetic algorithm. *Information*, 6(2):212–227, 2015.