applied sciences

MDPI

Article

Corridor Detection from Large GPS Trajectories Datasets

Claudia Cavallaro [1,*] and Jordi Vitrià [2]

[1] Department of Mathematics and Computer Science, University of Catania, 95125 Catania, Italy
[2] Department of Mathematics and Computer Science, Universitat de Barcelona, 08007 Barcelona, Spain; jordi.vitria@ub.edu
* Correspondence: claudia.cavallaro@unict.it

Received: 26 May 2020; Accepted: 15 July 2020; Published: 21 July 2020

Abstract: Given the widespread use of mobile devices that track their geographical location, it has become increasingly easy to acquire information related to users' trips in real time. This availability has triggered several studies based on user's position, such as the analysis of flows of people in cities, and also new applications, such as route recommendation systems. Given a dataset of geographical trajectories in an urban metropolitan area, we propose a new algorithm to detect corridors. Corridors can be defined as geographical paths, with a minimum length, that are commonly traversed by a minimum number of different users. We propose an efficient strategy based on the Apriori algorithm to extract frequent trajectory patterns from the geo-spatial dataset. By discretizing the data and adapting the roles of itemsets and baskets of this algorithm to our context, we find the longest corridors formed by cells shared by a minimum number of trajectories. After that, we refine the results obtained with a subsequent filtering step, by using a Radius Neighbors Graph. To illustrate the algorithm, the GeoLife dataset is analyzed by following the proposed method. Our approach is relevant for transportation analytics because it is the base to detect lacking lines in public transportation systems and also to recommend to private users which route to take when moving from one part of the city to another on the basis of behavior of the users who provided their logs.

Keywords: GPS trajectory; frequent urban corridor; movement patterns; trajectory data mining

1. Introduction

The analysis of large human mobility datasets has the potential to provide many useful suggestions to public operators as well as to individual users, but in order to provide this information in a timely manner, efficient algorithms are needed.

In this scenario, corridor detection has emerged as one of the key elements to make informed decisions about public transportation systems as well to recommend optimal routes to individual users. In this case, the use of a brute-force algorithm is not an alternative due to its computational cost and more efficient solutions are necessary. To this end, we are proposing in this paper a new approach, based on the Apriori algorithm [1], which is suitable to the use in very large datasets. In order to show its performance, we have analyzed the dataset of the GeoLife project [2] of Microsoft Research Asia, which includes geographical trajectories of 181 users that uploaded GPS recordings corresponding to their routes in Beijing and its surroundings.

The algorithm Apriori, introduced by Agrawal et al. in [1], is a popular data mining algorithm for market basket analysis to detect which items are frequently bought together. This algorithm can be used in our scenario to reduce computational complexity. The key point of its application is to consider different assignments for its *items* and *baskets*. Specifically, we consider two alternating roles for *items* and *baskets*: in the first case, GPS points are represented as a set of *items* and trajectories as a set of *baskets*, but later we reverse this assignment and consider GPS points as a set of *baskets* and trajectories as a set of *items*. This strategy is particularly suitable in our case, where we have to identify corridors for a large volume of data and the naive application of Apriori would exceed any memory setting. Both versions of the Apriori algorithm can be seen as filters that act on the set of points and trajectories. In the first case, we are able to detect sets of points that are shared by at least a number trajectories. These points are not necessarily aligned to form trajectories. In the second case, we detect sets of trajectories that contain at least a number of points. By combining both approaches, the set of candidates that must be check to determine the final set of trajectories is reduced to a very low cardinality. The keypoint for this strategy is to realize that the complexity of Apriori depends not only on the number of items and bags, but also on the number of actual itemsets that are present in bags. By reversing the roles of items and bags and doing a final check of the remaining candidates, we bypass this bottleneck.

The Apriori algorithm works in a discretized space and its results must be refined. To this end, we use the Radius Neighbors Graph, which uses the mapping graph together with the adjacency structure of the GPS points in the neighborhood of the points of a fixed trajectory.

This article is organized as follows. In Section 2 we survey the literature concerning the study of GPS datasets. In Section 3 we describe the dataset used to performed experiments and propose some pre-processing operations we have applied to clean it. Section 4 recalls basic concepts and definitions used in the paper and describe the naive brute-force algorithm to detect corridors. In Section 5 we provide a description of our approach. It can be decomposed into three main processing modules and we describe how to detect, in a given set of trajectories, those corridors that are frequently used by different trajectories. We explain the application of the Apriori algorithm in Section 5.1. In the following section we present how, with different strategies, a set of candidate corridors is induced. Section 6 presents an overview of experimental results. Section 7 shows the last phase of our process also showing a final corridor of GeoLife as an example. Section 8 concludes the article and presents perspective of future works.

2. Related Works

There are several approaches in the literature regarding the analysis of geographic trajectories. In fact, this is today an important research topic due to the increase in the volume of geographic information.

Zygouras and Gunopulos propose in [3] the analysis of real GPS trajectories collected from taxis operating in the city of Porto, moving buses in Dublin and Atlantic hurricanes from 1950 to 2004. In order to detect corridors, they discretize the trajectories using a grid and decompose the space in different sets of frequently observed locations. The Latent Dirichlet Allocation (LDA) model [4] is applied to the trajectory dataset to extract frequent traffic patterns. They apply a hierarchical clustering algorithm to the subtrajectories of each frequent set, following a bottom-up approach, using Dynamic Time Warping (DTW) [5] in order to measure the distance between two subtrajectories. Finally, another algorithm selects the set of corridors from the set of candidate corridors minimizing the Minimum Description Length (MDL) principle [6].

Bicocchi et al. [7] propose a system that suggests daily and local transport sharing opportunities for short-term trips, by analyzing the traces of urban mobility in Milan and Turin. Based on "Call Detail Record" (CDR) data, they propose algorithms to recognize similar paths that can be used to recommend shared rides. The data used for the experiments were provided by a telecom mobile operator and support a travel recommendation system for multiple users. General mobility routines are identified through an extension of the probabilistic generative LDA model, performed on the set of available trips, including among others, leisure mobility routines and commuting trips.

Both referred works seek paths that are common to multiple users, organizing the movements made in time slots, but with different approaches. In both cases geographical distance is measured through the Haversine formula.

Buchin et al. [8] consider the problem of detecting commuting patterns and propose new algorithms that cluster subpaths of given trajectories. The idea is to select a reference trajectory and then find all subtrajectories that are close to this one by using the Fréchet distance. They consider the optimization problem of finding the longest cluster of a fixed size and the largest cluster of a fixed length. The Fréchet distance is a distance measure for continuous shapes, such as curves and surfaces, and it is defined using reparameterisations of the shapes. Since it takes the continuity of the shapes under consideration, it is generally considered to be the most appropriate distance measure for curves.

