Distributed and Parallel Databases
https://doi.org/10.1007/s10619-020-07303-0

Location prediction: a deep spatiotemporal learning from external sensors data

Lívia Almada Cruz[1] · Karine Zeitouni[2] · Ticiana Linhares Coelho da Silva[1] · José Antonio Fernandes de Macedo[1] · José Soares da Silva[1]

Abstract
This paper proposes a deep multi-task learning framework to predict the next location from trajectories that are captured by external sensors (e.g., traffic surveillance cameras, or speed radars). The reported positions in such trajectories are sparse, due to the sparsity of the sensor distribution, and incomplete, because the sensors may fail to register the passage of objects. In this framework, we propose different preprocessing steps to align the trajectories representation and cope with a missing data problem. The multi-task learning approach is based on Recurrent Neural Networks. It utilizes both time and space information in the training phase to learn more meaningful representations, which boosts the learning performance of location prediction. The multi-task learning model, together with the preprocessing step, substantially improves the prediction performance. We conduct several experiments using a real dataset, and they demonstrate the validity of our multi-task learning model in terms of accuracy of 85.20%, which is more than 20% better than using a single-task learning model.

Keywords Location prediction · External sensors trajectories · Multi-task learning · Recurrent neural networks

1 Introduction

The high availability of tracking data brought opportunities for researches and industry to provide new methods to analyze and understand mobility patterns. Among the possible analysis, mobility prediction has gained some attention given its applicability. In general, the mobility prediction problem consists in inferring the next

✉ Lívia Almada Cruz
livia@insightlab.ufc.br

[1] Federal University of Ceará - Insight Data Science Lab, Campus do Pici, Bloco 952, Fortaleza, Ceará, Brazil

[2] University of Versailles, Versailles, France

Published online: 25 June 2020

relevant location of a moving object based on historical information and its most recent tracking. Many applications, such as smart transportation, traffic control, urban planning, and recommendation systems, can benefit from mobility prediction.

Unlike previous works that use GPS, in this work, we predict the movement of objects under the circumstance where their trajectories are captured by external sensors (e.g., traffic surveillance cameras) placed on the road-sides. Each sensor captures and registers the passage of moving objects. Assuming that each record contains enough information to identify the associated moving object uniquely, for instance, identifying vehicles by their plates, it is possible to derive from them (i.e., from the recorded information by different sensors) the trajectories of the moving objects as a sequence of sensor positions.

The location prediction from such kind of data is useful, especially for government agencies where GPS data is not available, for instance, police patrol applications to track criminals or traffic offenders; and traffic surveillance applications, among other services.

In [2] three levels of mobility prediction are considered: (1) object position; (2) path prediction; and (3) next place prediction. In levels 1 and 2, usually, the models learned from raw trajectories obtained by Global Positioning System (GPS) devices, and their predictions consider the movement of the object. Level 3 predicts stops rather than movements, usually by learning from sequences of points of interest or georeferenced events. While levels 1 and 2 perform fine granularity predictions, level 3 yields high-level predictions.

This work is in between level 1 (object position prediction) and level 2 (movement and path prediction). But different from what is usually found at level 1, these positions are predefined and more sparse than the case of continuous positioning with GPS. As in level 2, the positions are part of the object path, thus considering the road network constraints may improve predictions.

It is worth noting that in our context, object movements are constrained by a road network, and their observation only occurs at fixed (predefined) positions (i.e., sensors' location) on the road network. This assumption turns our prediction problem different from (i) the ones that consider GPS trajectories, which occur at any spatial location as [17] and [6] to name a few; and (ii) the next (stop) location prediction, usually applied to points of interest (PoIs) or event places as [11, 21, 9] and [20], since we predict movement rather than stops. So, despite many works on the literature in mobility prediction, to the best of our knowledge, none of those works have studied mobility prediction for trajectories based on external sensor data. We call this problem *External Sensor Trajectory Prediction* (EST Prediction for short).

Recurrent Neural Networks (RNN) are one of the most effective approaches to predict sequenced data as they can learn the relationship between consecutive values in a sequence. Recently, applying RNN to predict the next stop has demonstrated the potential of these approaches to capture the complexity of mobility data. The works at level 3, usually apply RNN to trajectories modeled as a sequence of labels, each label representing a location (region or point of interest) where the moving object has stopped. We can also model our problem as a sequence of sensors' labels, allowing us to apply RNN to EST prediction. Thus, we aim to investigate the following questions: (1) Can RNN adapt to incompleteness and sparsity of EST? (2) How

different data imputation strategies impact RNN models? (3) How to access the quality of predictions given the fact that the ground truth could be missed and thus unknown?

In this paper we tackle the problem introduced above and provide the following contributions:

1. An in-depth study to characterize the challenges to deal with trajectories derived from external sensor data;
2. Methods for coping with the problem of the incompleteness and sparsity of this data and their comparisons;
3. A recurrent multi-task learning approach to utilize both temporal and spatial information in the training phase to jointly learn more meaningful representations of time and space, which boosts the performance to EST prediction;
4. An extensive experimental evaluation over a real-world dataset, where we assess the validity of our proposal in terms of quality of results.

The remainder of this paper is organized as follows. Section 2 presents the problem statement. Next, we discuss some related works in Sect. 3. Section 4 highlights the imperfection of ESTs. Section 5 presents the proposed solution. In Sect. 7, we discuss the results of our experimental evaluation. Finally, in Sect. 8 we conclude the paper and discuss the future work.

2 Problem statement

Given a set of external sensors placed on road-sides which register the passage of moving objects and assuming it is possible to uniquely identify the moving objects (for example, by recording the vehicle plate), we call external sensor trajectories (EST) the ones derived from the observations captured by these sensors. In this work, we consider the sensors are imperfect and thus produce incomplete trajectories with missing data. Figure 1 presents a real trajectory of a bus collected from external sensors in one day; in such trajectory, the bus travels along the same route many times. The points (in green) represent the sensors, and sensors are connected by the lines (in red) when they appear in sequence on the same trajectory. We can observe that some sensors are skipped (not connected to the next expected sensor) despite the fact that the bus had done the same route. This is because, in such cases,

Fig. 1 Example of trajectory with missing sensors. The points represent the sensors and sensors are connected by the lines when they appear in sequence on the same trajectory

some sensors failed to capture the passage of the bus. Hereafter, we show some definitions to support our problem statement.

Definition 1 (*External Sensor Observation*) When an external sensor (ES) s registers the passage of a moving object m at a time t, it produces a tuple $o = (m, s, t)$. However, the sensors may fail to capture the passage of a moving object.

Definition 2 (*External Sensor Trajectory*) Let O be the set of observations generated by a set of sensors S. Let $O[m] \subset O$ be the set of observations related to the moving object m. Then, we define an *external sensor trajectory (EST)* of a moving object m that can be extracted from $O[m]$, as a the sequence of observations $es\_traj_m = \langle o_1, o_2, \ldots, o_j \rangle$ such that $\forall i, 1 \leq i \leq j, o_i \in O[m]$ and $t(o_i) \leq t(o_j)$.

In the experiments, we restrict trajectories to a period of 24 hours. Now, we are ready to present the next sensor prediction problem.

