UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

LÍVIA ALMADA CRUZ

LOCATION PREDICTION FROM EXTERNAL SENSORS TRAJECTORIES

FORTALEZA

2022

LÍVIA ALMADA CRUZ

LOCATION PREDICTION FROM EXTERNAL SENSORS TRAJECTORIES

Tese apresentada ao Programa de Mestrado e Doutorado em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Banco de Dados.

Orientador: Prof. Dr. Jose Antonio Fernandes de Macedo.
Coorientadora: Profa. Dra. Karine Zeitouni.

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C962l Cruz, Lívia Almada.
Location prediction from external sensors trajectories / Lívia Almada Cruz. – 2022.
105 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2022.
Orientação: Prof. Dr. José Antônio Fernandes de Macedo.
Coorientação: Profa. Dra. Karine Zeitouni.

1. Predição de localização. 2. Predição de trajetórias. 3. Aprendizado de representação. 4. Modelagem de trajetórias . 5. Sensores. I. Título.

CDD 005

LÍVIA ALMADA CRUZ

LOCATION PREDICTION FROM EXTERNAL SENSORS TRAJECTORIES

Tese apresentada ao Programa de Mestrado e Doutorado em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Banco de Dados.

Aprovada em: 09/08/2022.

BANCA EXAMINADORA

Prof. Dr. Jose Antonio Fernandes de Macedo (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Karine Zeitouni (Coorientadora)
Université de Versailles Saint-Quentin-en-Yvelines (UVSQ - France)

Prof. Dr. João Paulo Pordeus Gomes
Universidade Federal do Ceará (UFC)

Dra. Chiara Renso
Consiglio Nazionale delle Ricerche (CNR-Italy)

Dr. Francesco Lettich
Consiglio Nazionale delle Ricerche (CNR-Italy)

In memoriam of my grandmother, Maria José;
who is always in my heart.

ACKNOWLEDGEMENTS

I thank and give God praise for my life and for providing me the fortitude and perseverance to overcome all of life's obstacles.

I thank my family, who gives me the support I need. Particularly, my mother Silvaelena Almada and my sister Gizele Almada, who believe in me more than anybody else. I thank you for being my friends and walking alongside me.

To Carol Fernandes and Camila Sena, two of my lifelong friends on whom I can always rely.

To my advisor Prof. José Macedo. I sincerely appreciate your patience, guidance, and assistance with this research. Thank you for your support and understanding, especially when I could not give it my all. I hope that one day I will be able to repay you for all of the opportunities you have given me.

To my co-advisor Prof. Karine Zeitouni, who gently received me in David Lab, in Versailles, and assisted me to conduct this research. Your guidance and observations were essential for the accomplishment of this work.

To my friend, Ticiana Linhares. For all of the great times we had together: studying hours, projects, research collaboration, coffees, lunches, travels, and all of the fun times. You inspired me.

To my friend and partner in master's degree, research and work, Régis Pires. I am very thankful for the opportunity to learn with you every day.

To all my teachers and professors, who inspired me to believe in Education as an instrument of world and people transformation. Thank you for all of your lessons. Thank you for assisting in the transformation of my life and the lives of so many others.

To all my colleagues and friends from *Universidade Federal do Ceará* (UFC). Especially, Arthur Araruna and Márcio Costa who started with me in my undergraduate degree this challenging journey through Computer Science. I thank you for your friendship and for all the knowledge we have already exchanged.

To all of my friends and coworkers from UFC Campus Quixadá who, through education, make a significant contribution to the development of that city and Sertão Central of Ceará.

To my friends and colleagues from Insight Lab, all the ones who contributed directly and indirectly to this research: Dayana, Vitória, Igo Brilhante, José Neto, Wilken, Matheus,

Leopoldo, Lucas, Gustavo, Hinessa, Andreza, Arina, Francesco Lettich, David, Isabel, among others.

To all my friends from David Lab for the wonderful reception during my stay in France, at the University of Versailles, and for the good memories: Jingwei Zuo, Hadi Dayekh, Ahmad Ktaish, Mariem Brahem, and others.

To the UFC, my workplace, and my second home, for supporting my professional development and allowing me to participate in an exchange program at the University of Versailles in 2018.

I would like to thank my thesis committee, Prof. Dr. José Macedo, Profa. Dra. Karine Zeitouni, Prof. Dr. João Paulo Pordeus, Dr. Francesco Lettich and Dra. Chiara Renso, for all the considerations which improved this thesis.

To *Fundação Cearense de Apoio ao Desenvolvimento* (FUNCAP) for the funding of the project *Segurança Pública Integrada* (SPI) which this work is close related.

To the ones not cited here, but that have contributed in some way to this work. Thank you.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

“For my part, I know nothing with any certainty, but the sight of the stars makes me dream.” (GOGH, 1988, apud JANSEN *et al.*, 2009, p.638)

RESUMO

Esta tese propõe uma esquema baseado aprendizado profundo de múltiplas tarefas para prever a próxima localização a partir de trajetórias capturadas por sensores externos (por exemplo, câmeras de vigilância de tráfego ou radares de velocidade). As posições reportadas nessas trajetórias são esparsas, devido à distribuição dos sensores, e incompletas, porque os sensores podem falhar ao registrar a passagem de objetos. Nesta estrutura, são propostas diferentes etapas de pré-processamento para alinhar a representação das trajetórias e lidar com o problema de dados ausentes. Uma abordagem de aprendizado de múltiplas tarefas baseada em redes neurais recorrentes é apresentada, ela utiliza informações de tempo e espaço na fase de treinamento para aprender representações mais significativas. O modelo de aprendizado de múltiplas tarefas, junto com a etapa de pré-processamento, melhora substancialmente o desempenho da previsão. Esta tese também trata o problema do aprendizado de representação de trajetórias, que é uma tarefa essencial no aprendizado de máquina. Avalia-se como modelos de processamento de linguagem natural capturam a representação de sensores e trajetórias e como essas representações melhoram um modelo de predição de localização. Observa-se que o espaço de características identificados por tais modelos podem capturar a relações de similaridade espacial para sensores e trajetórias dentro de uma determinada vizinhança.

Palavras-chave: predição de localização; predição de trajetórias; aprendizado de representação; modelagem de trajetórias; sensores.

ABSTRACT

This thesis proposes a multi-task deep learning-based scheme to predict the next location from trajectories captured by external sensors (eg traffic surveillance cameras or speed cameras). The positions reported in these trajectories are sparse, due to the distribution of the sensors, and incomplete because the sensors can fail to register the passage of objects. This framework includes different pre-processing steps to align the representation of trajectories and deal with the problem of missing data. We present a multitasking learning approach based on recurrent neural networks. This approach uses time and space information in the training phase to learn more meaningful representations. The multi-task learning model jointly with the pre-processing step substantially improves the prediction performance. This thesis also deals with the problem of representation learning for trajectory data. Representation learning concerns the problem of learning low-dimensional representation from complex data, and it is an essential task in machine learning. We evaluate how natural language processing models capture the representation of sensors and trajectories. The empirical evaluation shows that the space of features identified by such models can capture the spatial similarity relationships for sensors and trajectories within a given neighborhood. We also evaluate how these representations improve a location prediction model.

Keywords: location prediction; trajectory prediction; representation learning; trajectory modeling; sensors.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

B	*Basic*
BC	*Basic with Clustering*
BERT	*Bidirectional Enconder Representations from Transformers*
CBMCE	*Class-Balanced Multi-Class Cross-Entropy*
CIOPS	*Integrated Police Operation Center*
EST	*External Sensor Trajectory*
FI	*Full Imputation*
FIC	*Full Imputation with Clustering*
GPS	*Global Positioning System*
KDE	*Class-Balanced*
KDE	*Kernel Density Estimation*
LBERT	*LSTM with BERT*
LBERT-FT	*LSTM with BERT and Fine-tuning*
LSTM	*Long Short-Term Memory*
LW2V	*LSTM with Word2Vec*
LW2V-FT	*LSTM with Word2Vec and Fine-tuning*
MLM	*Masked Language Model*
MT-B	*Multi-task Basic*
MT-BC	*Multi-task Basic with Clustering*
MT-FI	*Multi-task with Full Imputation*
MT-FIC	*Multi-task with Full Imputation and Clustering*
MT-NI	*Multi-task with Next Value Imputation*
MT-NIC	*Multi-task with Next Value Imputation and Clustering*
NI	*Next Value Imputation*
NIC	*Next Value Imputation with Clustering*
NLP	*Natural Language Processing*
RNN	*Recurrent Neural Network*
SPIA	*Approach Police System*
SSPDS	*Public Security Secretariat and Social Defense*
ST-CBMCE	*Spatial-Temporal Class-Balanced Multi-Class Cross-Entropy*

LIST OF SYMBOLS

$c_{tran_{s_i,s_j}}$	completed transition from sensor s_i to sensor s_j
d_R	road distance
d_{DTW}	DTW distance between trajectories
d_{ED}	Edit distance between trajectories
e_{ij_k}	k-th edge belonging to the shortest path from sensor s_i to sensor s_j
G	graph representing the road network
M	set of moving objects
O	set of observations
o_i	i-th observation of an object in a trajectory
(m,s,t)	tuple representing an observation of object m in sensor s at timestamp t
$p_{i,j}$	shortest path from sensor s_i to sensor s_j
μ_{s_i,s_j}	average of displacement time from sensor s_i to sensor s_j
σ_{s_i,s_j}	standard deviation of displacement time between sensor s_i to sensor s_j
S	set of sensors
$s_i \rightarrow s_j$	trajectory transition from sensor s_i to sensor s_j
$\mathbf{l}_k$	spatial label feature
$\mathbf{t}_k$	temporal feature

CONTENTS

1 INTRODUCTION

The popularization of location-based services (such as Google and Waze, for example) allowed the growth of trajectory data. Trajectory data brought new opportunities for discovering mobility patterns and understanding the evolution of mobility from those data. In this context, trajectory prediction, or location prediction, is the problem to infer the next relevant place of a moving object, based on its historical trajectories. Location prediction has many real applications, like a touristic recommendation, traffic management, and police patrol, among others; and has been called the attention of researchers in the last years.

The Public Security Secretariat and Social Defense (SSPDS in Portuguese) of the state of Ceará, in Brazil, has developed the Approach Police System (SPIA in Portuguese) to succeed in dealing with the mobility of criminals. SPIA uses the video surveillance system in conjunction with teams of police actions, especially motorcycle patrol. SPIA processes, in real-time, data obtained from cameras equipped with license plate recognition (LPR) system. SPIA checks whether each captured vehicle plate is related to a declared theft. If so, it informs the Integrated Police Operation Center (CIOPS in Portuguese), where a vehicular approach police action is fired. Next, CIOPS operators carry out the planning to approach the stolen vehicle through a visual examination of the video monitoring cameras scattered around the city. Implemented in 2017, SPIA has allowed a 48% reduction in vehicle theft actions in the state of Ceará. Figure 1 depicts a map of Fortaleza and its metropole region with the surveillance cameras (black points) used by SPIA in 2019 plotted on that map.