Rolim et al. propose in [9] a method to identify movement patterns of sets of trajectories and analyze simulated trajectories in a region of Itaim Quarter in Santa Catarina, Brazil. They consider the frequency distribution of points for identifying a set of frequent regions. The framework consists of two phases: a partitioning phase, in which the trajectories are segmented using the Minimum Description Length principle and a clustering phase, that considers the density to group similar segments in a cluster using the Fréchet distance as a measure of similarity between curves.

Devogele et al. [10] describe a new algorithm to compute Discrete Fréchet Distance (DDF) which aims to lower computation time and improved precision. It includes three different improvements: the first one is the Douglas and Peucker filtering process. Indeed, for GPS trajectories, the number of positions can be dramatically reduced. The second one is a pruning process. Only a small part of the two matrices required to compute the discrete Fréchet distance are computed. The last one is an improvement of the accuracy of the DDF. The proposed method is more complex than the regular discrete Fréchet distance, but CPU time is reduced. In terms of accuracy, a balance between CPU time and precision is required.

Crociani et al., in [11], propose an unsupervised learning approach, based on the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, for automatic lane detection in multidirectional pedestrian flows. Unlike our work, they focus on pedestrian dynamics in a video using a distance function that takes into account the angular distance between the vectors. After that, they aggregate instantaneous information on both position and speed of pedestrians to form clusters on short time windows.

DBSCAN algorithm is used in [12] in order to detect Point Of Interests (POIs) in Beijing. The proposed architecture is used for travel recommendations.

Zheng et al. [13] aim to mine interesting locations and classical travel sequences in a given geospatial region. They first model multiple individuals' location histories with Tree-based hierarchical graph (TBHG). Next, supported by the TBHG, they propose a Hypertext Induced Topic Search (HITS) [14] inference model, which consider user's location as a directed link from the user to the location.

Hosseinpoor et al. [15] introduce an approach for identifying critical points on a large volume of real data. Three trajectory descriptors, namely the curvature areas, the turning points and the self-intersecting points, are determined by detecting the critical point of trajectories using a convex hull (TCP-CH) algorithm. These characteristics allow the detection of the directions of change, represent the changes of the path and the presence of double-level intersections. They are useful in order to show, in addition to anomalous values, the geometric properties of the trajectories: shape, complexity, direction and distance. The approach allows, by selecting some trajectory parameters, to identify the number and positions of the points of interest and identify from them the similarities of the trajectories.

3. GeoLife Dataset

The GeoLife dataset [16,17], which was collected from 2009 to 2012, consists of more than 16 thousand trajectories with a total of 18 million GPS points. A total of 95% of trajectories were logged in a dense representation, e.g., every 1~5 s or every 5~10 m by point. In order to tackle data inconsistencies, we performed several cleaning operations. For each trajectory, we only considered the GPS points with a speed between 0 and 100 m/s. This filtering operation can be implemented by calculating the instantaneous speed of each point. Detected errors correspond to events where the GPS device stopped working properly, not recording the elapsed time.

We chose to analyze the trajectories inside the Beijing metropolitan area, corresponding to the following range of longitude and latitude: $[116.1, 39.7]$, $[116.7, 40.13]$. Since in this zone 0.0009 degree of latitude and 0.00118 degree of longitude correspond approximately to 100 m, the selected rectangular area is 51 long and 48 km wide. Figure 1 shows the filtered trajectories on a map of this area.

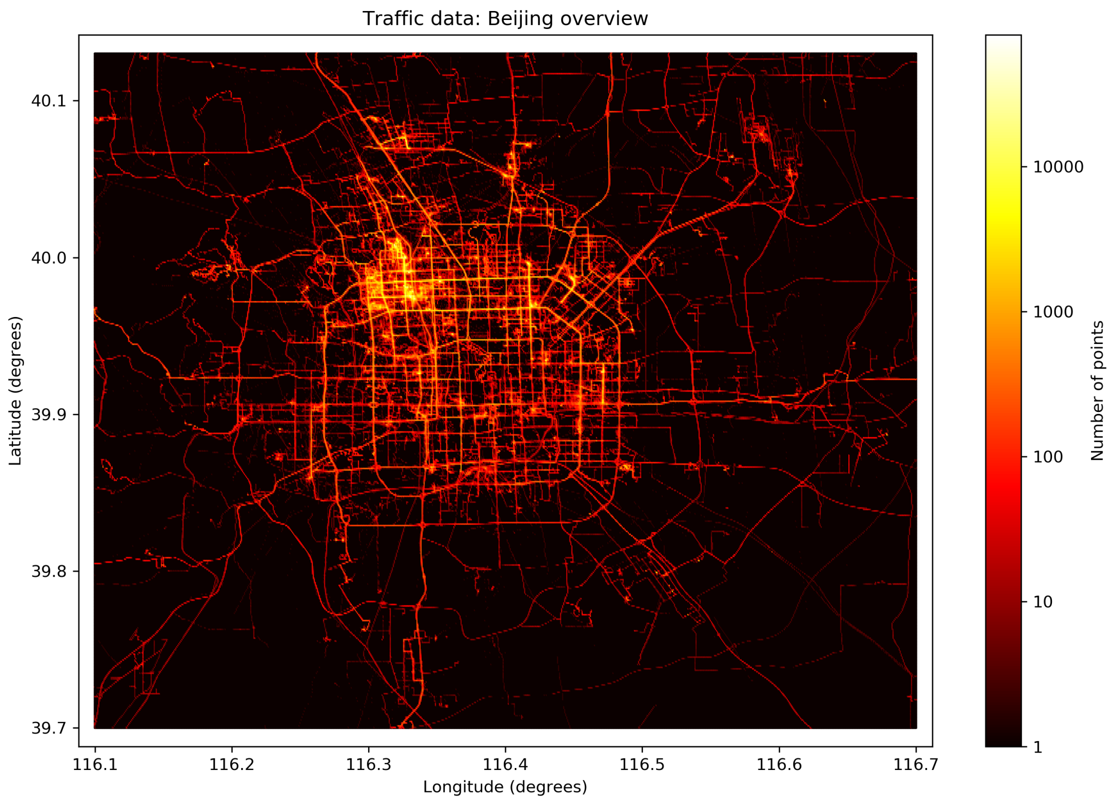

Figure 1. GeoLife trajectories in Beijing metropolitan area.

In this dataset some paths appear broken, probably due to the presence of buildings or tunnels that disturb the GPS signal. We have decided not to tackle this problem. We have divided the dataset into 6 time slots of 4 hours each in order to analyze flows of trajectories in different scenarios.