Problem definition [External Sensor Trajectory Prediction] Let G be a road network, S the set of external sensors deployed over G, O the set of historical observations produced by S, M the set of moving objects referred by the observations in O, and T_{EST} the set of historical trajectories derived from O describing the movement behaviors of the objects in M. Given the last w most recent observations in time and space of a moving object $m \in M$ produced by O, the problem consists of predicting the next sensor to be visited by m.

2.1 Characteristics of external sensor trajectories

Different from classification problems with many classes, to learn from trajectories we have to consider complex transitions patterns and time-dependence. As the sensors have spatial relationships among them, the proximity of the predicted value to the actual registered value is also important. Furthermore, the trajectories obtained from external sensors have several particularities that provide new opportunities while raising some challenges.

1. **Huge data** Sensors continuously capture a massive number of observations per day. The application needs to ingest multiple sensors data streams, compute the last observations of the moving object and make predictions online. The complexity to manage these tasks increases with the number of vehicles.
2. **Exhaustive types of trajectories** Moving objects are not restricted to a specific fleet of vehicles as usual in GPS data collection. Commuters, a fleet of taxis or buses, deliveries, etc., are all tracked by the road-side sensors. Thus, trajectories can have very different patterns. For example, commuters usually have temporal repetitively behavior like go to work and return home every week-day; on the other hand, taxis may have very different behaviors that depend on their passengers.

3. **Sparsity** The sensors are located in fixed positions, usually only on the main roads of the city. The complete tracking of moving objects is not available, and the reported positions per trajectory are very sparse in space and time. The sparsity turns the trajectories shorter and with fewer data points, making the prediction harder.
4. **Incompleteness and uncertainty** Sensors may fail to capture the passage of a vehicle, producing incomplete trajectories. It is not self-evident when one observation is not in the dataset because the sensor failed (most of the time when its battery discharges), or if it is because the object did not pass by the sensor. Incomplete trajectories bias the prediction since the model learns from wrong data. For example, moving objects which executed the same path may have a different sequence of locations (as explained before and exemplified in Fig. 1).

Finally, we assume that EST is constrained by the topology of the network; this may help to compensate the sparsity and incompleteness of the trajectory observations.

3 Related work

In the literature, there exists a vast amount of works dealing with the problem of location prediction and mobility learning. In the context of this work, we give an overview of two main classes of works that are of interest: the first proposes prediction models for location prediction, while the second class employs spatiotemporal multi-task learning.

3.1 Prediction models for location prediction

MyWay [18] is a system that predicts the position of the next movement of an object. MyWay uses the spatial match of trajectories with a set of profiles obtained by cluster the raw trajectories. The paper [17] proposed TPRED, an approach based on a Prefix-Tree to predict the next stop and when the user will leave the current location. TPRED mines the stops from raw trajectories, based on the time the user spends in a place. These works consider trajectories captured by GPS, and cannot be applied directly for sensor trajectory prediction. TPRED needs dense trajectories to mining stops. Although, MyWay forecasts the next movement, the similarity measure to cluster and match trajectories is not appropriated for very sparse trajectories.

The work of [16] predicts the next stop of indoor trajectories based on groups of users who share some characteristics (like gender and age). From the sequential rules, the method estimates the probability of visiting a specific location given the recent movement of the user and his group. GMove [22] uses spatial and temporal information, and geo-tagged text, extracted from social media, to predict the next stop of one user based on a group-level mobility model. GMove model is an ensemble of Hidden Markov Models (HMM); each HMM is built using a group of users that share similar movements. These works consider extra information not available for sensor trajectories.

The work of [1] proposed a Dynamic Bayesian Network (DBN) to predict the next location using data from call details records (CDR) taking into account the sparsity of this data. To solve the sparsity problem, their model incorporates patterns of users with similar trajectories. Besides the sparsity of CDR data, it is different from external sensor data since each location in CDR data covers an entire region, not one exact position.

The paper [11] proposes Seq2Seq, an attention-and an encoder-decoder LSTM(Long Short-Term Memory)-based learning approach to model semantic trajectories and predict the next location. LSTM extends RNNs and has been extensively used in sequence to sequence tasks since it processes variable-length input and can allow highly non-trivial long-distance dependencies to be easily learned. The attention mechanism in Seq2Seq gives the ability to store inputs and previous states for a long period of time, as well as to delete them when necessary. Seq2Seq accepts both raw GPS data as well as pure semantic location to train the model.

RNN was also successfully applied in Natural Language Processing to capture dependence between terms of sequences [15]. More recently, applying Recurrent Neural Networks (RNN) to location prediction has demonstrated the potential of these approaches to capture the complexity of mobility data. A Spatial-Temporal Recurrent Neural Network (ST-RNN) was proposed in [14] to predict the next location dealing with continuous values in spatial and temporal contexts. SERM [21] is a spatial-temporal model to predict the next stop using semantic trajectories from Social Media. The model of SERM uses spatiotemporal information and a bag-of-words obtained from check-ins, these features are mapped into a one-hot vector, the embedding layers are responsible for reducing the dimension, the features are concatenated, and the result is the input of the RNN. Finally, a vector with user preferences is summed to the output of RNN. TA-TEM [23] is a recommendation system for the next location, which predicts the next stop by learning from the sequence of check-ins and considering the preferences of each user in different levels of time and general user preferences. DeepMove [6] uses attention recurrent networks to predict the stop on the next time window. DeepMove uses a module to capture the mobility regularity and output the most related context vector to the current trajectory. DeepMove combines the context vector and the output of RNN to make the predictions.

To the best of our knowledge, only the approaches of [19] and [20]solve the trajectory prediction problem under road-network restrictions. Both of them are based on Recurrent Neural Networks. However, these works consider dense GPS trajectories as input. For [20], a trajectory is a sequence of edges in the road network. The problem is to predict the next road segment of a moving object, given its most recent past trajectory. The main idea of their approach is to change the activation function by applying a mask that nullifies illegal transitions based on the adjacency matrix of the road network. The authors in [19] apply a feature extraction that maps GPS trajectories in referenced points on the road network; then they use LSTM to make predictions. There are also works, like [10] which predict movement without learning from the historical data but by applying predictive query.

3.2 Spatiotemporal multi-task learning of mobility

Multi-task learning refers to models that learn multiple related tasks simultaneous to improve the overall performance [8].

The paper [9] proposes a semi-markov continuous-time Bayesian network model that jointly learns the user activity timing and location sequences based on incomplete activity participation information found in check-in data. The model predicts the next activity participated by an individual, the transition time from the current activity to the next one, and his/her next location given the next activity type and the current location.

In [13], the authors propose a model to predict travel time based on a multi-task learning framework to jointly learn the main task as well as other auxiliary tasks. In their model, different tasks share most part of parameters except the output layer, that is unique for each task. They used pre-trained learned representations for links on road-network based on unsupervised graph embedding techniques. For spatial and temporal features they create a spatial graph and a temporal graph, respectively. For the spatial grid representation, they used the concatenation of latitude and longitude embedding and for temporal representation, they used the concatenation of the time in the day and day of the week. Then, the model learns meaningful representations using path information available on the training period. The approach proposed in [13] predicts travel-time, but not the next location.