Despite the efficiency of SPIA, one of the most complex activities is to predict the trajectory of vehicle movement. Currently, this is done manually using several police agents checking the video monitoring cameras simultaneously. The *modus operandis* also requires the availability of several police vehicles scattered throughout the city, which must be dedicated to the service in question. Consequently, this current mode of operation is costly and subject to errors.

This thesis develops approaches for predicting the next vehicle movement, which will allow predicting the displacement of a vehicle among the video surveillance cameras. 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

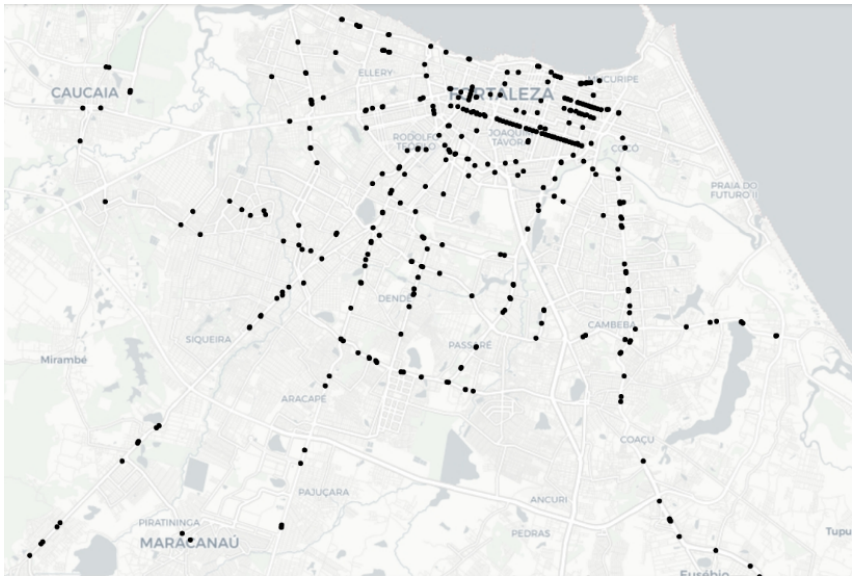


Figure 1 – Map of Fortaleza city with external sensors (surveillance cameras) positioned on the road network, in 2019.

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. Given this contextualization, we can present the research questions investigated in this thesis.

1.1 Research Questions

The research questions of this work concern two main problems: (i) the location prediction from external sensors trajectories and (ii) the representation of this kind of data for the use by prediction models.

1.1.1 Location Prediction from External Sensors Trajectories

Bucher (2017) defines three levels of mobility prediction: 1) object position; 2) path prediction; and 3) next place prediction. In object position and path prediction levels (1 and 2), the models usually learn from raw trajectories obtained by Global Positioning System (GPS) devices, and their predictions consider the movement of the object. The next place prediction 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. We positioned this work between level 1 (object position prediction) and level 2 (movement and path prediction). 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 and 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 (ROCHA *et al.*, 2016) and (FENG *et al.*, 2018) to name a few; and (ii) the next (stop) location prediction, usually applied to points of interest (PoIs) or event places as (YAO *et al.*, 2017a), (KARATZOGLOU *et al.*, 2018a), (HASAN; UKKUSURI, 2017) and (WU *et al.*, 2017b) since we predict movement rather than stops. Thus, despite many works in the literature on mobility prediction, to the best of our knowledge, just a few of them have studied mobility prediction for trajectories based on external sensor data. We call this problem External Sensor Trajectory Prediction (EST Prediction, for short).

A solution for location prediction from EST requires dealing with the data quality problems; like sparsity, incompleteness, and uncertainty. Our research questions regarding the EST prediction are listed below.

RQ1 - How to adapt prediction models to incompleteness and sparsity of EST?

RQ2 - How the different data imputation strategies impact location prediction models?

RQ3 - How to predict the next location from EST?

RQ4 - How to access the quality of predictions given the fact that the ground truth could be missed and thus unknown?

1.1.2 Representation Learning for External Sensors Trajectory

Representation learning refers to methods that seek to transform complex, high-dimensional (and often redundant) data into an effective and low-dimensional representation while preserving information embedded in the raw data (BENGIO *et al.*, 2013).

Representation learning for trajectory modeling is concerned with building statistical or machine learning models of observed trajectories of vehicles or people. Such models may

have different uses, like computing the probability of observing a given trajectory for anomaly detection (JI *et al.*, 2020); estimating the importance of different characteristics that drivers may consider relevant when following a trajectory; recovering sparse or incomplete trajectories as the ones observed from external sensors (WU *et al.*, 2016); aiding drivers to choose an optimal route from an origin to a destination, or predicting online the next location of a vehicle given its current location (FENG *et al.*, 2018). Representation learning is an important task to build effective machine learning models and also to have useful and interpretable representations for complex data.

Mikolov *et al.* (2013a) proposes Word2Vec, a framework that uses neural networks for representation learning in Natural Language Processing (NLP). Word2Vec extracts the features of words' semantic meaning from their sequential orders. As result, the model generates word vectors where geometric distances between them reflect semantic similarity and difference vectors encode semantic and syntactic relations (MIKOLOV *et al.*, 2013c). Since then, many other models for learning text representations were proposed; like GloVe (PENNINGTON *et al.*, 2014), FLAIR (AKBIK *et al.*, 2019) and Paragraph Vector (LE; MIKOLOV, 2014), to name a few. NLP word representation methods model the semantics of a word, and its similarity with other words, by observing the many contexts of use of the word in the language.

In this thesis we take inspiration from NLP approaches to investigate the following research questions:

RQ5 - Could NLP embedding models, more specifically, language models and word embeddings, be used to represent the vector space of trajectories in location prediction tasks?

RQ6 - Are the representations of sensors/locations from representation models in NLP able to capture their context, i.e., the closest sensors/locations, both in terms of road distance and connectivity?

RQ7 - Could trajectory representations from NLP embedding models adequately capture trajectories similarity?

1.2 Contributions and Publications

Driven by the research questions presented in Section 1.1.1, this work proposes a framework for location prediction based on EST in order to improve the quality of predictions, accessed in terms of the accuracy and road network distance between the predicted and expected locations. The solution and the problems related to the trajectory prediction from EST are

discussed in Chapter 3. The results are addressed in the following published research papers, listed together with the classification of the conference or journal according to the QUALIS[1], the official scientific index used by the Brazilian Higher Education Personnel Improvement Coordination (CAPES, in Portuguese), and its contributions.

- The paper *"Trajectory Prediction from a Mass of Sparse and Missing External Sensor Data"* (CRUZ *et al.*, 2019b) published at IEEE International Conference on Mobile Data Management (MDM), QUALIS A2, presents (i) an imputation approach to deal with incompleteness and sparsity of EST; (ii) different strategies to train the learning models considering the data imputation; (iii) a framework that integrates strategies of data imputation, clustering and learning model based on Recurrent Neural Network; (iv) experiments with real EST data set considering different combinations of data imputation and clustering strategies.
- The paper *"TrajSense: Trajectory Prediction from Sparse and Missing External Sensor Data"* (CRUZ *et al.*, 2019a) published at IEEE International Conference on Mobile Data Management (MDM), QUALIS A2, presents a system to visualize and compare the effectiveness of the proposed strategies.
- The paper *"Location prediction: a deep spatiotemporal learning from external sensors data"*(CRUZ *et al.*, 2021) published at Distributed and Parallel Databases Journal, QUALIS B1, in the special edition of the best papers from MDM extends (CRUZ *et al.*, 2019b) by proposing a multitask strategy for EST prediction. The multitask model jointly learns both spatial and temporal information, which had boosted the quality of sensor predictions.
- The paper *"Predicting the Next Location for Trajectories From Stolen Vehicles"*(NETO *et al.*, 2021) published at IEEE International Conference on Tools with Artificial Intelligence, QUALIS B1, evaluates how semantic information and spatial representation impact predictions on EST prediction obtained from stolen vehicles. This contribution resulted in a Master's Thesis.

Driven by the research questions presented in Section 1.1.2, we performed an analysis of representation learning of sensors and external sensors trajectories using NLP models, we evaluate these representations under the location prediction task and evaluate how sensors and trajectory representations reflect the spatial similarities.

[1] The QUALIS index is divided into eight strata, in descending order of value: A1, A2, B1, B2, B3, B4, B5, and C.

– During this thesis written, the results of this investigation were condensed in the paper *"Representation Learning for External Sensors Trajectories"* submitted and accepted to be published in the Sensors Journal, QUALIS A1.

1.3 Document Structure

The remaining of this document is organized as follows. Chapter 2 reports an overview of the related work in the literature about prediction methods for mobility, including location prediction and multi-task learning; and representation learning for mobility data. Chapter 3 introduces the Location Prediction problem and presents the related challenges. In Chapter 4, we present the proposed framework for EST prediction. Chapter 5 details our investigation on representation learning for EST. Finally, Chapter 6 summarizes the contributions of this thesis and indicates future work.

2 RELATED WORK

In the literature, there exists a vast amount of works dealing with the problem of prediction models for mobility and representation 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 mobility data, including location prediction and spatiotemporal multi-task learning; while the second class employs machine learning models for representation learning on mobility data.

2.1 Prediction Models for Mobility

We split the works in this section into two categories: works that solve next location prediction for trajectories, and works that apply multi-task learning for mobility prediction in general.

2.1.1 Next Location Prediction

MyWay (TRASARTI *et al.*, 2017) predicts human movements based on three different strategies, one based on the individual behavior of a single user, a global model based on the behavior of all users, and a hybrid approach. On their methods, personal mobility profiles capture the routines of users. They cluster the trajectories of one user; then, they can define routines as the representative trajectories of each cluster. An individual profile is a set of routines of one user, and a global profile is a set of all individual profiles. The approach is based on raw GPS trajectories, so the clustering procedure considers spatial similarity and temporal information to interpolate points to deal with different sampling rates. For the predictions, they match the current trajectory with the profiles. Again, MyWay assumes trajectories are dense to be able to use the similarity measure proposed by the authors, which is not valid for EST trajectories. TPRED (ROCHA *et al.*, 2016) is a individual approach to predict the next stop of a moving object. The method is based on a prefix tree, built from the historical data of a single user. In this tree, the nodes represent the permanence of the object in a specific location within a temporal partition and edges represent observations of movements between nodes. The edges are annotated with probabilities that combine spatial and temporal probabilities. The space was divided by a grid and relevant places are cells of this grid with some frequency of stops by the user. TPRED predicts the next relevant location where an user will stop and when the user will leave from its current place. TPRED assumes trajectories are dense in order to be able to detect

stops, different from EST trajectories (Scenario 1). EST trajectories are sparse and each location registers the passage when the moving object is in movement that does not allow to identify a specific place as stop. Also, TPRED does not use the road network.