4. Basic Concepts and Definitions

GPS coordinates represent latitude and longitude, which are angles that uniquely define points on a sphere.

Consider the equatorial plane, i.e., the plane containing the equator, which passes also through the center C of the sphere. The latitude of a point P on the surface is the angle between this plane and the straight line passing through P and C. The latitude is positive if P is above the equatorial plane, and it is negative if P is below. The longitude of a point P on the surface is the angle between the plane containing the prime meridian and the plane containing the meridian passing through P. The longitude is positive if P is to east of the prime meridian, whereas it is negative if P is to west. Therefore, the longitude can range up to $+180$ degrees.

A trajectory T is defined as $T = ((lat_1, long_1, date\_time_1); \ldots; (lat_n, long_n, datetime_n))$, where lat_i and $long_i$ for $i = 1 \ldots n$ are the latitude and longitude in decimal degrees of the GPS points of the trajectory, date_time$_i$ is the time and the date in which every point has been registered and n is the length of the trajectory T (the total number of its points). Each trajectory in the GeoLife database is identified by a trajectory ID and by the user ID who traveled it.

A trajectory log is a set of geographic locations corresponding to a user in motion, ordered by time, from the moment he starts recording his journey until the moment he stops. It can also contain additional information such as transportation mode labels: driving, taking a bus, riding a bike or walking.

The objective of this paper is to propose a practical method to detect, in a large database of trajectories T, all connected paths $P_j = \{(lat_1, long_1), \ldots, (lat_n, long_n)\}$ with a minimum length M (in terms of cells) that are commonly traversed by a minimum number S of different trajectories. These paths are called corridors.

The application of brute-force corridor detection strategies is not possible in real datasets because of their computational complexity: computing the distances of all possible pairs of points for each pair of trajectories is not feasible. In order to tackle this problem, we propose a filtering strategy based on the Apriori algorithm.

Considering previous works in the field, for computing distances between points and consequently the length of paths, we will consider the Haversine distance [18] instead of the Euclidean one. Important in navigation, the first table of Haversines was published by Andrew in 1805, but the term Haversine was coined in 1835 by Inman [18]. Haversine formula is a particular case of the law of Haversines, that relates the sides and angles of spherical triangles. The Haversine distance [19] $d(P_i, P_j)$ between two points P_i and P_j, which gives the great-circle distance in kilometers between two points on a sphere, is defined as follows:

$$d(P_i, P_j) = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_i - \varphi_j}{2}\right) + \cos(\varphi_i)\cos(\varphi_j)\sin^2\left(\frac{\lambda_i - \lambda_j}{2}\right)}\right)$$

where φ_i, φ_j are latitude of P_i and latitude of P_j (in radians), λ_i, λ_j are longitude of P_i and longitude of P_j (in radians), and R is the Earth's mean radius in kilometers.

This formula calculates the shortest distance over the earth's surface, a measure that is called the geodesic curve, deliberately ignoring the effect of elevation differences.

Naive Corridor Detection

In this work, we initially apply a grid of uniformly sized cells to discretize the trajectories. The collection of trajectories D of the original dataset is transformed into a new collection of trajectories D', where each trajectory is represented as a sequence of grid cells. If two or more consecutive coordinates are mapped into the same grid cell, we report only the first instance, not allowing a trajectory to have consecutive points of the same grid cell.

Additionally, we consider that trajectories are not characterized by traveling direction: going forward or backwards on a sequence of cells results in two instances of the same trajectory. Hence, trajectories can be seen as a path in an undirected graph.

Given this trajectory representation, we can define a corridor: A *(M,S)*-corridor is a sequence of M cells in D' such that all cells are shared by at least S trajectories in a given time slot.

A first naive approach to detect corridors is described in Algorithm 1.

For every trajectory, this algorithm scrolls down its elements to decide whether a pair of consecutive cells are shared by at least other $S - 1$ trajectories. If this happens, it must be checked whether the subsequent cell, together with the frequent pair of cells found, constitutes a frequent triple for at least S trajectories. This continues until you find the largest set of frequent cells.

The complexity of this algorithm is a function of L and N. The number of performed operations is $\sum_{i=1}^{L-S+1}(N - M + 1) * (L - i) * (N - M + 1) * 2$, which results in a complexity $O(N^2L^2)$. This complexity is to high when considering datasets with millions of data points and thousands of trajectories.

Algorithm 1: Brute-force approach to find corridors

```
**Procedure:** Find_Corridors(($T_1, T_2, \ldots, T_L$),$M$,$S$)
**Input:** A set $T_1, T_2, \ldots, T_L$ of $L$ trajectories formed by up to $N$ cells each, $M$: the minimum
  number of cells for a corridor, $S$: the minimum number of trajectories for a corridor.
**Output:** $C$ the set of ($M$,$S$)-corridors.
$C = \{\}$
for $T$ from 1 to $L - S + 1$ do
    for $i$ from 1 to $N - M + 1$ do
        $A = (T[i], T[i+1], \ldots, T[i+M-1])$
        $s = 1$
        for $T' > T$ do
            for $j$ from 1 to $N - M + 1$ do
                if $A = (T[j], T[j+1], \ldots, T[j+M-1])$ or
                $A(T[j+M-1], \ldots, T[j])$ then
                    $s = s + 1$
                end
            end
        end
        if $s \geq S$ then
            add $s$ to $C$
        end
    end
end
```

5. Proposed Method

The proposed method comprises two phases: first the generation of candidate corridors, which are coarse traces of several distinct trajectories, and then filtering candidates for the selection of fine-grained corridors.

For the first phase, we define a grid of uniformly sized cells on the map of interest, mapping the coordinates (in decimal degrees) of the points for every trajectory to discrete grid cells. For example, with square cells 1 km wide, we get a 51×48 cell grid to apply to the selected dataset (see Figure 2). In Figure 2, it is possible to notice more intense cells which indicate a greater concentration of trajectories that cross them.