The paper [8] proposes the multi-task learning framework TITAN. The model is trained using data from incident records. For each incident occurrence, the traffic speed readings of minutes before and after the occurrence are given by the model, and it predicts the future impact of this given incident in terms of the temporal duration of this traffic incident. TITAN also combines the information of the road network connectivity in its model design since an incident occurrence could not only cause traffic pattern change at local road segments but also spreads the pattern change on other adjacent roads that have a close spatial correlation. The problem addressed by [8] is different from the location prediction targeted in this paper. However, it shows the importance of multi-task learning for spatiotemporal features.

4 Data sparsity and incompleteness

In this section, we study the sparsity and incompleteness of sensors data. We analyzed 272 sensors from Fortaleza city in September 2017, these are used in a real application. Initially, we have 22,338,916 observations at all. For each day, we computed the trajectories from the sequences of observations of the same moving object. After that, we partitioned the trajectories according to a pre-defined condition (we explain in the following), and then we discard the shortest trajectories. The reason is to avoid train the model on short sequences, which would bias the prediction since we use a sliding window of k observations to predict the next one. Furthermore, a minimum number of observations is needed to train the models. We chose six as the minimum length for this analysis. This means that each moving object trajectory passed by at least six sensors.

Let $\mu_{s_k,s_{k+1}}$ be the average and $\sigma_{s_k,s_{k+1}}$ be the standard deviation of the displacement time between the sensors s_k and s_{k+1}. Let the trajectory $\langle o_1, o_2, \ldots o_{n-1}, o_n \rangle$, consider we split such trajectory into two sequences: $\langle o_1, o_2, \ldots o_i \rangle$ and $\langle o_{i+1}, \ldots, o_n \rangle$ when $(t_{i+1} - t_i) > (\mu_{s_k,s_{k+1}} + 2\sigma_{s_k,s_{k+1}})$. This condition captures an unusual behavior on the path from s_k to s_{k+1}. Then, we filtered out the trajectories with less than six observations. In the end, only 825,218 trajectories have more than six observations for 236,517 distinct vehicles. This means an average of 3,48 trajectories per vehicle.

From this point, we define a transition $s_k \rightarrow s_{k+1}$ as two sensors s_k and s_{k+1} that appear consecutively on the same trajectory. In this case, the transition starts at s_k. In the following, we show the analysis we performed using sensors observations, trajectories transitions and by applying data imputation.

4.1 Analysis of sensor transitions

In this analysis, we investigate the sparsity of the sensors by analyzing the observed transitions between them. We first computed the road network distance between the sensors that appear in a transition for the set of trajectories analyzed. Figure 2a shows the Kernel Density Estimation (KDE) of the road distances and Fig. 2b shows the KDE of road distance between the other four closest sensors from each sensor (we chose the value four empirically). The road network distances between sensors in the trajectory transitions are greater than 1765.88 m for more than 50% of all transitions. Some of these distances are greater than 14 km. However, 50% of the distances between one sensor and its four closest sensors along the road network are less than 755.61 m. It is likely that the moving objects pass by close sensors. Assuming that, we observe that the sensors frequently fail to capture the passage of the moving objects. Also, many transitions connect sensors that are far away from each other, which means that maybe some observations are missing.

The number of transitions from one sensor to another is in total 7,333,122. If the sensors were uniformly distributed, each sensor should have, on average, 26,886 transitions (i.e., #transitions/#sensors). Table 1 presents the percentiles of the number of transitions for each sensor. We observed that 30% of the sensors do not appear

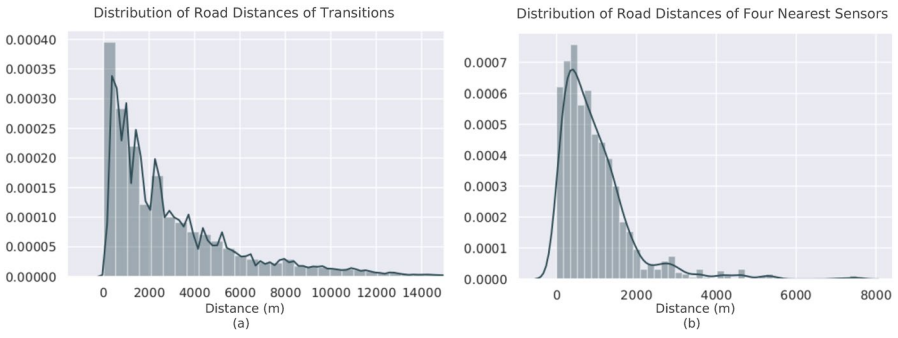

Fig. 2 **a** KDE of road distances between transitions on the trajectories. **b** KDE of road distances between the 4 nearest sensors

Table 1 Percentiles and maximum values for the numbers of transitions starting at each sensor for all sensors

PCTL	10	30	50	70	90	Max
Value	0	0	9496	25,616	59,189	247,768

in any transition, they might be out of service. And 70% of sensors appear in less than 25,616 transitions, this number is only 0.35% (i.e., 70th percentile/#transitions) of all transitions.

We made the same analysis by removing the sensors without any departure transition as presented in Table 2. We can observe that some sensors have more transitions than the average. Half of these sensors have less than the average number of transitions. The reason for this could be the quality of these sensors since sensors with lower quality do not capture all possible observations. Another reason could be the distribution of traffic in the city. Sensors placed in regions with lower traffic have few observations. However, we know that these sensors are on the main roads of the city, which usually are roads with intense traffic.

4.2 Analysis with data imputation

In this section, we aim at estimating the amount of missing data in the data generated by the sensors. Missing Data imputation requires to fill the values of features that are unknown (or missing) with values that ensure a desired degree of reliability. Usually, the tuples and features having missing values are given for data imputation. In the case of the sensors, the missing positions are not known. In fact, when two sensors appear consecutively in a trajectory, this trajectory may be incomplete and there exists a missing sensor, or the vehicle did not pass by any road containing sensors. That means we also have uncertainty as to whether they are missing values, or not.

Based on the hypotheses that the drivers usually prefer to choose the shortest paths [10], to deal with missing values we take into account the sensors in the shortest path between two consecutive sensors. Our strategy assumes that, when s_i, and s_j appear consecutively in a trajectory, all sensors on the fastest path from s_i to s_j are potentially missing sensors. In that way, we complete the transitions according to the shortest path between the sensors. We call *completed transitions* the result of this process, formally defined as follows.