A group-based approach was proposed by (NASERIAN *et al.*, 2018) predicts the next stop of a single user. The trajectories were obtained from the movements of a group of travelers in the indoor environment. The stops are geographic regions where a user stayed for some period. In that method, profile information (like gender and age) is used to identify different types of groups. The approach extracts sequences of general locations and group-specific locations to mine sequential rules from significant sequential patterns. From the sequential rules, the method estimates the probability of visiting a specific place given the recent movement of the user and his group. This method cannot be applied when the profile information of users is not available, as the case of EST.

GMove (ZHANG *et al.*, 2016) uses spatial and temporal information, and geo-tagged text extracted from the social media, to predict the location of one user by means a group-level mobility model. In this work, a trajectory is a sequence of geo-referenced social media records. The model proposed is an ensemble of Hidden Markov Models (HMM), each HMM is built using a group of users that share similar movements. The latent states have a geographic center and a set of keywords. Keyword correlations are used as auxiliary knowledge on the group construction. As in (NASERIAN *et al.*, 2018) requires semantic information of the user from social media, that is not available in the scenarios of this project. The work of (ALHASOUN *et al.*, 2017) 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.

Recurrent Neural Network (RNN) was also successfully applied in Natural Language Processing to capture dependence between terms of sequences (MIKOLOV *et al.*, 2013b). More recently, applying 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 (LIU *et al.*, 2016) to predict the next location dealing with continuous values in spatial and temporal contexts. Different from ST-RNN we do not deal with continuous trajectories in space and in time. SERM (YAO *et al.*, 2017a) is a spatial-temporal model to

predict the next stop using semantic trajectories from Social Media. The model of SERM uses spatiotemporal information, user preferences, and a bag-of-words obtained from the user check-ins. The spatial, temporal, and semantic features are mapped into one-hot vectors. In the model, the embedding layers are responsible for reducing the dimension of the vectors, the features are concatenated, and the result is the input of the recurrent layer. Finally, a vector with user preferences is summed to the output of RNN. Different from SERM, we do not have semantic features and user preferences available.

Karatzoglou *et al.* (2018a) proposes Seq2Seq, an attention-and an encoder-decoder learning approach based on Long Short-Term Memory (LSTM) 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. DeepMove (FENG *et al.*, 2018) 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. To learn personal preferences, DeepMove leverages user embedding which is not available in our application. Then, DeepMove combines the context vector and the output of RNN to make the predictions.

Wu *et al.* (2017b) and Wu *et al.* (2017a) solve the trajectory prediction problem under road-network restrictions. Both of them are based on Recurrent Neural Networks and consider dense GPS trajectories as input. For Wu *et al.* (2017b), 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 (WU *et al.*, 2017a) 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 (HENDAWI *et al.*, 2013) and (HENDAWI *et al.*, 2017) which predict movement without learning from the historical data but by applying predictive query. Panda* (HENDAWI *et al.*, 2017) handles predictive Spatial-temporal queries on moving objects' trajectories in Euclidean space, such as range, predictive KNN, and predictive aggregate queries. Panda* uses a grid structure that split the space into cells and a Travel Time Structure to

find out the set of reachable cells, given one cell.

2.1.2 Multi-task Learning for Mobility Prediction

Multi-task learning refers to machine learning models where multiple related tasks at the same time in order to improve the overall performance. From now, we cite some works which employ multi-task learning on mobility data.

The paper (HASAN; UKKUSURI, 2017) proposes a semi-Markov 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 (LI *et al.*, 2018b), 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 (LI *et al.*, 2018b) predicts travel-time, but not the next location.

Fu *et al.* (2019) 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 problems addressed by (HASAN; UKKUSURI, 2017; LI *et al.*, 2018b; FU *et al.*, 2019) are different from the location prediction targeted in this thesis. However, they show the importance of multi-task learning for spatiotemporal features.

2.1.3 Comparative of Prediction Models for Mobility

Table 1 presents a comparative summary of related work on the prediction models for mobility data. The column **Base Model** refers to the machine learning model used on the work, the column **Task** refers to the prediction task solved, and the column **Data** is the type of mobility data collected to compose the trajectories, and finally, **Additional Information** is the type of the additional data available to enrich the model.

Table 1 – Comparative table of related work on prediction models for mobility data.

	Base Model	**Task**	**Data**	**Additional Information**
Rocha *et al.* (2016)	Prefix Tree	Next Stop	GPS trajectories	-
Naserian *et al.* (2018)	Sequential Rules	Next Stop	Indoor trajectories	User Profile
Zhang *et al.* (2016)	HMM	Next Stop	Check-ins	Geo-tagged text
Alhasoun *et al.* (2017)	Bayesian Network	Next Stop	Call Detailed Records	Similar Strangers
Karatzoglou *et al.* (2018a)	Seq2Seq LSTM	Next Stop	GPS trajectories	Semantic Location
Liu *et al.* (2016)	Continuous RNN	Next Stop	Check-ins	-
Wu *et al.* (2017a)	LSTM	Next Road Segment	GPS	Road Network
Wu *et al.* (2017b)	LSTM	Location	GPS	Road Network
Yao *et al.* (2017a)	RNN	Next Stop	Check-ins	Geo-tagged text
Feng *et al.* (2018)	RNN	Stop at next time window	Check-ins	-
Hendawi *et al.* (2013)	Predictive query processor tree	Final Destination	GPS trajectories	Travel Time
Hasan e Ukkusuri (2017)	Multi-task Markov	Next Activity and Duration	Check-ins	-
Li *et al.* (2018b)	Multi-task RNN	Travel-time Prediction	GPS trajectories	Road-network
Fu *et al.* (2019)	Multi-task learning	Incident duration	Traffic speed readings	Road Network
This work	Multi-task RNN	Next location	Sensor readings	Road Network

2.2 Representation Learning for Mobility

Representation learning is a category of unsupervised learning method that aims at transforming complicated, high-dimensional real-world data into low-dimensional data while preserving information embedded in the raw data (BENGIO *et al.*, 2013).

Several works have been proposing representation learning for different applications, such as modeling sequential interactions between users and items/products for social networks or e-commerce. JODIE (KUMAR *et al.*, 2019) embeds each user/item in a Euclidean space, and an embedding trajectory models its evolution in this space. JODIE learns via embeddings the temporal iterations between users and items, where the item embeddings are based on learning a future user embedding projection. JODIE uses a coupled mutually-recursive recurrent neural network, one RNN updates user embeddings, and the other RNN updates item embeddings. The hidden states of RNNs represent user and item embeddings. A temporal attention layer predicts the embedding of users after some time. JODIE learns embedding representation for the recommendation of items based on the user/item sequences, which is a different problem from the one presented in this thesis.

Habit2vec (CAO *et al.*, 2019) models a person's habit as a vector which upgrades the word2vec model according to particular characteristics of trajectories. The paper aims to find the similarity of living patterns as engaging in similar behavior at similar times instead of staying in geographically neighboring locations. Habit2vec trains the trajectories into the neural network model Continuous Bag-of-Words (CBOW) to learn the embedding vectors of habits. After modeling the living habit into an embedding in a single space, the authors use classical clustering methods, such as K-Means and density-based methods, to cluster similar living habits. Mob2Vec (DAMIANI *et al.*, 2020) is a framework for learning representation of human mobility using trajectories obtained from Call Details Records (CDR). Mob2Vec uses Sqn2Vec (NGUYEN *et al.*, 2019), an unsupervised approach based on Paragraph Vector (PV) (DAI *et al.*, 2015). Mob2Vec firstly summarizes trajectories, removing irrelevant and noisy locations. Then, Mob2Vec obtains vector representations of trajectories using Sqn2Vec. Finally, it aggregates the vectors of trajectories from the same user to obtain a unique representation. In spite of (CAO *et al.*, 2019; DAMIANI *et al.*, 2020) propose learning representation for mobility data, the goal of them is to represent habits or behaviors of human mobility which is different from our proposal.

TA-TEM (ZHAO *et al.*, 2018) is a time-aware trajectory embedding model for

the next-location recommendations which deal with sequential information and data sparsity problems. TA-TEM learns from the sequence of check-ins and considers the preferences of each user at different levels of time and general user preferences. The system incorporates temporal factors in the next location. The temporal factors are the trajectory sequence itself, the dynamic and static user preferences for capturing changes in user interests, and weekly and daily check-in patterns, on a day and an hour scale respectively. The approach uses a model similar to the CBOW(MIKOLOV *et al.*, 2013b), where the context is used to predict an item on the sequence. Trajectories are modeled as a sequence of check-ins, each one with its timestamp (day and hour). Embedding vectors model locations, user preferences per month, general user preferences, and preferences in periods of days and of weeks. As in this work, TA-TEM learns how to represent sparse trajectories. However, they user preferences are not available in our context.

Li *et al.* (2018a) presents a sequence-to-sequence (Seq2Seq) model for learning trajectory representations for similarity computation between trajectories. The approach generates low sampling trajectories from high sampling trajectories without changing the underlying route. On training, the encoder embeds the low sampling trajectory in a vector representation, and the decoder tries to recover the original trajectory using the embedded vector. To incorporate spatial proximity in the model, they proposed a spatial proximity loss function that penalizes the error as much as the distance between spatial cells. Unlike Li *et al.* (2018a), this work learns representations for location prediction, not for similarity computation.

Trembr (FU; LEE, 2020) proposes to learn trajectory embeddings via road networks. The method has three main steps: apply the map-matching of trajectories on the road network; learn road segments embedding representations; learn trajectory representations. For representation learning of road segments, they proposed Road2Vec, which uses a binary classifier to learn if two road segments co-occur on the same trajectory. For trajectory representation learning, they propose Traj2Vec a multi-task RNN encoder-decoder that learns both spatial and temporal properties. The model receives trajectories represented as a sequence of road segments embedding. A function to capture the road segment relevance is used in the decoding process. Trembr requires road network matching to transform trajectories into sequences while considering the road connectivity in the model. Unlike Trembr, the whole path of the vehicle on the road network is not available for our problem.

TraceBERT (CRIVELLARI *et al.*, 2022) addresses the problem of location-based trajectory modeling. TraceBERT infers the lack of spatial observation of moving objects based

on previously visited locations along the trajectory. The proposed solution models each location as a word in the Bidirectional Encoder Representations from Transformers Masked Language model (BERT (MLM) (DEVLIN *et al.*, 2018). BERT MLM works by masking certain words over text, and it tries to predict them based on the context provided by the unmasked words. As in this work, TraceBERT trains the BERT model on trajectories rather than text, masking the locations along the path and using the BERT model to infer the missing locations. Unlike TraceBERT, this work applies the NLP models to obtain the pre-trained embedding of sensors in order to incorporate them into a location prediction model. We evaluate how the embedding representation impacts the performance of predictors, with and without fine-tuning. Additionally, we investigate how the learned representation can capture spatial and connectivity relationships between sensors and trajectory similarities.