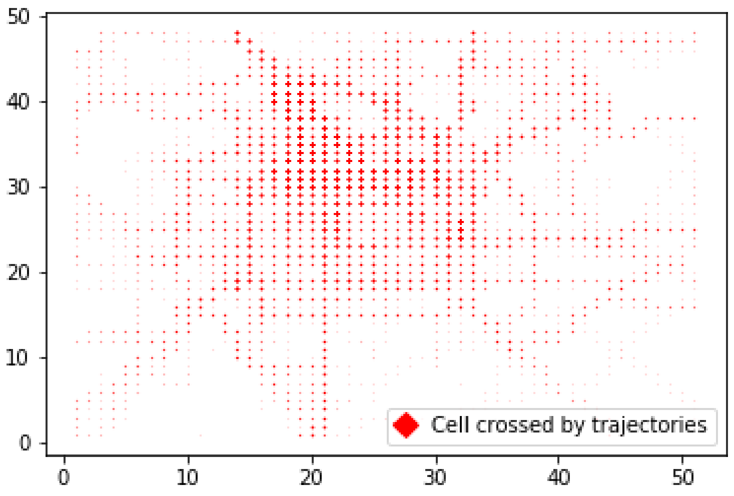

Figure 2. Trajectories mapping into a grid with cells of 1 km per side.

Based on this trajectory representation, we can generate a set of candidate corridors to obtain information on the areas that approximate the sub-routes common to several users with the Apriori algorithm.

In the second phase, this information is filtered using a Radius Neighbors Graph and used to find fine-grained frequent corridors.

5.1. Generation of Candidate Corridors

The Apriori algorithm is a frequency-based approach that allows us to find sub-paths of the road network which are shared by at least a fixed number of trajectories.

The Market-Basket Model of Apriori is based on supposing that we have to different sets:

- A large set of items, e.g., things sold in a supermarket.
- A large set of baskets, each of which is a small set of the items, e.g., the things one customer buys in one day.

The Apriori algorithm objective is to find sets of items that appear frequently in the baskets. To this end, we can define the support for an itemset I as the number of baskets containing all items in I. Given a support threshold s, those sets of items that appear in more than s baskets are called frequent itemsets.

Apriori is an iterative search algorithm in which itemsets of cardinality K are used to produce itemsets of cardinality $K + 1$. This is based on a simple principle: if a set of items is frequent, all subsets of this set are also frequent. The advantage is that at each Apriori level, the algorithm checks whether a tuple is frequent only if it is made up of other frequent tuples with smaller support.

Apriori consists of a "Join" function, which generates a new combination of elements (the candidates) and a "Prune" function in which combinations of elements that do not satisfy the minimum support are eliminated. Its pseudocode [20] is presented in Algorithm 2.

The application of the Apriori algorithm in the case under consideration can be improved by setting constraints: for example, by considering that K-tuples can only be formed by groups of adjacent cells.

Extending the search for other frequent cells to be aggregated only to the neighborhood located at the edge of these cells, since a cell can have at most eight adjacent cells, would lead to the reduction of the execution time in the generation of the candidate cells.

We propose the application of the Apriori algorithm in three steps, following a filtering strategy to improve execution time significantly.

The Apriori algorithm allows the progressive identification of cells that are common to several trajectories and the identification of subsets of cells with low support of trajectories. Our goal is to find the largest subsets of cells shared by at least S trajectories. After that we verify if these subsets are corridors.

We will illustrate some practical considerations related to the method by explaining its application to the GeoLife dataset.

5.2. Step 1

The first step of our approach is to discretize the trajectories. Each trajectory, that was first considered as a sequence of GPS points, now becomes a list of consecutive and distinct geographical cells.

Algorithm 2: Apriori

```
Input: $D'$ : transaction database; min_sup: the minimum support threshold for every itemset
Procedure: Apriori($D'$)
$L_1$=find_frequent_1-itemsets($D'$)
for ($k = 2$; $L_{k-1} \neq \emptyset$; $k++$) do
    $C_k$=Apriori_gen($L_{k-1}$)
    foreach transaction $t \in D'$ do
        $C_t = \text{subset}(C_k, t)$
        // get the subsets of $t$ that are candidates
        foreach candidate $c \in C_t$ do
            $c.count++$
        end
    end
    $L_k = \{c \in C_k | \text{ c.count} \geq min\_sup\}$
end
return $E=\bigcup_k L_k$

Procedure: Apriori_gen($L_{k-1}$: frequent($k-1$)-itemsets)
foreach itemset $\mathcal{L} \in L_{k-1}$ do
    foreach itemset $\mathcal{N} \in L_{k-1}$ do
        if $(\mathcal{L}[1] = \mathcal{N}[1]) \cap (\mathcal{L}[2] = \mathcal{N}[2]) \cdots \cap (\mathcal{L}[k-2] = \mathcal{N}[k-2])$
          $\cap(\mathcal{L}[k-1] < \mathcal{N}[k-1])$ then
            $c = \mathcal{L} \times \mathcal{N}$
            // join step: generate candidates
            if has_infrequent_subset($c, l_{k-1}$) then
                delete $c$
                // prune step: remove unfruitful candidate
            else
                add $c$ to $C_k$
            end
        end
    end
end
return $C_k$

Procedure: has_infrequent_subset($c$ : candidate $k-$itemset; $L_{k-1}$ : frequent($k-1$)-itemsets)
foreach ($k-1$)-subset $s$ of $c$ do
    if $s \notin L_{k-1}$ then
        return TRUE
    else
        return FALSE
    end
end
```

Then, we apply the Apriori algorithm by considering that discretized trajectories correspond to "bags" and geographical cells correspond to "items":

- First the 1-frequent items/cells L_1 set is found, scanning the dataset of bags/trajectories to count number of occurrences of each item/cell. All the individual items/cells that satisfy the minimum support are counted. That is, they are included in at least S trajectories (S corresponds to the *min_sup* parameter of the Apriori algorithm).

- From the elements of L_1, the set C_2 is formed by considering all possible pairs of cells (candidate frequent pairs of cells).
- To find the L_2 set, the algorithm applies a pruning step, considering among the pairs of C_2 only those that have at least S trajectories (2-frequent cells).
- By increasing the size of the cells at each K iteration of Apriori the sets C_K and L_K are formed and C_k is generated from L_{k-1}.
- The algorithm stops at job K when it is no longer possible to find a set of frequent cells of size $K + 1$.

As a result of this step, we have identified the most frequent cell sets in trajectories. It is important to stress the fact that those cell sets may not be necessarily close in geographical terms, and for this reason they could represent disconnected paths.

In the GeoLife case, we used a grid consisting of 1443 square cells (39 horizontally per 37 vertically) which result in cells whose side is 1300 m (see grid in Figure 3) and about half of these are crossed by at least one trajectory.