Table 2 Percentiles and maximum value for the numbers of transitions starting at each sensor with transitions

PCTL	10	30	50	70	90	Max
Value	6906	9869	22,881	39,064	77,635	247,768

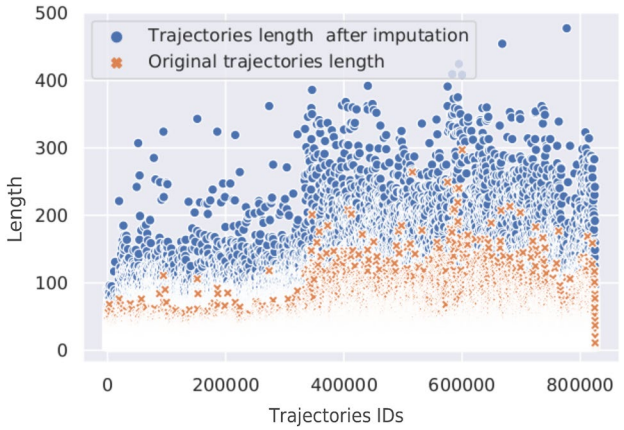

Fig. 3 Original lengths and lengths after imputation of each trajectory in number of sensors

Definition 3 (*Completed Transition*) Let be $p_{i,j} = \langle e_{ij_1}, e_{ij_2}, \ldots, e_{ij_n} \rangle$ a sequence of distinct and connected edges of the underline road network representing the shortest path from s_i to s_j. Let $S[p_{i,j}] \subseteq S$ be the subset of sensors such that $s_k \in S[p_{i,j}]$ iff the path $p_{i,j}$ passes by s_k. We define $\prec$ as a total order on $S[p_{i,j}]$ where $s_k \prec s_{k+1}$ iff $dist(s_i, s_k) < dist(s_i, s_{k+1})$, for all $s_k, s_{k+1} \in S[p_{s_i,s_j}]$, where $dist(s_i, s_k)$ is the length of the shortest path $p_{i,j}$. The *completed transition* $c\_tran_{i,j}$ from sensors s_i to s_j is the sequence of sensors $\langle s_i = s_{k_1}, s_{k_2}, \ldots, s_{k_{m-1}}, s_{k_m} = s_j \rangle$, where $S[p_{i,j}] = \bigcup_{l=1}^{m} \{s_{k_l}\}$, with $s_{k_i} \prec s_{k_{i+1}}$.

Basically, the completed transition $c\_tran_{i,j}$ is the sequence of sensors which the path $p_{i,j}$ visits, ordered by the increasing distance from s_i. Note that since the road network is a directed graph, $c\_tran_{i,j}$ may be completely different of $c\_tran_{j,i}$.

From the transitions that appear in the dataset of trajectories, we estimate all the completed transitions. Table 3 shows the percentiles of trajectories before and after imputation process; and the ratio between the new and the original values (length with Imputation/original length). The length is computed in the number of sensors.

From Table 3 we can see that, in general, the data imputation doubles the original trajectory length, and that the rate increases for longer trajectories.For instance, 50% of trajectories had a length of fewer than eight sensors, but after the imputation process, the median length becomes 16.

Table 3 Percentiles and mean of trajectories lengths

Percentile	10	30	50	70	90
Original	6	7	8	10	16
With imputation	9	12	16	22	37
Ratio	1.5	1.71	2	2.2	2.31

Figure 3 presents the lengths of all trajectories before and after data imputation, which shows clearly the difference.

We analyze the number of imputed sensors considering the transitions that appear in the data, to understand how the possible missing values are distributed over transitions. From 7,164,653 transitions in total, 46.5% of them had some imputed data. For these transitions, the average number of imputed sensors is 2.64, and the median is 2. We can conclude that more than 25% of all transitions have more than two sensors imputed, which is a significant number considering the distances between sensors. These results indicate that the imputation process effectively complements the sensor trajectories.

5 EST prediction framework

In this section, we present our solution to the EST prediction problem. Our strategy consists of three main steps. The first one is the Pre-processing, which aims at pre-computing the sensor pairwise distances and simplifying the representation of trajectories. The second is the Learning step which includes data imputation and the model training. And finally, the Prediction itself. How these steps interact in the proposed model architecture is presented in the next section. Hereafter we explain each step.

5.1 Pre-processing

The main reason for the prediction errors is the frequent failure of some sensors in capturing the passage of moving objects. To compensate this, the data imputation process fills the object trajectory by sensors (assumed missing values) located in the shortest path between the reported sensors (the actual observations). To ease this data imputation, in this step, we pre-compute once for all the completed transitions (Definition 3) between every pair of distinct sensors and maintain them in a data structure *CP* that maps each pair of sensors (s_i, s_j) to its completed transition $c\_tran_{i,j}$, along with the distances between sensors in the road network.

Additionally, to minimize the impact of skew of the locations of the sensors, and harmonize the representation of trajectories, we group sensors which belong to the same road-network region. The groups form corridors since they tend to chain close sensors according to the road distance, as presented in Fig. 4. Once we have the clusters, on the training step, we can replace the sensors by their correspondent clusters. The idea is to get a unique representative for dense sensors to simplify and harmonize the trajectory representation. This is analogous to stemming in text mining. To that end, we define the distance $\overline{d}(s_i, s_j)$ between two sensors s_i and s_j as $\overline{d}(s_i, s_j) = min\{d_R(s_i, s_j), d_R(s_j, s_i)\}$, where $d_R(s_i, s_j)$ is the road-network distance from s_i to s_j. Then, we apply the DBSCAN [5] algorithm to cluster the sensors using the distance $\overline{d}$. We set the *MinPts* parameter in DBSCAN algorithm, to be equal to one since we want the cluster to chain close sensors by density, and tune the parameter *Epsilon* empirically.

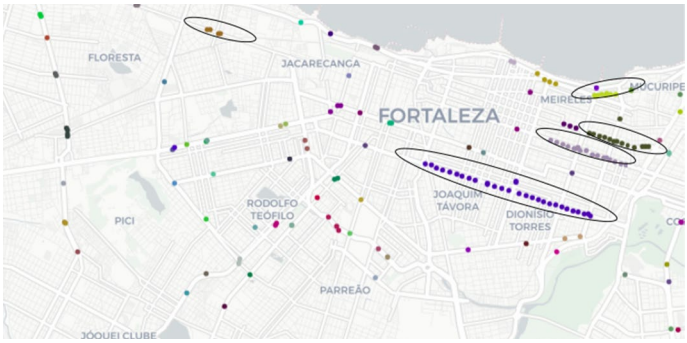

Fig. 4 Some external sensors in Fortaleza city. Inside ellipses, some corridors identified by the clustering

5.2 Learning process

In the learning step we perform the data imputation and the training process itself. Our approach receives the last k observations of a moving object, and the data imputation step prepares the input data to train the model. Our strategy for data imputation maintains all the completed transitions and the road distances between every ordered pair of distinct sensors. We use the completed transitions to enrich the most recent sub-trajectory received to increase the amount of information available for learning. As a result, we have a new sequence of sensors as input. Obviously, the uncertainty remains since the completed transition is only a probable sequence of missing sensors along the shortest path, while the real trajectory may use a less probable route.

Considering a window of $k = |w|$ last observations of a moving object m given by $sub\_tr = \langle o_i, o_{i+1}, \cdots o_{i+k-1} \rangle$, where $o_j = (m, s_{m,j}, t_j)$, $i \leq j \leq i + k - 1$, since we want to predict $s_{m,i+k}$ and as this value is known on the training step, the imputation process is applied in different ways for training and prediction phases.