2.2.1 Comparative of Representation Learning for Mobility

Table 2 presents a comparative summary of related work on representation learning for trajectory data. The column **Base Model** is the model used to learn representations. The column **Representation Data** is the type of data for which the embedding vectors are learned. The column **Primary Task** describes the main task solved in the work, in other words, the task for which the vectors generated by the representation learning model are used. The column **Additional Information** refers to some additional data used.

Table 2 – Comparative table of related work on representation learning for trajectory data.

	Base Model	**Represented data**	**Primary Task**	**Data**	**Additional Information**
Zhao *et al.* (2018)	CBOW	Location and user preferences	Recommendation	Check-ins	User Preferences
Li *et al.* (2018a)	Seq2Seq	Low-sampling trajectories	Trajectory similarity	GPS trajectories	Road Network
Cao *et al.* (2019)	Word2Vec and Clustering	Living Habits	Representation learning	POIs	POI type
Kumar *et al.* (2019)	Coupled RNN	User/item iterations	Recommendation	Sequence of user/item	-
Damiani *et al.* (2020)	Paragraph Vector	Human Mobility	Mobility behavior similarity	CDR	-
Fu e Lee (2020)	Encoder-Decoder	Road segments, Trajectory	Travel-time and destination predictions, trajectory similarity	GPS	Road network
Crivellari *et al.* (2022)	BERT MLM	Locations	Trajectory reconstruction	CDR	-
This work	BERT MLM, LSTM	Road Sensor, Trajectory	Next location	Sensor readings	-

3 LOCATION PREDICTION FROM EXTERNAL SENSOR TRAJECTORIES

In this chapter, we introduce our problem statement and discuss the challenges of working with trajectories collected by external sensors.

3.1 Problem Statement

Given a set of external sensors placed on roadsides that 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 2 presents a real trajectory of a bus collected from external sensors in one day; in such a 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, 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 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 (External Sensor Trajectory (EST)) of a moving object m*

Figure 2 – 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.

that can be extracted from $O[m]$, as a the sequence of observations $es\_traj_m = \langle o_1, o_2, \cdots, o_j \rangle$ such that $\forall i, 1 \leq i \leq j, o_i \in O[m]$ and $o_i.t \leq o_j.t$.

Now, we are ready to present the next sensor prediction problem.

Definition 3 (Location Prediction From External Sensor Trajectory) *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.*

3.2 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 Figure 2).

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.

3.3 Data Sparsity and Incompleteness

In this section, we study the sparsity and incompleteness of sensor data. We analyzed 272 sensors from Fortaleza city in September 2017, that are used in a real application. Initially, we have 22,338,916 observations. For each day, we computed the trajectories from the sequences of observations of the same moving object. After that, to compute trips, we partitioned the trajectories according to a pre-defined condition (we explain in the following), and then we discard the shortest trajectories.

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.

3.3.1 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 3(a) shows the *Kernel Density Estimation* (KDE) of the road distances and Figure 3(b) 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 1,765.88 meters for more than 50% of all transitions. Some of these distances are greater than 14 kilometers. However, 50% of the distances between one sensor and its four closest sensors along the road network are less than 755.61 meters. 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.

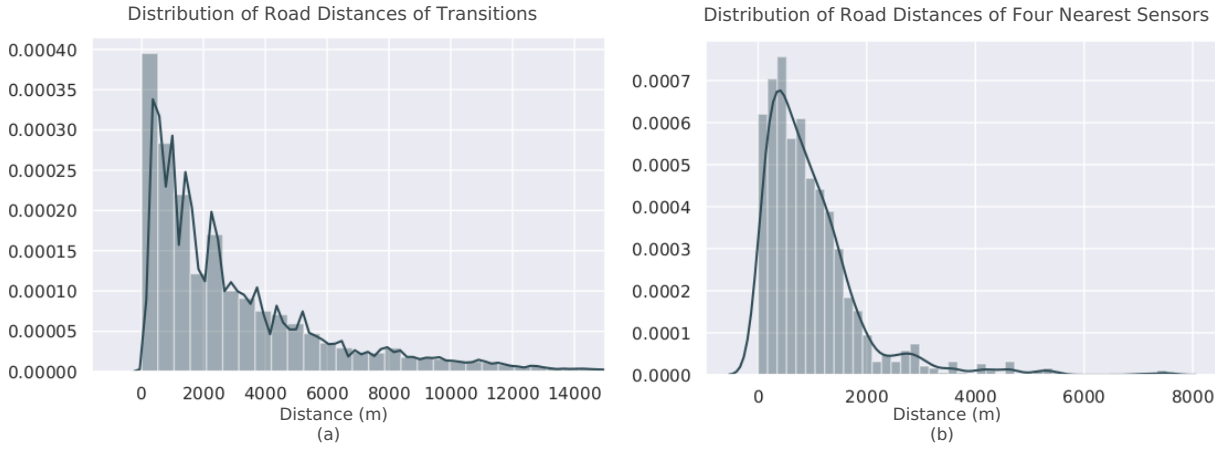


Figure 3 – (a) KDE of road distances between transitions on the trajectories. (b) KDE of road distances between the 4 nearest sensors.

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 3 presents the percentiles of the number of transitions for each sensor. We observed that 30% of the sensors do not appear 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., 70^{th} percentile/#transitions) of all transitions.

We made the same analysis by removing the sensors without any departure transition as presented in Table 4. 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.

Table 3 – 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	9,496	25,616	59,189	247,768

Table 4 – Percentiles and maximum value for the numbers of transitions starting at each sensor with transitions

PCTL	10	30	50	70	90	Max
Value	6,906	9,869	22,881	39,064	77,635	247,768

3.4 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 hypothesis that the drivers usually prefer to choose the shortest paths (HENDAWI *et al.*, 2013), 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 5 – 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

Definition 4 (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 5 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 5 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.

Figure 4 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.

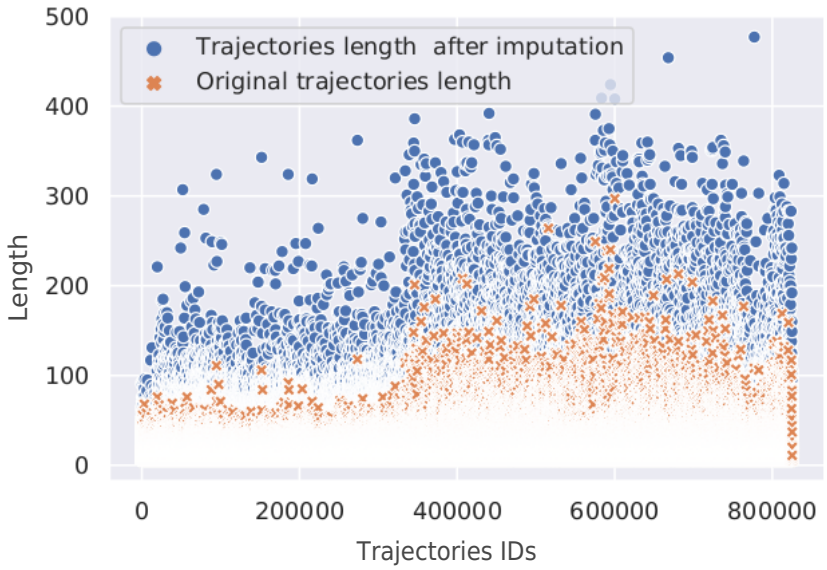


Figure 4 – Original lengths and lengths after imputation of each trajectory according to the number of sensors.

3.5 Discussion

In this chapter, we introduced the problem of predicting the next location for external sensor trajectories. We applied an analysis which evidenced the sparsity and incompleteness of these trajectories. We estimate the amount of missing data by applying an imputation approach based on the road network.

4 PREDICTION FRAMEWORK FOR EXTERNAL SENSOR TRAJECTORIES

This chapter is conducted by the research questions presented in Chapter 1 that are listed below.

- **RQ1** - *How to adapt prediction models to incompleteness and sparsity of EST?*
- **RQ2** - *How to predict the next location from EST?*
- **RQ3** - *How the different data imputation strategies impact location prediction models?*
- **RQ4** - *How to access the quality of predictions given the fact that the ground truth could be missed and thus unknown?*

We present the strategy for EST prediction consisting of three overall steps. The first one is Data pre-processing, which aims at pre-computing the sensor pairwise distances and simplifying the representation of trajectories by clustering the sensors on the road network. The second is the Learning step which includes data imputation and 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. Figure 5 depicts the general framework for EST prediction. Complementary to the results in this chapter, we implement a demonstration tool, briefly presented in Appendix A and published in (CRUZ *et al.*, 2019a). Furthermore, we discuss and present complementary experiments to evaluate some approaches to update the models in Appendix B.

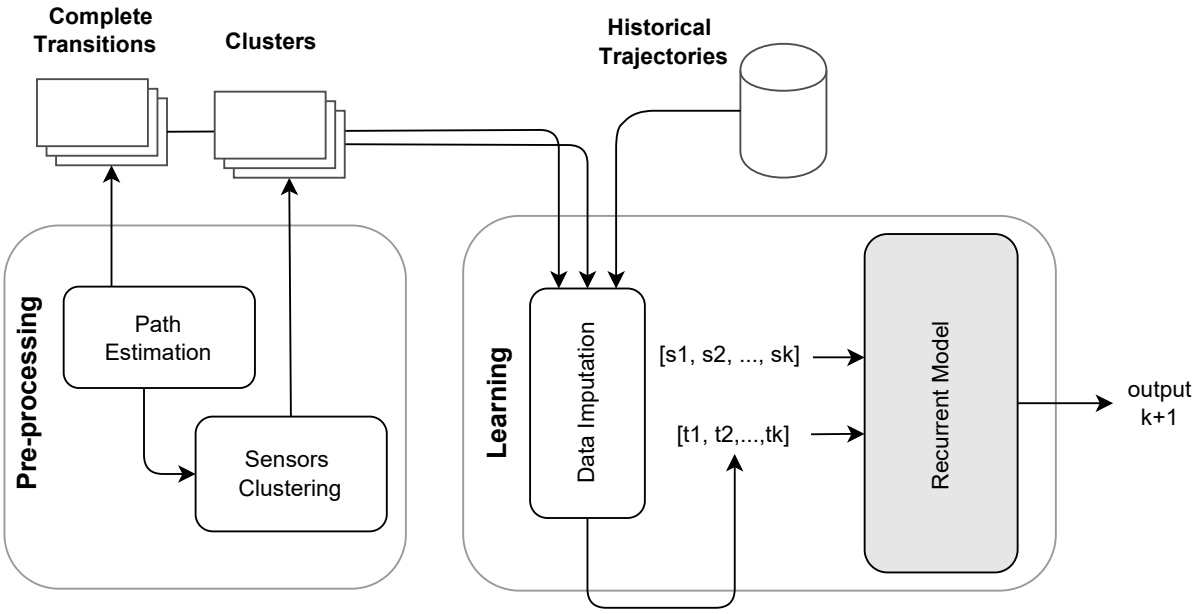


Figure 5 – General proposed framework.