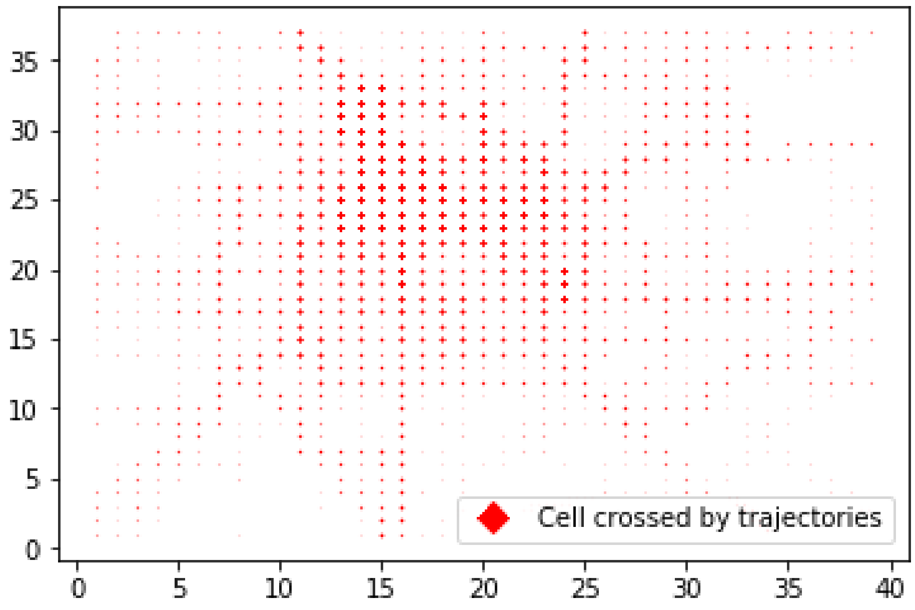

Figure 3. Trajectories mapping into a grid with cells of 1300 m per side.

5.3. Step 1.2

Once we have identified in the previous step the most frequent cell sets in trajectories, we can proceed to process the data by reversing the roles of items and bags in the Apriori algorithm.

In this case, we will consider that trajectories correspond to items and geographical cells correspond to bags. In this way, at the end we will get the most frequent trajectory sets, that is, the sets of trajectories that share at least a given number of cells.

This procedure can be described as follows:

- The algorithm starts by first identifying all the individual trajectories which satisfy the minimum support, that is, those which are at least M cells long (M corresponds to the min_sup parameter of the Apriori algorithm, one of the parameter of the corridor): this is the set L_1.
- From the elements of L_1 with the Join step the set C_2 is formed, correponding to all possible pairs of trajectories (candidate pairs of trajectories). Candidates are the itemsets containing all potentially frequent itemsets.

- To form L_2 the algorithm does the pruning, considering among the pairs of C_2 only those that satisfy the *min_sup* of cells (2-frequent trajectories).
- By increasing the number of trajectories at each K^{th} iteration ($K \geq 2$) of Apriori, it generates candidate K-trajectories C_K from the frequent $(K-1)$-itemsets L_{K-1} of the last iteration. Now only the frequent sets L_{K-1} and candidates C_K reside in memory, whereas other itemsets of previous iterations are discarded.
- The algorithm stops at job K when it is no longer possible to find a set of frequent trajectories of size $K+1$, (that is, when there are no $K+1$ trajectories which cross at least M cells together).

Compared to the previous step, in this step we fixed the desired minimum length of the corridors in terms of cells. Given that, the Apriori algorithm finds sets of trajectories that share at least a minimum number of cells.

In the GeoLife case, this step consists of the application of the Apriori algorithm on a grid of 500 m (102×96 cells), but turning the problem upside down: seeing the cells as baskets and the trajectories as items. By setting the minimum support parameter equal to 51 cells (which make a total length of 51×500 m = 25.5 km) and taking into account that the "populated" cells in the first slot, for example, are 4603 (cells crossed by trajectories), we have obtained a percentage of support greater than that of Step 1.

Since we only considered the last level of the output of the Apriori algorithm, in this step we turned to analyze the subsets of cells that dot not appear in the last result of Apriori: the trajectories which do not participate in frequent itemsets of the longest cells. In addition, to optimize the execution time and vary the results, instead of using all the trajectories each time, we considered a subset of 1000 random trajectories. We chose to build a grid of cells 1 km wide, so the total number of cells are 51 (horizontal) $\times$ 48 (vertical).

We applied Apriori by following this convention: the role of "baskets" is played by trajectories and the role of "itemsets" is played by cells. As for the input of Apriori we chose a minimum support of 50 trajectories on the total number of the random set: the support of a cell is the percentage of trajectories in which that cell occurs.

This support threshold being greater than the one chosen for Step 1 and Step 1.2 leads to an improvement in the execution time of the algorithm. In fact, the output in this case is the tuples of cells shared by at least 50 trajectories and so there are less candidate corridors that contain 50 subtrajectories compared to Step 1 where the minimum support was 15. Moreover, the reduction in the number of the sample of trajectories also contributes to bringing this advantage. By varying the sample several times, different results will be obtained.

6. Experimental Results in the GeoLife Case

6.1. Analysis of Step 1

The application of Apriori results in the list of the K-frequent itemsets (i.e., subsets of frequent cells of cardinality K, the second parameter of the corridors), where frequent means that they are shared by a number of trajectories greater than or equal to the minimum support of trajectories chosen as Apriori parameter in input.

In Step 1 we chose 15 trajectories for each time slot as the minimum support threshold. For example, the total number of trajectories in the first time slot is 5878, so the set percentage of *min_sup* on the total was low but the advantage is to be able to find corridors even for a limited number of trajectories. Setting the minimum number of trajectories, the length of the corridor candidates obtained at each iteration of the Apriori algorithm increases.

In our case, Apriori returns a list of over 4 million K-frequent itemsets of cells, for K from 1 to 21. The largest subset of cells, which is the last iteration of the algorithm, corresponds to a path of about 25 km.

The total run time for Apriori algorithm in every step is defined as (see [21] Suneetha and Krishnamoorti, 2011):

$$\sum_{i=1}^{n}(t_s * m_k + t_c + p_{k+1} * k + 1/2 * t_s * n_k/B)$$

where, t_s be the time cost of a single scan of the database, t_c be the time cost of generating C_{k+1} from L_k, m_k is set to be the amount of itemsets in C_k, the variable p_{k+1} is set to be the amount of itemsets in C_{k+1} and the variable n_k is set to be the amount of itemsets in L_k. B is the number of records in the database and n represents the dimension of the data.

At this point we considered all the trajectories passing from the first cell of the 21-tuple of cells. We deleted all the trajectories that did not pass for the second cell, then we also deleted the trajectories that did not pass for the third cell and so on until the 21st cell. At the end we found 15 trajectories, that are exactly the trajectories passing for all the 21 cells. After verifying that these trajectories cross these cells in the same or in the reverse order and consecutively, we considered the 21-tuple as a candidate corridor. Figure 4 shows the longest candidate corridor obtained in Step 1: the interruption in the upper left part of the graph is probably due to the loss of the GPS signal during the recording.