We consider two different strategies for imputation in the training phase: ***full imputation*** and ***next value imputation***. For both of them, we receive $sub\_tr = \langle o_i, o_{i+1}, \ldots o_{i+k-1} \rangle$ and the target prediction o_{i+k}. The *full imputation* strategy performs the imputation for all transitions, including the transition from the last observation on the trajectory to the location to be predicted (which is known in the training phase) by completing all the transitions from o_j to o_{j+1}, for all j where $i \leq j \leq i + k$, with sensors labels using the completed transitions $c\_tran_{s_{m,j}, s_{m,j+1}}$. In the *next value imputation*, we assume that the real target value is the first imputed sensor that appears on the completed transition $c\_tran_{s_{m,i+k-1}, s_{m,i+k}}$, that is the next value just after the last observation. Thus, we complete all transitions from o_j to o_{j+1}, for all j where $i \leq j \leq i + k - 1$ and change the target.

Figure 5 exemplifies both approaches for data imputation. The light gray circles are the originally observed sensors used as input for prediction, the dark grays circles are the original target predictions, and the dashed squares are the imputed values. Full imputation maintains the original target, while next-value imputation drops it off and replaces it by the first imputed vector just after the last observation O_5. Our

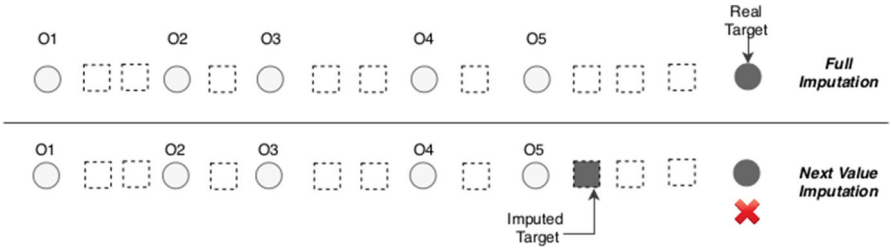

Fig. 5 Data imputation strategies for training. Circles are real observations captured by the sensors. Dashed squares are imputed data. The dark circle is the next sensor to be predicted

idea for the full imputation strategy is to force the model to learn the whole path until the observed value. On the other hand, using next-value imputation the model learns the next step of the path. After imputation, we replace the sensor identifier by the cluster identifier to which the sensor belongs.

5.3 Data imputation integration on prediction

Data imputation on the prediction phase differs from the one on the training phase since the value of o_{i+k} is unknown. Thus, in this phase, we apply the imputation by completing only the values until o_{i+k-1}. Any information about the shortest path from the last observed value to the target prediction is not accessible to the model.

After the data imputation process, the new sequences may have an arbitrary length. As the model receives vectors with fixed lengths, we defined a parameter w_{max} as the maximum size of the resulting sequence after the data imputation. In case the new sequence length is greater than w_{max}, we discard the values at the beginning of the imputed sequence until its length arrived at w_{max}.

6 RNN schema for location prediction

This paper extends our previous work [4] by introducing a multi-task approach to jointly learn the next sensor and the time slot when the vehicle will reach the next sensor. The general RNN schema proposed in [4] was based on the SERM approach [21]. However, the SERM model was designed for semantic trajectories derived from check-ins; it utilizes a bag of words retrieved from Social Media as a semantic feature. Also, SERM estimates users' preferences and adds this information to the output of the RNN to give higher weights to the most preferred locations.

In this work, the RNN model receives a window of the w last observations. Each observation has two features: the spatial feature, that corresponds to the sensor label; and the temporal feature that is the time slot in a day when the external sensor observation occured. For instance, the day time can be sliced by intervals of 15 min. Our approach represents each feature using the one-hot vecor representation. A one hot encoding is a representation of categorical variables as binary vectors. Each value is represented as a binary vector that is all zero values except the index of the element

to be represented, which is marked as 1. So given one observation $o_k = (m, s, t)$ at step k, the spatial feature is a vector $l_k = \left[l_{k_0}, l_{k_1}, \ldots, l_{k_n}\right]$, where n is the number of sensors and $l_{k_i} = 1$ if and only if the object was observed at sensor $s = k_i$, such as $s < n$, otherwise $l_{k_i} = 0$. The temporal feature is a vector $t_k = \left[t_{k_0}, t_{k_1}, \ldots, t_{k_m}\right]$, where m is the number of time slots and $t_{k_i} = 1$ if and only if the time slot of $t = k_i$, otherwise $t_{k_i} = 0$.

Before discussing the proposed architecture, it is worth to mention that our goal is to predict the next location, however during the modeling process, we need to consider multiple factors for the next location. One of them is that usually, the trajectories constrained to the road network exhibit strong spatiotemporal regularity—the current location and time slot can have a strong influence in deciding the next location. We believe that a multi-task RNN that learns the time and space features, improves the sequential prediction for location. We confirmed experimentally that compared to the previous results [4], using the time and space information in the training phase our approach learns more meaningful representations, which boosts the learning performance of EST prediction.

Figure 6 shows the proposed architecture, described hereafter:

(1) An embedding layer, that applies a linear transformation on each one of the input vectors, the spatial and the temporal features, separately. This linear transformation is responsible for reducing the dimensions of input vectors while maintaining the proximity of vectors with similar patterns in the new space. We employ an embedding layer to transform the representation of the spatial feature, and another embedding layer to transform the representation of the temporal feature.
(2) A layer to concatenate the output of the embedding layers in order to get a unique input feature vector for the next layer.
(3) A recurrent layer to learn the complex patterns from sequences. The proposed solution uses two recurrent layers in parallel, each one receives the output of the concatenation layer.
(4) Each recurrent layer is connected to a fully connected layer with Softmax as activation function, which converts the result of the recurrent layer into the set

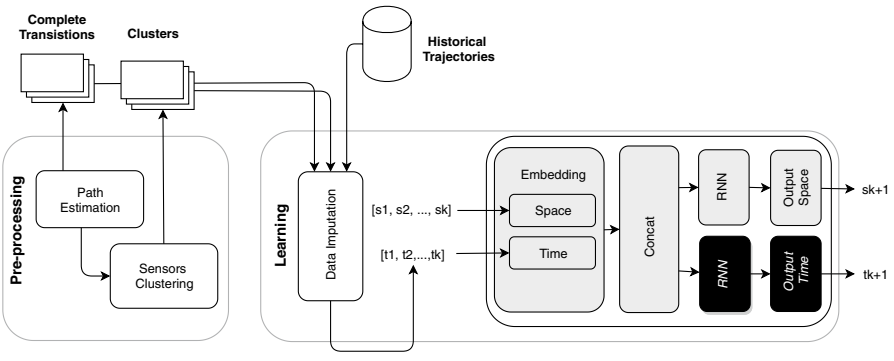

Fig. 6 Architecture of our solution for EST prediction

of probabilities to be assigned to each class label. The proposed model has two dense connected layers, one to predict the location and another one to predict the time slot. Here the number of classes is the number of sensors (or clusters) in the output space layer, and equally, to the number of time slots in the output time layer.

This multi-task learning architecture differs from our previous work [4] by including a new recurrent layer connected and a densely connected layer that learns the time slot. These layers appear in Fig. 6 in the black boxes. From now, we also refer to our strategies proposed in [4] as single-task strategies.