The rest of this chapter details the solution and experimental evaluation, and it is ar-

ranged in the following way. Section 4.1 explains the data pre-processing we applied. Section 4.2 refers to **RQ1** and explains how we integrate the data imputation with the learning process. Section 4.3 is aligned to **RQ2** and explains the recurrent models used for EST prediction, including the single-task and the multi-task models. In Section 4.4, we show the experimental evaluation, which investigate the research questions **RQ3** and **RQ4**. Finally, Section 4.5 summarizes the chapter.

4.1 Data Pre-processing

The pre-processing steps we consider are path estimation to complete missing data, clustering to reduce the impact of the skew distribution of data, and sliding window.

4.1.1 Data Imputation and Clustering

One of the main reasons for the prediction errors is the frequent failure of some sensors in capturing the passage of moving objects. To compensate, 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 4) between every pair of different sensors and maintain them in a data structure 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 that 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 Figure 6. Once we have the clusters, in 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 (ESTER *et al.*, 1996) 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.

The reason is to avoid training the model on short sequences, which would bias the

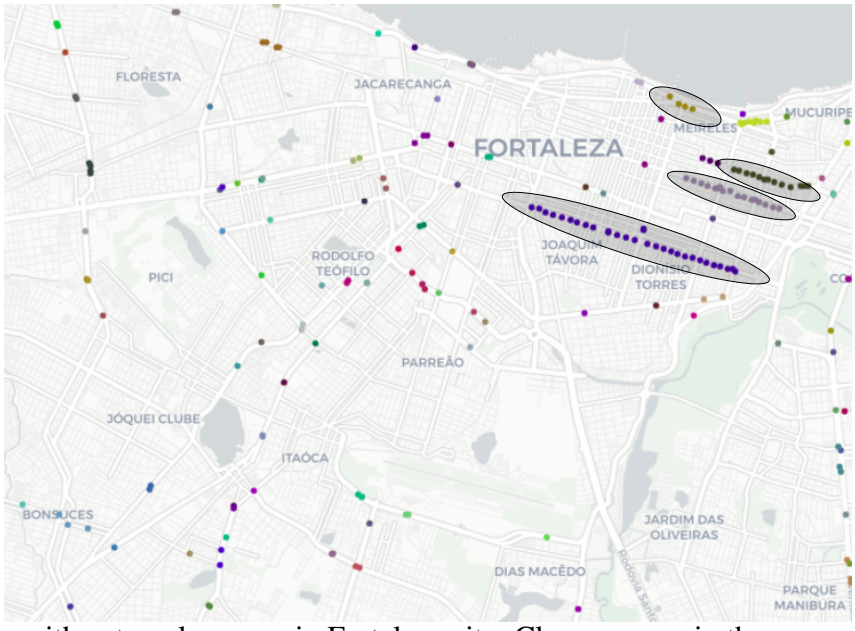


Figure 6 – Map with external sensors in Fortaleza city. Close sensors in the same color represent the clusters. Inside ellipses, some corridors identified by the clustering.

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 visited at least six sensors.

4.1.2 Sliding Window

In the training process, the *Sliding Window* technique was used to transform the trajectories into sequences of fixed length. The sliding window examines each sequence by advancing one position until it reaches the end. Points inside the window are defined as *features*, and the position immediately outside the window is used as *label*. Figure 7 illustrates this process for a window of size $m = 5$.

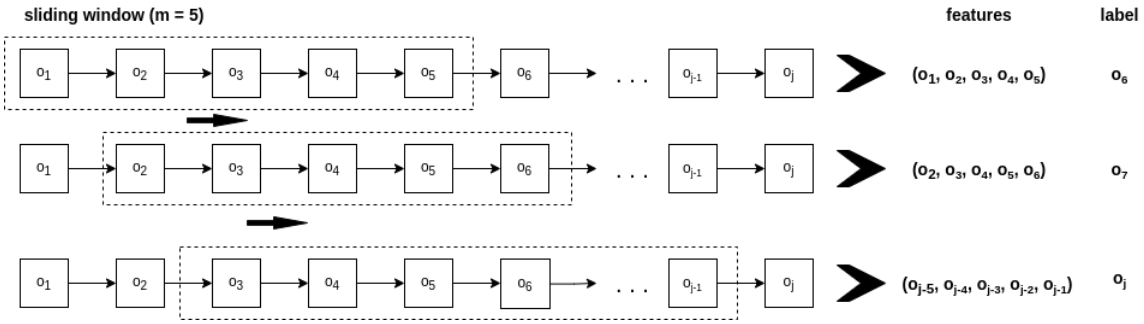


Figure 7 – Sliding window technique. Adapted from (WU *et al.*, 2017a).

4.2 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 $sub\_tr = \langle o_i, o_{i+1}, \ldots\ o_{i+k-1} \rangle$, 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 8 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 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.

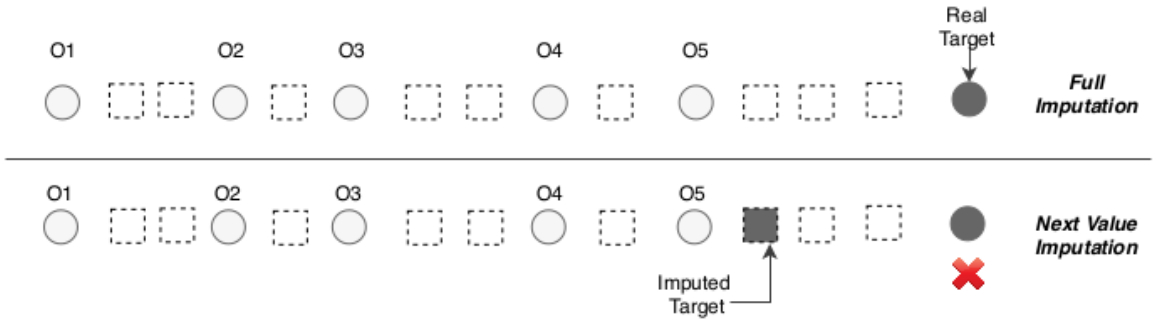


Figure 8 – Data imputation strategies for training. Circles are real observations captured by the sensors. Dashed squares are imputed data. The dark circle is the next sensor to be predicted.

Data imputation in 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}.

4.3 EST Prediction Models

In this work, the recurrent model receives a window of the w last sensors observations. Each observation has two features: the spatial feature, which corresponds to the sensor label; and the temporal feature which is the time slot in a day when the external sensor observation occurred. For instance, the daytime can be sliced into intervals of 15 minutes. Our approach represents each feature using the one-hot encoding 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 $\mathbf{l}_k = \left[l_{k_0}, l_{k_1}, \ldots, l_{k_n}\right]$, 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 $\mathbf{t}_k = \left[t_{k_0}, t_{k_1}, \ldots, t_{k_m}\right]$, 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 architectures, it is worth mentioning 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.

4.3.1 Recurrent Neural Networks

The solutions proposed in this chapter are based in a specific machine learning model called Recurrent Neural Network (RNN). RNN composes a class of neural networks that work on sequences of arbitrary lengths. Sequential data has a special characteristic given by the fact that the order in which the instances appear in the sequence matters. Different of Multilayer Perceptron (MLP) networks which process each instance of data independently of the previous one, RNN are able to remember past information in the sequence and update the network weights considering this past information.

Figure 9 presents the basic architecture of a RNN. It is composed by the input layer ($\mathbf{x}$), the hidden layer ($\mathbf{h}$), and the output layer ($\mathbf{y}$). We consider the input of a RNN a sequence with size n denoted by $\mathbf{x} = (x_{(0)}, x_{(1)}, x_{(2)}, \ldots, x_{(n)})$, where $x_{(t)}$ indicates the instance order in the sequence, i.e., $x_{(t)}$ is the instance that appears at time step t. At each time step t, the hidden unit $\mathbf{h}^t$ receives both, the value of the input sequence at time t ($\mathbf{x}^t$) and the output of the hidden layer on the previous step, $\mathbf{h}^{t-1}$.

The linear transformations in the RNN utilize the weight matrices $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{hy}$. $\mathbf{W}_{xh}$ is applied in the linear transformation between $\mathbf{x}^t$ and $\mathbf{h}^t$, $\mathbf{W}_{hh}$ is used in the recurrent edge and $\mathbf{W}_{hy}$ does the linear between the hidden layer and the output layer. The activation of a hidden unit $\mathbf{h}^{(t)}$ is given by $\mathbf{h}^{(t)} = \theta_h(\mathbf{z}_h^{(t)})$, where $\theta_h(.)$ is the activation function of the hidden layer and $\mathbf{z}_h^{(t)} = \mathbf{W}_{xh}\mathbf{x}^{(t)} + \mathbf{W}_{hh}\mathbf{h}^{(t-1)} + \mathbf{b}_h$. Finally, the output $\mathbf{y}^{(t)}$ at time step t, is calculated as $\mathbf{y}^{(t)} = \theta_y(\mathbf{W}_{hy}\mathbf{h}^{(t)} + \mathbf{b}_y)$, where $\theta_y(.)$ is the activation function of the output layer. A version of the Back-propagation Algorithm, called Back-propagation Through Time (BPTT), is used to train the network.

4.3.2 Single-task Model

In the single-task model used in our framework is basically the SERM (YAO *et al.*, 2017a) architecture simplified, where we substituted the POIs information with sensor identifiers and dropped the semantic attributes and user preferences.

Figure 10 shows the architecture, described hereafter.

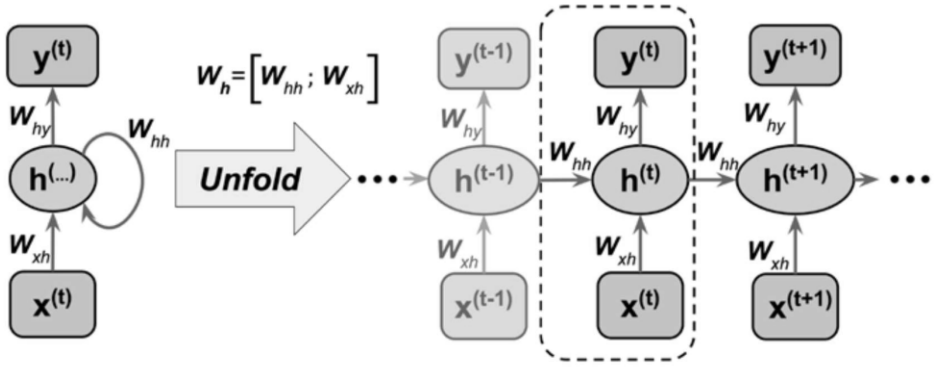


Figure 9 – Rolled and Unrolled visualization of a RNN. Reprinted from (RASCHKA; MIRJALILI, 2017).