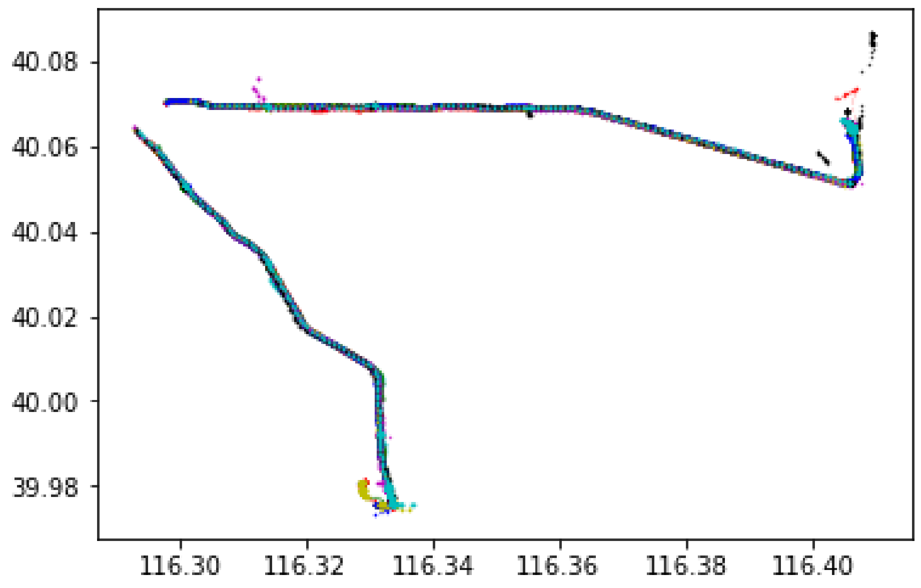

Figure 4. A candidate corridor detected with Step 1, with parameters $M = 19$ cells and $S = 15$ trajectories.

6.2. Analysis of Step 1.2

The output produced by the algorithm consists of 55,109 sets of K elements of frequent trajectories (in the time slot Δt_1), for K from 1 to 14: these trajectories share candidate corridors of over 25 km.

The advantage is that, compared to the previous step, it improves the execution time. Indeed the scanning of the trajectories to check if they are frequent or not is faster because they have few elements (on average they have a length of about 16 cells of 500 m each) and also the support threshold has been raised. In fact, if the min_sup is larger, then the cardinality of the frequent set L_K and therefore also those of C_{K+1}, are much smaller than in Step 1. Moreover, the number of iterations of the algorithm is also lower (it made 14 iterations instead of 21). Another improvement was that a thinner grid which gives greater precision of the road results.

The longest corridors of these 14 trajectories are similar to the ones found in the previous case (Step 1). Among these 14 trajectories, 7 are in common with the 15 obtained in the Step 1. The plot of trajectories crossing 51 cells consists of about 25 km visible in Figure 5, and like the previous case we verify that this sequence is a candidate corridor of parameter $M = 51$ and $S = 14$.

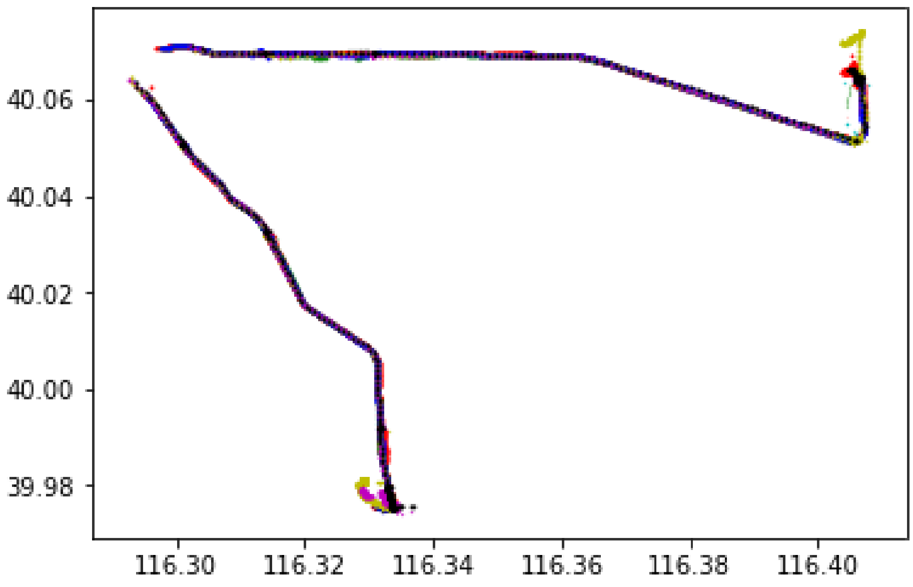

Figure 5. Candidate corridor detected with Step 1.2.

Step 1.2 is a verification of Step 1, also allowed to refine the grid and therefore to have a greater precision (for instance we used a grid with side cells 500 m wide instead of a grid with cells 1300 m per side as in Step 1—see Figure 6).

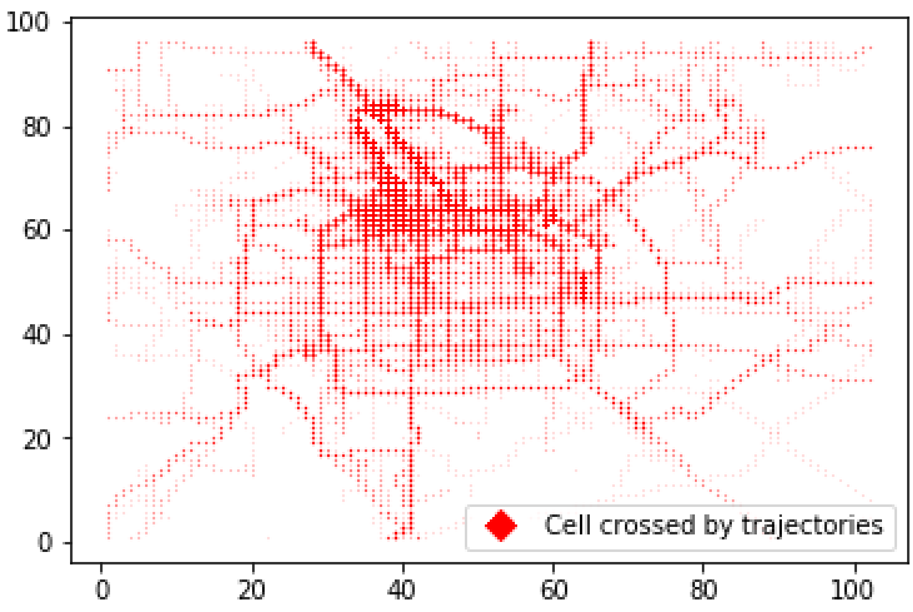

Figure 6. Trajectories mapping into a grid with cells of 500 m per side.