In the proposed architecture, the embedding layers are shared with both predictors, their weights are trained based on the mean of the loss function of output space and output time. Even though our goal is not to predict the next slot of time, our proposed architecture has an output layer for time. We observe an improvement in the experiments in terms of the next location prediction accuracy because the weights are better adjusted based on the loss function of space and time outputs. We believe this strategy will tune the embedding and result in a better representation of space and time features. We confirm our hypothesis since the experiments of the new results leveraging this multi-task learning approach reached 85.20% of accuracy, more than 20% of the best strategy from the previous results [4].

7 Experimental evaluation

In this section, we present the experimental evaluation conducted to assess our multi-task learning approach considering the clustering and the imputation processing.

Baselines As this approach is the continuation of our previous work in [4] which employs a pure RNN scheme, and since there are no other competitors in our context as emphasized in Sect. 3, we use the models proposed in [4] as the baselines: Basic (B)—the pure RNN scheme; Basic with Clustering (BC)—the basic RNN with the clustering strategy; Next Value Imputation (NI) and Full Imputation (FI)—RNN with the respective imputation strategy; Next Value Imputation with Clustering (NIC) and Full Imputation with Clustering (FIC)—RNN with the respective imputation strategy and clustering. It is worth to mention that B, BC, NI, FI, NIC, and FIC are single-task learning models. Furthermore, other baselines are some basic approaches for sequence prediction, considering only the sequence of labels, here we use the First-order Markov model (1st-Markov) and 5th-order Markov model (5th-Markov). We also compared with a multi-task version of the Markov models (MT-1st-Markov and MT-5th-Markov), that considers each possible combination of location label and time-slot as a state in the Markov model. Our multi-task learning approach has some variants, called here: Multi-Task Basic (MT-B) - the basic multi-task schema; Multi-Task with Clustering (MT-C); Multi-Task with Next Imputation (MT-NI); Multi-Task with Full Imputation (MT-FI); Multi-Task with Next Value Imputation and Clustering (MT-NIC) and Multi-Task with Full Imputation with Clustering (MT-FIC). It is worth to mention that the RNN models

were implemented using [3]. For the experiments with Markov Models, we used the implementation available in [7].

Dataset For the purpose of experimental evaluation, we consider the dataset collected from a real application of traffic monitoring, described in Sect. 4.

Experimental methodology In our experiments, we used $w = 5$ sized sliding window (this justifies we used 5th-order Markov Model); the maximum size after imputation was $w_{max} = 10$; and the time interval is 30 min, which means that the temporal feature is a vector with 48 dimensions (slots per day). We fixed the time embedding and the spatial embedding with size 32 and the user embedding with size 64. We use Adam [12] as the optimizer, Cross-entropy as loss function and Softmax as activation function on the last layer. We evaluate all the metrics using holdout 80–20. We executed each experiment 3 times and computed the average. For the models that use a clustering-based strategy, we vary the value of *eps* in the range (1, 2, 3, 4, 5) km.

Metrics of evaluation The methods were evaluated concerning two metrics: *accuracy* and *closeness error*. The accuracy is the percentage of correct predictions. The closeness error is a measure of the proximity on the road network (given by the road distance) from the observed value to the predicted one. The closeness is a complementary way to estimate the quality of the models, which helps to understand if the predicted sensors are close or not from the expected sensors. For the clustering-based approaches, the closeness error is defined as the maximum road distance between the sensors in the predicted cluster and the observed cluster. The maximum distance intra-clusters is used to not benefit the clustered approaches in comparison to the other methods.

7.1 Evaluation of accuracy

In this section, we study how the single and the multi-task learning models perform in terms of accuracy. Table 4 presents a comparison of the accuracy for the all evaluated models. For the clustering-based strategies, we report the accuracy for *eps* value with the best-achieved result.

Table 4 Comparison of accuracy between the single-task models and the multi-task models, the best results for single-task and multi-task models are in bold letters

Single-task models	Accuracy (%)	Multi-task models	Accuracy (%)
1st-Markov	45.08	MT-1st-Markov	45.24
5th-Markov	42.71	MT-5th-Markov	39.81
B	48.07	MT-B	46.71
BC	47.20	MT-BC	48.49
NI	32.26	MT-NI	32.48
FI	41.28	MT-FI	73.33
NIC	**65.84**	MT-NIC	65.56
FIC	41.14	MT-FIC	**85.20**

Considering the single-task learning models, we see that in general, the RNN based models outperform the Markov models. We believe the RNN based models are more resilient to the noise introduced by the missing sensors than the Markov model. NIC model reached the best accuracy, with 65.84%, followed by the basic model with a single RNN (B) with 48.07%. However, it is worth mentioning that NI reached lower accuracy, only 32.26%. This means that only clustering the sensors or using the next value imputation step does not improve the accuracy such as combining both in the same model. The second worse model was the 5th-Markov, with 42.71%. The full imputation strategy with clustering (FIC) does not outperform FI, perhaps because the noise of the full imputation combined with the clusters is significant and does not add real value to the model. Table 5 reports the accuracy of the approaches that use the clustering strategy (BC, NIC, FIC) and how they perform when *eps* varies. The best *eps* value is $eps = 5$ for NIC and $eps = 1$ for FIC, which indicates that higher intra-cluster distances impact more FIC than NIC, adding noise to the learning process. The accuracy for BC is slightly the same when *eps* varies, which highlights the effectiveness of the imputation approaches.

Considering the multi-task learning models, MT-FIC outperforms the other approaches, achieving 85.20% of accuracy that is 20% better than NIC (the best result of the single-task learning approaches), followed by MT-FI with 73.33% and then by MT-NIC with 65.56%. While MT-FIC and MT-FI outperform all single-task learning models, MT-NIC performs sightly similar to NIC. But we can claim that in general, multi-task learning models boost the learning performance compared with single-task learning models. The multi-task version of Markov, MT-5th-Markov, had performed slightly worse than 5th-Markov. Overall, the models MT-FI, MT-NIC, and MT-FIC seem to be benefited from the use of multi-task learning and are resilient to noise since both imputation and clustering can add some noise in the data. Table 5 reports how the accuracy of multi-task clustering-based (MT-BC, MT-NIC, MT-FIC) approaches changes when *eps* varies. The best *eps* value is $eps = 5$ for MT-NIC and for MT-FIC, which indicates that differently from NIC and FIC, higher intra-cluster distances do not impact negatively the accuracy and the multi-task learning can adjust the noise introduced by them into the learning process. MT-BC does not improve the results with the variation of *eps*, which highlights once again the effectiveness of the imputation approaches combined with clustering.

Table 5 Accuracy of the approaches that use the clustering strategy when *eps* varies, the best result of each approach is in bold letters

eps (km)	1	2	3	4	5
BC	46.31	**47.20**	45.95	46.51	46.67
NIC	34.43	38.07	43.98	52.96	**65.84**
FIC	**41.14**	40.33	38.62	35.43	30.23
MT-BC	**48.48**	47.56	48.20	47.34	46.09
MT-NIC	34.03	37.85	43.83	54.48	**65.56**
MT-FIC	72.47	74.01	79.52	82.80	**85.20**

7.2 Evaluation of closeness

To evaluate the quality of the models based on the closeness error, we calculate the closeness error of each prediction on the test set and the percentiles of these values until the 90th percentile. For the clustering-based strategies, we report the closeness error for *eps* value with the best-achieved result.