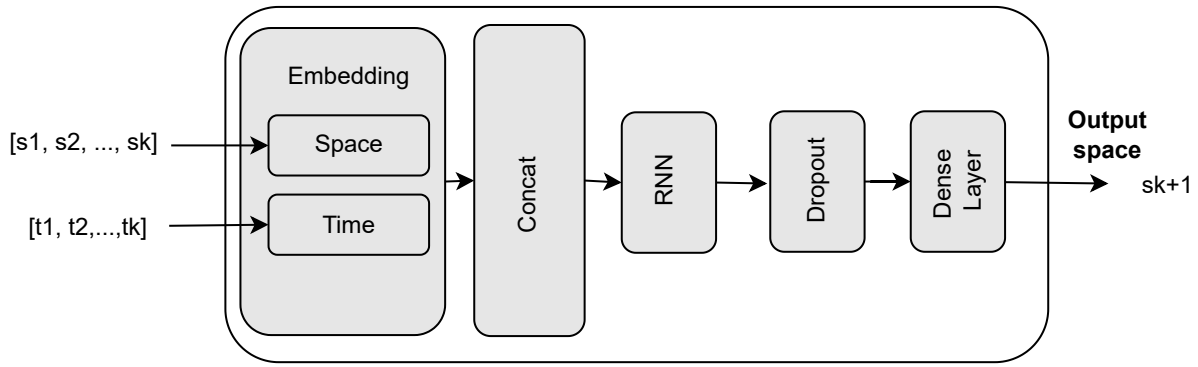


Figure 10 – Single-task architecture for EST prediction model.

1) An embedding layer, that applies a linear transformation on each 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 concatenates the embedding layers' output to get a unique input feature vector for the next layer.
3) A recurrent layer to learn the complex patterns from sequences.
4) The recurrent layer is connected to a dropout layer, to prevent over-fitting, followed by a fully connected layer with Softmax as an activation function. The function converts the result of the recurrent layer into the set of probabilities to be assigned to each class label. Here the number of classes is the number of sensors (or clusters) in the output space layer.

4.3.3 Multi-task Model

The multi-task RNN learns the time and space features to improve the sequential prediction for location. Using the time and space information in the training phase, our approach learns more meaningful representations which boost the learning performance of EST prediction.

Figure 11 shows the multi-task architecture, described hereafter. This multi-task learning architecture (CRUZ *et al.*, 2021) differs from the primarily proposed single-task (CRUZ *et al.*, 2019b) by including a new recurrent layer connected and a densely connected layer that learns the time slot. These layers appear in Figure 11 in the black boxes.

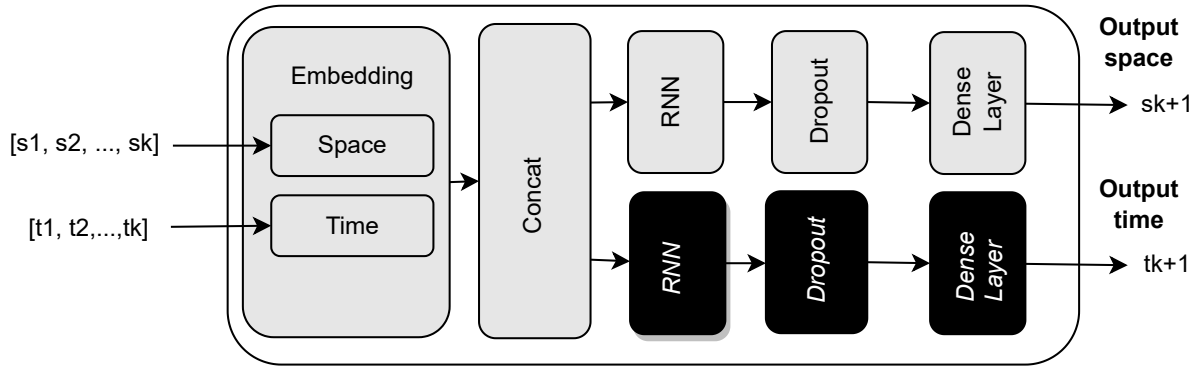


Figure 11 – Multi-task architecture for EST prediction model.

1) An embedding layer and a concatenate layer, as described in the single-task solution.
2) Two recurrent layers in parallel, one to learn sequential patterns for prediction of space and another one to learn sequential patterns for prediction of time. Each recurrent layer receives the output of the concatenation layer.
4) As in the single-task model, each recurrent layer is connected to dropout layer followed by a fully connected layer with Softmax as an activation function. 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 (CRUZ *et al.*, 2021) differs from the primarily proposed single-task (CRUZ *et al.*, 2019b) by including a new recurrent layer connected and a densely connected layer that learns the time slot. These layers appear in the black boxes in Figure 11.

In the multi-task model, 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. In our experiments, the multi-task learning approach reached 85.20% of accuracy on the clustered sensors model, more than 20% of the best strategy from the single-task approach.

4.4 Experimental Evaluation

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

4.4.1 Data and Models

For the purpose of experimental evaluation, we consider the dataset collected from a real application of traffic monitoring, described in Section 3.3.

We use as the baseline the single-task pure RNN scheme, an simplification of (YAO *et al.*, 2017b) adapt for our context, named as Basic (B). 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), which considers each possible combination of location label and time slot as a state in the Markov model.

The other models evaluated in our experiments are: 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 mentioning that B, BC, NI, FI, NIC, and FIC are single-task learning models. Our multi-task learning approach has some variants, called here: Multi-Task Basic (MT-B) - the basic multi-task schema; Multi-Task Basic with Clustering (MT-BC); 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 and Clustering (MT-FIC). It is worth mentioning that the RNN models were implemented using (CHOLLET *et al.*, 2015). For the experiments with Markov Models, we used the implementation available in (FOURNIER-VIGER *et al.*, 2016).

In our experiments, we used a $w = 5$ 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 minutes, 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 (KINGMA; BA, 2014) as the optimizer, Cross-entropy as the loss function, and Softmax as the activation function on the last layer.

4.4.2 Evaluation and Metrics

We evaluate the models 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 $(1,2,3,4,5)$ kilometers.

The methods were evaluated concerning two metrics: *accuracy* and *closeness error*. The accuracy is the percentage of correct predictions (Equations 4.1 e 4.2).

$$accuracy = \frac{\sum_{i=1}^{N} I(l_i, \overline{l_i})}{N} \tag{4.1}$$

Where l_i is the label of the observed next location, $\overline{l_i}$ is the label of the predicted next location and N is the number of predictions.

$$I(a,b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

The closeness error is a measure of the proximity of the observed value to the predicted on the road network (given by the road distance). 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 to the expected sensors. We evaluate the closeness error of models by taking their percentiles on the test data.

$$closeness\_error(l_i, \overline{l_i}) = d_{RN}(p_i, \overline{p_i}) \tag{4.3}$$

Where p_i and $\overline{p_i}$ are the geo-referred points of locations l_i and $\overline{l_i}$, respectively, $d_{RN}(p_i, \overline{p_i})$ is the size of the shortest path from p_i to $\overline{p_i}$.

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.

4.4.3 Evaluation of Accuracy

Table 6 – Comparison of Accuracy between the single-task models and the multi-task models

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**

In this section, we study how the single and multi-task learning models perform in terms of accuracy. Table 6 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.

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 7 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 which 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 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 7 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 7 – Accuracy of the approaches that use the clustering strategy when *eps* varies.

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**

4.4.4 Evaluation of Accuracy with Free Simulation

With data imputation, we assume that the next sensor may be the imputed data and not the observed data thus is acceptable when the model does not predict the next location directly. To compensate, we evaluate the imputation strategies on single-task models using the free simulation approach. The free simulation approach takes the prediction made by the model to generate a new input. Supposing the model receives vectors with size k, we discard the first

observation from the previous input and create a new one to maintain the same size. We assumed constant speed and estimated the timestamp of the new entry based on road distances. Then, we use the new input to make a new prediction. The method was repeated for three iterations. To define the number of iterations on the free simulation, we take into account more than 25% of all transitions have at least two imputed sensors observed. Thus we expect to find the originally observed sensor until the third iteration. We compute the accuracy of the clustering-based approaches by evaluating if the observed sensor belongs to the predicted cluster.

The results are presented in Table 8. The accuracy of models trained based on the next value imputation (NI, NIC) does not considerably increase with the iterations of free simulation, which means these strategies find the correct value at the first iteration in most of the cases. However, FI improved the accuracy in about 14% and FIC in about 10% until the third iteration.

Table 8 – Comparison of the Accuracy

Iteration	NI	FI
1st	32.26	41.28
2nd	35.46	52.10
3rd	36.38	55.95

Table 9 – Accuracy of Next Value Imputation with Clustering (NIC)

eps (km)	1	2	3	4	5
1st Iteration	34.43	38.07	43.98	52.96	**65.84**
2nd Iteration	35.426	39.18	44.98	54.69	**67.32**
3rd Iteration	36.39	40.87	46.12	56.67	**67.93**

Table 10 – Accuracy of Full Imputation with Clustering (FIC)

eps (km)	1	2	3	4	5
1st Iteration	**41.14**	40.33	38.62	35.43	30.23
2nd Iteration	**48.27**	47.24	46.98	43.86	37.31
3rd Iteration	**51.54**	50.42	50.91	46.08	40.07

4.4.5 Evaluation of Closeness Error

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 90^{th}

percentile. For the clustering-based strategies, we report the closeness error for *eps* value with the best-achieved result.

Figure 12 reports the percentiles of closeness error obtained for the single-task learning models. The approaches NI and NIC (*eps=5*) 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 models. NI achieved 40% of all predictions with closeness error less than 105 meters. 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 meters. Among the other models, the best ones are BC and FIC with 90% of all predictions less than 6.3 km of closeness error. The results showed that the next value imputation and clustering pre-processing affected the results positively regarding closeness. Figure 13 presents the percentiles of closeness when we vary the *eps* value in the NIC models. We can observe that NIC can find better closeness error than the one presented in Figure 12. With $eps = 1$, NIC obtained 90% of all predictions less than 1.8 km. Figure 14 presents the percentiles for FIC model for different *eps*. The quality of this approach considering closeness for different *eps* values was almost the same, without improvements. After the median, 1st-Markov and 5th-Markov obtained the worst results for all percentiles.

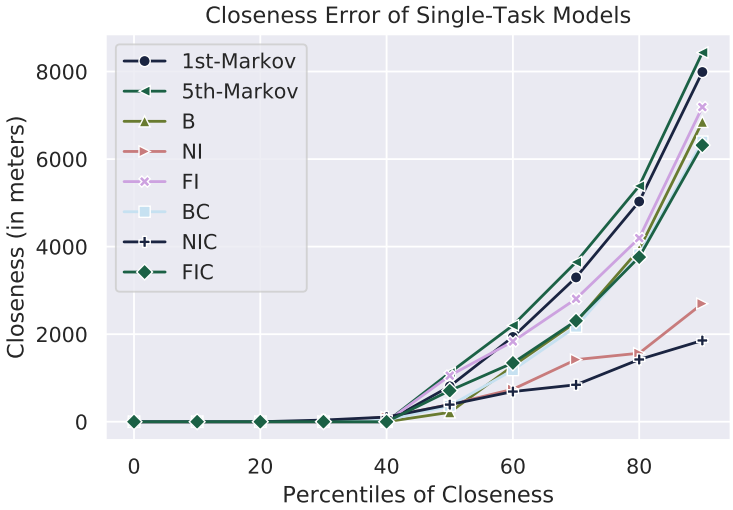


Figure 12 – Comparison of closeness error of single-task models.