6.3. Step 2

A further advantage is the reduction of execution times significantly. We found two candidate corridors, in the same path shared by a union of 22 trajectories, found in Steps 1 and 1.2.

The grid is out of phase, because 500 (meters) is not a multiple of 1300 (meters), but this gives a greater number of trajectories that are recorded on the same road, with the union of the results obtained.

Analysis of Step 2

The output obtained for data in Δt_1 was a list of frequent itemsets of cells, a table of 1398 K-frequent itemsets, for K from 1 to 9. The longest sets of cells were three 9-frequent cells:

$F = \{1956, 1703, 1804, 1550, 1905, 1906, 1652, 1754, 1855\}$,

$G = \{1956, 1703, 1550, 1905, 1906, 1652, 2007, 1754, 1855\}$ and

$H = \{1956, 1703, 1804, 1905, 1906, 1652, 2007, 1754, 1855\}$.

Compared to the previous steps, the higher trajectory support (min_sup equal to 50) gives us shorter paths of shared cells.

We set $A = F$. Among the 8-frequent itemsets we looked for those that were not a subset of A. If there was a 8-frequent itemset that shared at most one cell with F, then we saved it as a potential corridor. At the end, we added to A all the elements of the 8-frequent itemsets that were not in A and we repeated the procedure with the 7-frequent itemsets. Continuing in this way until the 2-frequent itemsets, we found 7 potential corridors with at least 2 cells.

Doing the same starting from G or H, we found the same potential corridors of parameter M from 2 to 5:

C_1 = $[1601, 1602, 1603, 1604, 1656]$, shared by 53 trajectories;

C_2 = $[1753, 1702, 1703]$, shared by 58 trajectories;

C_3 = $[1548, 1549]$, shared by 58 trajectories;

C_4 = $[1548, 1599]$, shared by 64 trajectories;

C_5 = $[1550, 1551]$, shared by 91 trajectories;

C_6 = $[1600, 1601]$, shared by 74 trajectories;

C_7 = $[1804, 1805]$, shared by 53 trajectories.

The performance of this procedure depends on the number of K- and $(K-1)$-cells and size K for every iteration and on the number of new cells found until level K.

Starting for example from the candidate corridor C_1 obtained, we carried out the filtering which is the final phase of our work. C_1 is not contained in the last result of Apriori, which is formed by the 3 sets F, G, H of 9 cells each. So the length of the candidate corridor is less than 9 kilometers and its plot, zoomed in Figure 7, is given by 52 sub-trajectories that cross diagonally 5 adjacent cells.

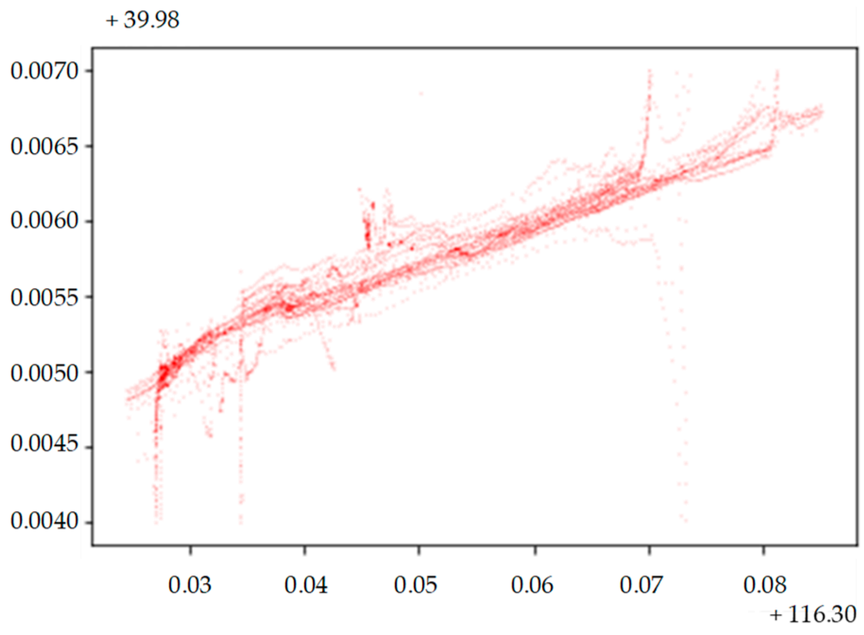

Figure 7. C_1 candidate corridor detected in Step 2, with parameters $M = 5$ cells and $S = 52$ trajectories.

7. Filtering Results

As a result of the approximation of the grid, the previous result does not assure that all sub-trajectories have a similar initial and final part. Then, we have to restrict the search for the corridor to a sub-area, starting from the determined corridor candidate.

Given a radius r set by the user, the Radius Neighbors Graph algorithm constructs a graph in which all the points of a trajectory, seen as nodes of the graph, are connected to all the other points of the other trajectories that are close to it within distance r. The adjacency matrix allows to quickly distinguish which sub-trajectories are in the neighborhood and which sub-trajectories are to be discarded.

Every corridor selected with this filtering phase (applying Radius Neighbors Graph to the cells containing the candidate corridors) is a road segment that can be considered as a "hot" route. The result of the filtering process on candidate corridor detected above is shown in Figure 8.

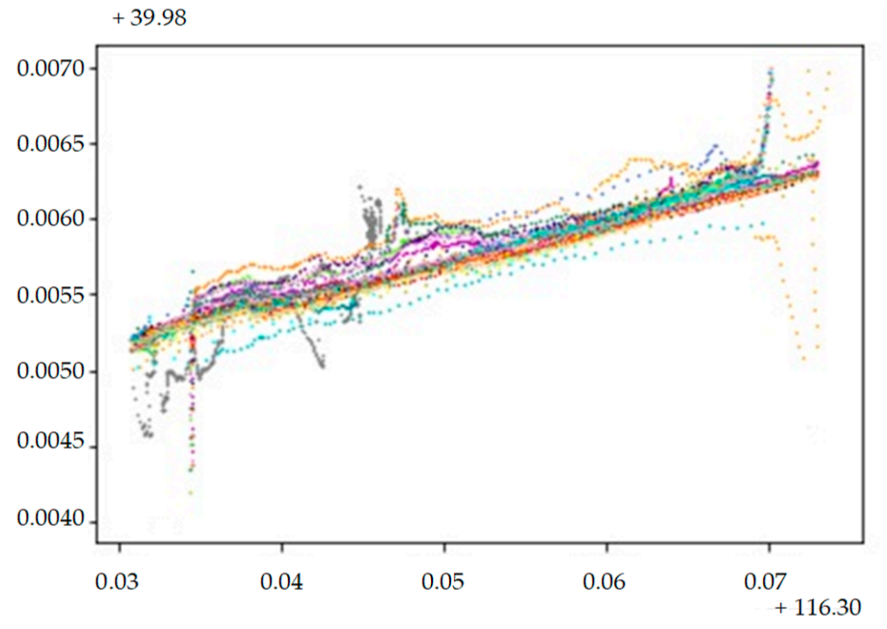

Figure 8. A final corridor obtained by the filtering process, shared by $S = 50$ trajectories and 3.7 km long.