Figure 7 presents the results of closeness error for all multi-task learning models and for the best single-task learning models in terms of closeness, NI and NIC (*eps* = 5). The single-task approaches NI and NIC achieve 90% of all predictions with closeness error less than 2.7 km. Although NI is the model with worse accuracy, it predicts closest values when compared to the other single-task models. NI achieved 40% of all predictions with closeness error less than 105 m. By looking at the percentile that corresponds to the top accuracy of NIC (*eps* = 5), 60% of all predictions report the closeness error less than 782 m. We refer our previous work [4] for a more detailed evaluation of single-task approaches.

MT-FI outperforms the other models by achieving 70% of its closeness error equals zero (as expected considering its accuracy), 80% less than 1.1 km and 90% less than 2.7 km. The second-best model is MT-NI, that for 60% of samples achieved closeness error equal to zero. The clustering-based approaches, MT-NIC (*eps* = *5*) and MT-FIC (*eps* = *5*) achieved in 90% of closeness error less than 3.2 km and 3.5 km, respectively. However, the 80th percentile for MT-NIC was 1.4 km and MT-FIC was 1.1 km. If we compare MT-NI with MT-NIC and MT-FIC, despite MT-NI has more samples with closeness error equals to zero, MT-FIC and MT-NIC are better until 90% of cases. The other models, MT-B and MT-BC, achieved promising results until the median, then the closeness error increased significantly when compared to the other RNN-based models. The closeness error of MT-1st-Markov and MT-5th-Markov increased significantly after 50% of cases.

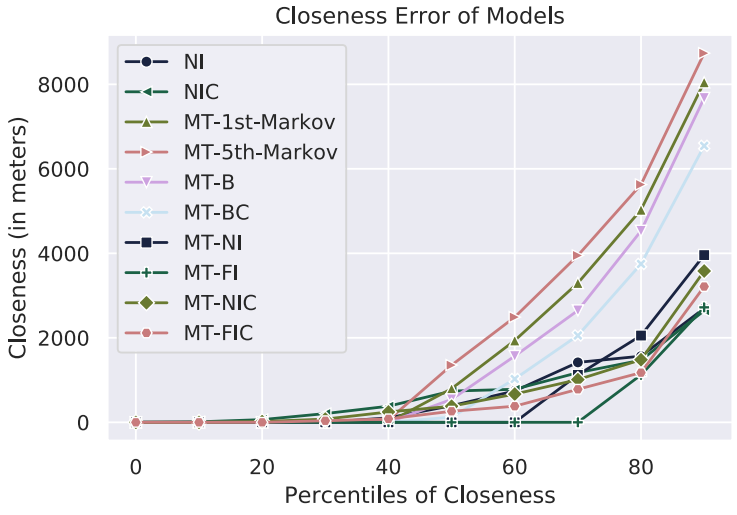

Fig. 7 Comparison of closeness error of multi-task models

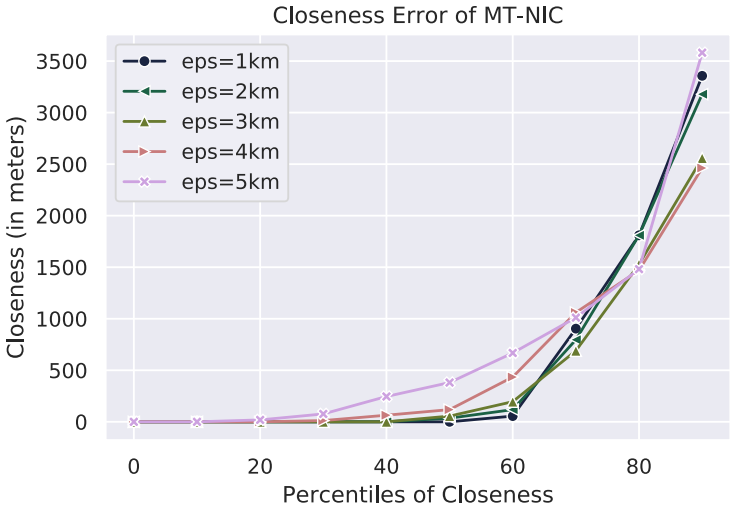

Fig. 8 Comparison of closeness error of MT-NIC when *eps* changes

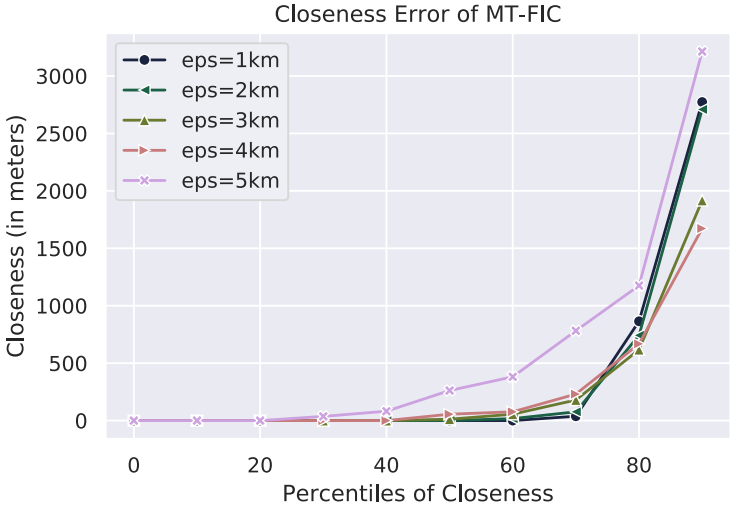

Fig. 9 Comparison of closeness error of MT-FIC when *eps* changes

All in all, we can conclude that the multi-task learning approach could improve the quality of models based on the full imputation strategy. Figures 8 and 9 present, respectively, how the closeness error of MT-NIC and MT-FIC changes when *eps* increases. Concerning the MT-NIC strategy, even though the better accuracy was obtained with $eps = 5$, the model with $eps = 1$ found better closeness error for 60% of cases. After the 70th percentile, the models with $eps = 3$ and $eps = 4$ started to become better in terms of closeness error. Finally, MT-NIC with $eps = 4$, 90% of the closeness error was less than 2.4 km, the lower value of this percentile. Similar to

MT-NIC, MT-FIC with $eps = 1$ obtained closeness error equals to zero for 50% of cases, and less than 55 m for 60% of cases. However, if we consider more samples with $eps = 4$ and $eps = 5$, the results improved. With both $eps = 4$ and $eps = 5$, in 90% of cases, the closeness error was less than 2.5 km, while for $eps = 1$ the 90th percentile is 3.3 km.

7.3 Final considerations

From the experiments, we observe that in terms of accuracy, the multi-task approaches perform better, or at least similar, when compared to their equivalent single-task learning models, except MT-B that achieved lower accuracy than B. Full imputation strategy helps the models to improve accuracy. While FI achieved 41.28% of accuracy, its multi-task version achieved 73.33%. The biggest improvement was obtained by MT-FIC. While FIC, its single-task learning version presented 41.14% of accuracy, its multi-task version achieved 85.20%. This shows the effectiveness of a multi-task strategy when applied together with the proposed prepossessing steps of data imputation and clustering.