Figure 15 presents the results of closeness error for the multi-task learning models. 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.

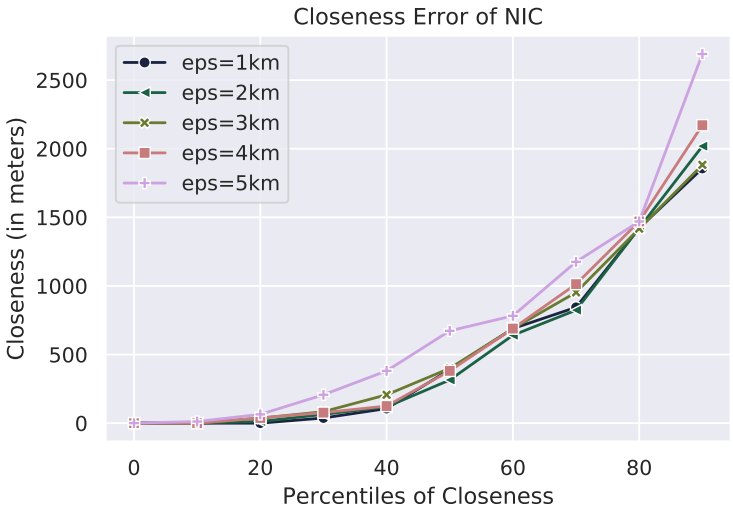


Figure 13 – Comparison of closeness error of NIC when *eps* changes.

Figure 14 – Comparison of closeness error of FIC when *eps* changes.

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 80^{th} 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.

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 16 and 17 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 70^{th} 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 meters 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 90^{th} percentile is 3.3 km.

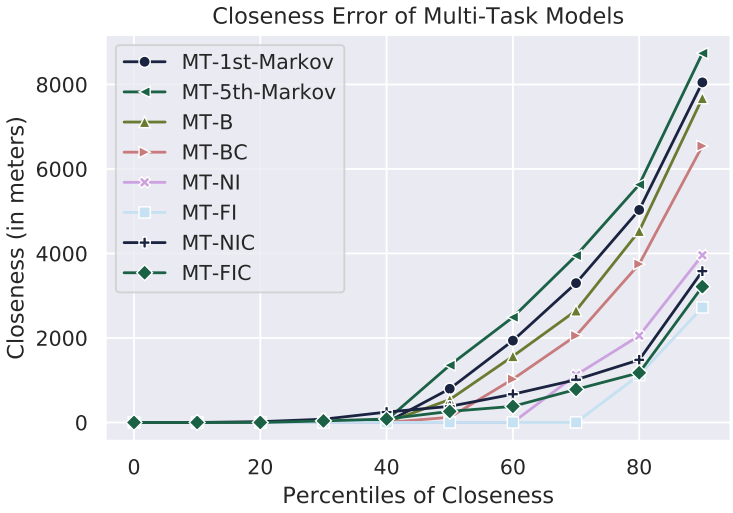


Figure 15 – Comparison of closeness error of multi-task models.

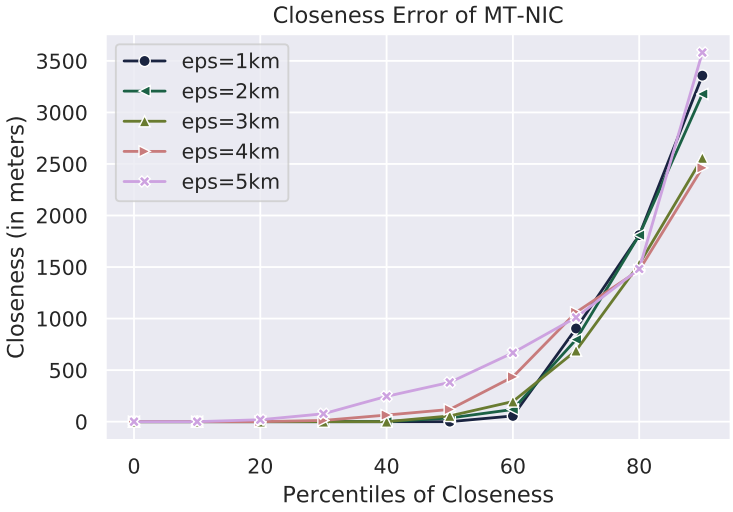


Figure 16 – Comparison of closeness error of MT-NIC when *eps* changes.

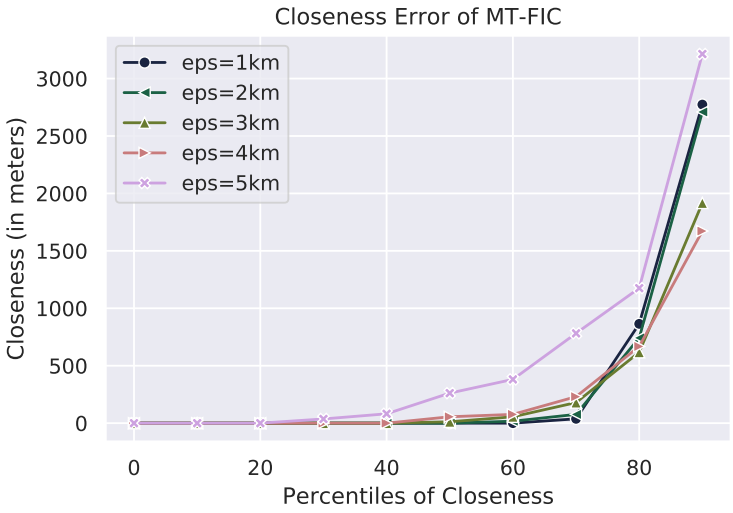


Figure 17 – Comparison of closeness error of MT-FIC when *eps* changes.

4.4.6 Experiments Discussion

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 which 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 90^{th} 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 70^{th} 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 sensor labels, which helps to improve the accuracy. As smaller the number of sensors labels, the model has more ability to learn from them. Also, clustering the sensors leads 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 groups 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 Figure 3, 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.

4.5 Discussion

In this chapter, we investigated an approach to deal with the missing data of external sensors. We propose a single-task and multi-task learning schema based on RNN for EST prediction. The multi-task schema learns both time and space information in order to solve EST prediction. We evaluated the proposed models using real-world data. The experiments showed that the clustered multi-task method could increase the accuracy by about 42% compared to the Markov Model and about 37% compared to the basic RNN. Also, we discussed the precision of the approaches concerning the closeness error. The experiments have shown that all the multi-task approaches with data imputation could reduce the overall closeness error. Finally, we present a demonstration tool of our approach.

5 REPRESENTATION LEARNING FOR EXTERNAL SENSOR TRAJECTORIES

In this chapter, we consider the problem of learning representation models for vehicles' trajectories in a road network observed by sensors located on sparse fixed points on the roadside. We assume that each trajectory is generated by a driver and captured by external sensors (e.g., traffic surveillance cameras).

Representation learning for trajectory modeling is concerned with building statistical or machine learning models of observed trajectories of vehicles or people. Such models may have different uses: computing the probability of observing a given trajectory for anomaly detection (JI *et al.*, 2020); estimating the importance of different characteristics that drivers may consider relevant when following a trajectory; recovering sparse or incomplete trajectories as the ones observed from external sensors (WU *et al.*, 2016); aiding drivers to choose an optimal route from an origin to a destination, or predicting online the next location of a vehicle given its current location (FENG *et al.*, 2018), which is the application study in this paper to learn representations for trajectories.

A driver's recent trajectory could help us predict the next location (e.g., sensor) he will pass by. A partial trajectory is a sequence of sensors $< s_1, s_2, \ldots, s_t >$, in which each sensor s_i is located in a fixed location on the side of a road in the road network. A noteworthy line of research is to build a representation learning for trajectories based on external sensor data aiming to predict the next sensor location. This problem has been called *External Sensor Trajectory Prediction* (EST Prediction for short) and has been investigated in other works (CRUZ *et al.*, 2021; CRUZ *et al.*, 2019b; NETO *et al.*, 2021).

We take inspiration from NLP word representation methods that model the semantics of a word, and its similarity with other words, by observing the many contexts of use of the word in the language. Geometric distances between word vectors reflect semantic similarity and difference vectors encode semantic and syntactic relations (MIKOLOV *et al.*, 2013c).

Papers from state-of-the-art include the adoption of neural embeddings to model locations, points of interest (PoIs) (CAO *et al.*, 2019; CRIVELLARI *et al.*, 2022; DAMIANI *et al.*, 2020) or to learn the temporal interactions between users and items (KUMAR *et al.*, 2019). The papers (CRUZ *et al.*, 2021; CRUZ *et al.*, 2019b; NETO *et al.*, 2021) also focus on the EST prediction task and assume that a road network constraints object movements. Their trajectories report predefined positions (i.e., sensors' location) on the road network. This assumption turns the EST prediction problem different from (i) the papers that consider GPS trajectories, which

occur at any spatial location (ROCHA *et al.*, 2016; FENG *et al.*, 2018); and (ii) the next (stop) location prediction, usually applied to points of interest or event places as (KARATZOGLOU *et al.*, 2018b), (YAO *et al.*, 2017b), (HASAN; UKKUSURI, 2017) and (WU *et al.*, 2017c) since the prediction involves movement rather than only stops. The main difference between this work and others is that none investigated whether the embeddings encode the relations between the sensors in terms of distance and connectivity in the road network. Neither investigates the learning representations to leverage the EST Prediction models.

The research questions already presented in Chapter 1, and listed following, guide this study. Here, we focus on investigating natural language models to represent the trajectory sensors and exploring whether the prediction model improves.

- **RQ5:** Could NLP embedding models, more specifically, language models and word embeddings, be used to represent the vector space of trajectories in location prediction tasks using recurrent architectures?
- **RQ6:** Are the representations of sensors/locations from representation models in NLP able to capture their context, i.e. the closest sensors/locations, both in terms of road distance and connectivity?
- **RQ7:** Could trajectory representations from NLP embedding models adequately capture trajectories similarity?

We can observe a substantial similarity between natural language and EST. First, natural language and EST can be approximated by context. We can predict the next word in the natural language given the context. Likewise, a road network constraints the movement of drivers, so their observation only occurs at fixed (predefined) positions (i.e., sensors' location) on the road network. Thus, given a sequence of external sensors crossed by a driver, we can predict the next sensor on the driver's trajectory.