8. Conclusions

The use of spatio-temporal information aims to understand the movement patterns of voluntary users. This information can be used to make an appropriate recommendations such as to know, for example, in which time slot you mostly visit a certain urban area through certain roads.

More specifically, the problem of path detection is important for transport services, which are interested in finding corridors in public transport networks that share multiple routes (e.g., by bicycle, subway, car sharing).

The complexity of naive approaches to detect corridors in GPS datasets in too high in real world cases, and more efficient strategies are necessary. In this paper we have presented a new approach that, based on the application of the Apriori algorithm, allows the filtering of a dataset to obtain a small set of candidate corridors. At the end, the processing of real datasets can be achieved in minutes.

The results of this analysis could have many interesting applications. For instance it would be possible to create a social network to connect people to the same travel location, give travel recommendations or point out the lacking of public transport lines in certain areas to the city council.

Author Contributions: Conceptualization, C.C. and J.V.; methodology, C.C. and J.V.; software, C.C.; validation, C.C.; writing–review and editing, C.C. and J.V.; visualization, C.C.; supervision, J.V. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: This work was partially developed during a visit of the first author to the University of Barcelona. She would like to express her gratitude for providing a fruitful work environment.

Funding: The first author was supported by *PO FSE Sicilia 2020* , Ph.D. grant. Part of the research was supported by RTI2018-095232-B-C21 and 2017 SGR 1742 grants.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Agrawal, R.; Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann, Los Altos, CA, USA, 12–15 September 1994; pp. 478–499.
2. Zheng, Y.; Xie, X.; Ma, W.Y. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.* **2010**, *33*, 32–39.
3. Zygouras, N.; Gunopulos, D. Corridor Learning Using Individual Trajectories. In Proceedings of the 19th IEEE International Conference on Mobile Data Management (MDM), Aalborg, Denmark, 25–28 June 2018; pp. 155–160. [CrossRef]
4. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
5. Berndt, D.J.; Clifford, J. Using Dynamic Time Warping to Find Patterns in Time Series. In *AAAIWS'94, Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*; AAAI Press: Seattle, WA, USA, 1994; pp. 359–370.
6. Rissanen, J. Modeling by Shortest Data Description. *Automatica* **1978**, *14*, 465–471. [CrossRef]
7. Bicocchi, N.; Mamei, M.; Sassi, A.; Zambonelli, F. On Recommending Opportunistic Rides. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3328–3338. [CrossRef]
8. Buchin, K.; Buchin, M.; Gudmundsson, J.; Löffler, M.; Luo, J. Detecting Commuting Patterns by Clustering Subtrajectories. *Int. J. Comput. Geom. Appl.* **2011**, *21*, 253–282. [CrossRef]
9. Rolim, V.B.; Silva, M.R.; Flamarion, F.C.; Filho, C.C.G.F.; Braz, F.J. A Method for Identifying Patterns of Movement of Trajectory Sets by Using the Frequency Distribution of Points. In Proceedings of the GEOProcessing 2017: The Ninth International Conference on Advanced Geographic Information Systems, Applications, and Services, Nice, France, 19–23 March 2017.
10. Devogele, T.; Etienne, L.; Esnault, M.; Lardy, F. Optimized Discrete Fréchet Distance between trajectories. In *BigSpatial'17, Proceedings of the 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 11–19. [CrossRef]
11. Crociani, L.; Vizzari, G.; Gorrini, A.; Bandini, S. Identification and Characterization of Lanes in Pedestrian Flows Through a Clustering Approach. In *AI*IA 2018—Advances in Artificial Intelligence. AI*IA 2018. Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2018; Volume 11298, pp. 71–82. [CrossRef]
12. Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. Multi-Agent Architecture for Point of Interest Detection and Recommendation. In Proceedings of the 20th Workshop From Objects to Agents, Parma, Italy, 26–28 June 2019; CEUR Workshop Proceedings; Volume 2404, pp. 98–104.
13. Zheng, Y.; Zhang, L.; Xie, X.; Ma, W.Y. Mining interesting locations and travel sequences from GPS trajectories. In Proceedings of the International conference on World Wild Web (WWW 2009), Madrid, Spain, 20–24 April 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 791–800. [CrossRef]
14. Kleinberg, J.M. Authoritative Sources in a Hyperlinked Environment. *J. ACM* **1999**, *46*, 604–632. [CrossRef]
15. Hosseinpoor, A.H.; Abbaspour, R.A.; Claramunt, C. A Geometric Framework for Detection of Critical Points in a Trajectory Using Convex Hulls. *ISPRS Int. J. Geo-Inf* **2018**, *7*, 14. [CrossRef]
16. Zheng, Y.; Fu, H.; Xie, X.; Ma, W.Y.; Li, Q. *Geolife GPS Trajectory Dataset—User Guide*; Geolife Gps Trajectories 1.1 ed.; Microsoft.: Redmond, WA, USA. 2011.
17. Zheng, Y.; Li, Q.; Chen, Y.; Xie, X.; Ma, W.Y. Understanding Mobility Based on GPS Data. In Proceedings of the ACM conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea, 21–24 September 2008; Association for Computing Machinery: New York, NY, USA, 2008; pp. 312–321. [CrossRef]
18. Inman, J. *Navigation and Nautical Astronomy, for the Use of British Seamen*; F. & J. Rivington: London, UK, 1849.
19. Sinnott, R.W. Virtues of the Haversine. *Sky Telesc.* **1984**, *68*, 159.

20. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*, 3rd ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2011.
21. Suneetha, K.; Krishnamoorti, R. Web Log Mining using Improved Version of Apriori Algorithm. *Int. J. Comput. Appl.* **2011**, *29*, 23–27. [CrossRef]