Considering the closeness error, the best multi-task learning model was MT-FI with 70% of samples with closeness error equal to zero and 90% less than 2.7 km, while the best single-task model had the same result for the 90th percentile, but it got only 40% of the closeness error equal to zero. We also highlight that the multi-task learning models that use only imputation or imputation and clustering strategies had 70% of closeness error less than 1 km, while in their equivalent single-task learning models, only NIC had the 70th percentile of closeness error less than 1 km. This confirms the better performance of our multi-task learning proposal.

Finally, there is a trade-off for the choosing of *eps*. Big *eps* values produce bigger clusters and consequently a smaller number of sensors labels, which helps to improve the accuracy. As smaller is the number of sensors labels, the model has more ability to learn from them. Also, by clustering the sensors lead us to better accuracy in a clustered model, this is expected since it could refer to many possible sensors. Huge clusters also imply higher closeness errors. A proper value to *eps* is an intermediary value that group sensors that are not that far, maintaining a higher accuracy but also a smaller closeness error. For our data set and for the best clustering-based algorithms (MT-NIC and MT-FIC), the best *eps* were 3 km and 4 km. We can observe from Fig. 2, that most of the distances between sensors transitions are less than 4 km, which indicates that the distribution of distances between consecutive sensors in a trajectory, gives us a good cold start to define the *eps* value.

8 Conclusion and future work

In this paper, we discussed the external sensor trajectory (EST) prediction problem. EST may capture very different mobility patterns since they are not restricted to a fleet or a community of users. They are also sparse, incomplete and uncertain. We provided a data analysis of EST which evidenced the sparsity and incompleteness

of this data. We investigated an approach to deal with the missing data of external sensors. We proposed a new multi-task learning schema based on RNN to learn both time and space information in order to solve EST prediction. We evaluated the proposed scheme using real-world data. The experiments showed that our method could increase the accuracy by about 42% compared to the Markov Model analyzed and about 37% compared to the basic RNN. Also, our approach is more precise concerning the closeness error. As future work, we plan to study how to learn from road distances and minimize the closeness error. We also intend to improve the data imputation method to take into account the uncertainty coming from this process itself.

Acknowledgements This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior —Brasil* (CAPES) under Finance Code 001, *Fundação de Apoio a Serviços Técnicos, Ensino e Fomento a Pesquisas* (FASTEF) (Grant Number 31/2019) and in part by *Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico* (FUNCAP No 8789771/2017).

Disclaimer This work reflects only the author's view and that the EU Agency is not responsible for any use that may be made of the information it contains.

References

1. Alhasoun, F., Alhazzani, M., Aleissa, F., Alnasser, R., González, M.: City scale next place prediction from sparse data through similar strangers. In: Proceedings of ACM KDD Workshop, Halifax, Canada (2017)
2. Bucher, D.: Vision paper: Using volunteered geographic information to improve mobility prediction. In: Proceedings of the 1st ACM SIGSPATIAL Workshop on Prediction of Human Mobility, PredictGIS'17, pp. 2:1–2:4. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3152341.3152344
3. Chollet, F., et al.: Keras. https://keras.io (2015)
4. Cruz, L.A., Zeitouni, K., de Macedo, J.A.F.: Trajectory prediction from a mass of sparse and missing external sensor data. In: 2019 20th IEEE International Conference on Mobile Data Management (MDM), pp. 310–319. IEEE (2019)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd **96**, 226–231 (1996)
6. Feng, J., Li, Y., Zhang, C., Sun, F., Meng, F., Guo, A., Jin, D.: Deepmove: predicting human mobility with attentional recurrent networks. In: Proceedings of the 2018 World Wide Web Conference, WWW '18, pp. 1459–1468. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2018). https://doi.org/10.1145/3178876.3186058
7. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European conference on machine learning and knowledge discovery in databases, pp. 36–40. Springer (2016)
8. Fu, K., Ji, T., Zhao, L., Lu, C.T.: Titan: a spatiotemporal feature learning framework for traffic incident duration prediction. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 329–338. ACM (2019)
9. Hasan, S., Ukkusuri, S.V.: Reconstructing activity location sequences from incomplete check-in data: a semi-markov continuous-time bayesian network model. IEEE Trans. Intell. Transp. Syst. **19**(3), 687–698 (2017)
10. Hendawi, A.M., Bao, J., Mokbel, M.F.: iroad: a framework for scalable predictive query processing on road networks. Proc. VLDB Endow. **6**(12), 1262–1265 (2013)
11. Karatzoglou, A., Jablonski, A., Beigl, M.: A seq2seq learning approach for modeling semantic trajectories and predicting the next location. In: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '18, pp. 528–531. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3274895.3274983

12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014). arxiv:1412.6980
13. Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., Liu, Y.: Multi-task representation learning for travel time estimation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1695–1704. ACM (2018)
14. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, pp. 194–200. AAAI Press (2016). http://dl.acm.org/citation.cfm?id=3015812.3015841
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 26, pp. 3111–3119. Curran Associates Inc., New York (2013)
16. Naserian, E., Wang, X., Dahal, K., Wang, Z., Wang, Z.: Personalized location prediction for group travellers from spatial-temporal trajectories. Future Gener. Comput. Syst. **83**, 278–292 (2018). https://doi.org/10.1016/j.future.2018.01.024
17. Rocha, C.L., Brilhante, I.R., Lettich, F., De Macedo, J.A.F., Raffaetà, A., Andrade, R., Orlando, S.: Tpred: a spatio-temporal location predictor framework. In: Proceedings of the 20th International Database Engineering; Applications Symposium, IDEAS '16, pp. 34–42. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2938503.2938544
18. Trasarti, R., Guidotti, R., Monreale, A., Giannotti, F.: MyWay: location prediction via mobility profiling. Inf. Syst. **64**, 350–367 (2017). https://doi.org/10.1016/j.is.2015.11.002
19. Wu, F., Fu, K., Wang, Y., Xiao, Z., Fu, X.: A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. Algorithms **10**(2), 37 (2017). https://doi.org/10.3390/a10020037
20. Wu, H., Chen, Z., Sun, W., Zheng, B., Wang, W.: Modeling trajectories with recurrent neural networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, pp. 3083–3090. AAAI Press (2017). http://dl.acm.org/citation.cfm?id=3172077.3172319
21. Yao, D., Zhang, C., Huang, J., Bi, J.: Serm: a recurrent model for next location prediction in semantic trajectories. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2411–2414. ACM (2017)
22. Zhang, C., Zhang, K., Yuan, Q., Zhang, L., Hanratty, T., Han, J.: GMove : group-level mobility modeling using geo-tagged social media. In: In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1305–1314 (2016). https://doi.org/10.1145/2939672.2939793
23. Zhao, W.X., Zhou, N., Sun, A., Wen, J.R., Han, J., Chang, E.Y.: A time-aware trajectory embedding model for next-location recommendation. Knowl. Inf. Syst. (2017). https://doi.org/10.1007/s10115-017-1107-4