Both domains can be viewed as time-dependent series. Regardless of the language, there can be multiple choices from the dictionary to succeed a word in a sentence. Likewise, there are various choices of paths in the driver's trajectory; therefore multiple possibilities for external sensors observation.

To motivate the use of natural language models in the representation learning of trajectory sensors, we verify that the frequency of trajectory sensors approximately follows a Zipf Law. The Zipf Law controls the word frequency distribution in natural language (ZHANG *et al.*, 2014). In (CAO *et al.*, 2019), the authors observed the same Zipf's Law behavior on human

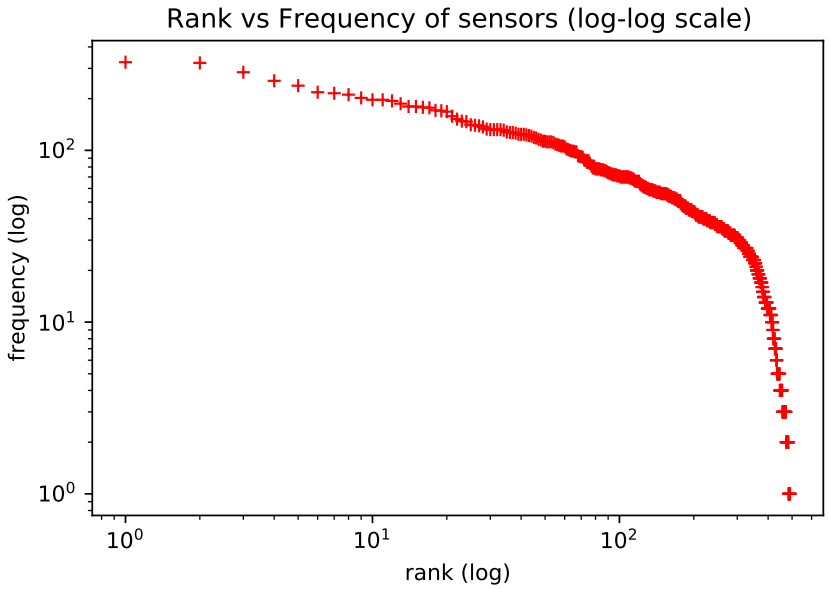


Figure 18 – Statistics of sensor observations satisfies the Zipf's Law.

mobility habits and used this analogy to apply natural language models to learning representation for living habits. Figure 18 (observed in the dataset utilized in our experiments) shows the distribution of sensor observations, which roughly follows Zipf's Law as expected. In the plot, axis y is the frequency in which each sensor appears in trajectories; axis x is the frequency rank, where rank k is the kth most frequent sensor.

Different from previous works, we study how to extract effective representations for sensors and trajectories in terms of distance and connectivity in the road network. We demonstrate that our derived representations leverage the task of EST prediction for a recurrent network model.

The rest of this chapter is structured as follows. Section 5.1 discusses the data preparationn, the NLP methods for obtaining embedding vectors, and the next location prediction architecture we use to achieve our goals. Section 5.2 presents and discusses the experimental evaluation. Section 5.3 concludes this work and suggests future developments.

5.1 Data and Methods

This section explains the methodology used in this work to investigate the use of representation learning architectures for NLP on external sensors trajectories.

5.1.1 Data Preparation

We use the same data set, and pre-processing steps of (NETO *et al.*, 2021). The trajectory data is from a real traffic monitoring application, which has 489 sensors monitoring vehicles at Fortaleza city in Ceará, Brazil. The dataset was collected from January 2019 to June 2019. It is composed of 22,530 sensor observations of 2,990 stolen vehicles. Each collected record is comprised of the vehicle license plate, timestamp, latitude, longitude, and speed. The sensor identification is inferred by its location.

We segment the trajectories in order to identify trips. Firstly, the trajectories were segmented by day, where the collection of points from the same vehicle on the same day composed one trajectory. Using the result of this segmentation, we compute the average time and standard deviation of the time elapsed between successive observations. Then, when the time of displacement between two consecutive sensors was bigger than the average time plus two standard deviations, we apply the segmentation in the daily trajectories. Finally, we maintain in the dataset only trajectories bigger than two records.

In the same way, as explained in Section 3.3, we complete the transitions between two consecutive sensors according to the shortest path. In other words: If sensor s_j appears after sensor s_i in a trajectory, we assume that all sensors on the shortest path from s_i to s_j are missing sensors; thus, we complete the trajectory considering these sensors. Finally, we keep only the sensors that occur more frequently than average in the training set.

5.1.2 Experimental Methodology

To apply NLP approaches, we firstly represented the trajectories as sequences of labels of sensors. Since we want to evaluate the potential of those methods to capture the spatial representation, we discard the temporal feature. We label each sensor using a geocode hash function applied to its geolocation.

Using these trajectories, we train a text representation learning model to learn a representation of labels' sensors. Doing a mapping to the text representation, we consider one sensor label as one word, a trajectory as a text, and the set of trajectories as the corpus of documents. The output of this step is a set of embedding vectors, each one representing a sensor label.

We hypothesize that the vector embedding captures a relationship between sensors,

such as spatial proximity or frequent use together. Thus, we experiment with the embedding vectors using different embedding models from NLP, such as Word2Vec and BERT, as explained following.

We evaluate the quality of embedding vectors we obtained in two ways, using an extrinsic and an intrinsic evaluation. The extrinsic evaluation evaluates the embedding vectors when they are integrated into a real trajectory task, such as EST prediction in our case. The intrinsic evaluation tests the quality of representation independent of specific tasks, directly testing the relationships between trajectories. Our intrinsic evaluation considers the spatial relation for sensor embeddings and similarity search for trajectory embeddings.

5.1.3 Word Embedding Models

Word2Vec (MIKOLOV *et al.*, 2013b) is a representation learning framework for words based on neural networks. The Word2Vec framework presents two architectures: the Continuous Bag-of-Words (CBOW) and the Skip-gram. In both architectures, word vectors are trained based on a sliding window of n-grams on the corpus. In the training phase, the task is the prediction of words.

The CBOW model (Figure 19) is trained to predict the current word y_j based on the window context $(x_{1k}, x_{2k}, \ldots, x_{Ck})$. The matrix $\mathbf{W}_{V\times N}$ and The matrix $\mathbf{W}'_{N\times V}$ contain the weights between input and output vectors. Each entry x_{ik} represents one word in a V-dimension one-hot encoded vector. Observe that V is also the vocabulary size. Each row of $\mathbf{W}_{V\times N}$ is a N-dimension representation associated with a different word in the vocabulary. The hidden layer outputs the sum or the average of vectors extracted using $\mathbf{W}_{V\times N}$. The output layer applies the softmax activation on the vector extracted using the matrix $\mathbf{W}'_{N\times V}$ on the output of the hidden layer, which results in a V-dimensional vector where the position i give the probability of target be the word i. Skip-gram (Figure 20) follows a similar idea of CBOW, but different from CBOW, Skip-gram model seeks to predict the surrounding words into the sliding window $(y_{1j}, y_{2j}, \ldots y_{Cj})$ given the current word x_k. The main drawback of Word2Vec is that it is trained on a separate local context window, which does not allow representing the words based on the global context.

Transformers architecture (VASWANI *et al.*, 2017) introduced the self-attention mechanism. The self-attention is a sequence model able to create an embedding space of words that provides token representations based on the representation of the more relevant tokens for one word in the sequence. The Bidirectional Encoder Representations from Transformers

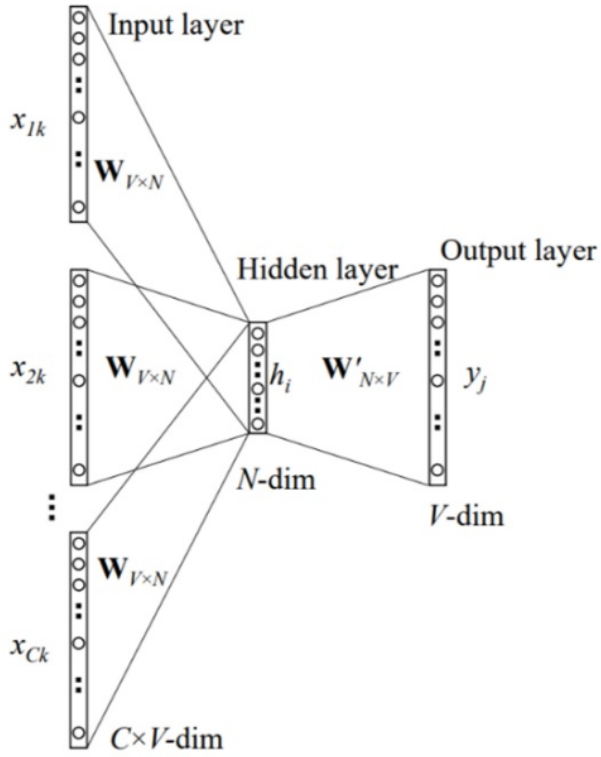


Figure 19 – Word2Vec CBOW architecture. Reprinted from (RONG, 2014).

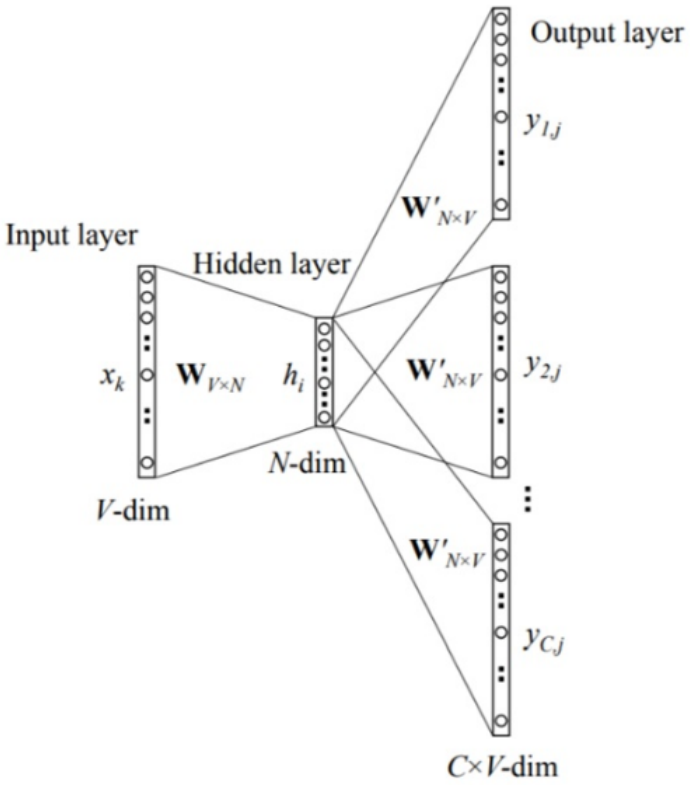


Figure 20 – Word2Vec Skip-gram architecture. Reprinted from (RONG, 2014).

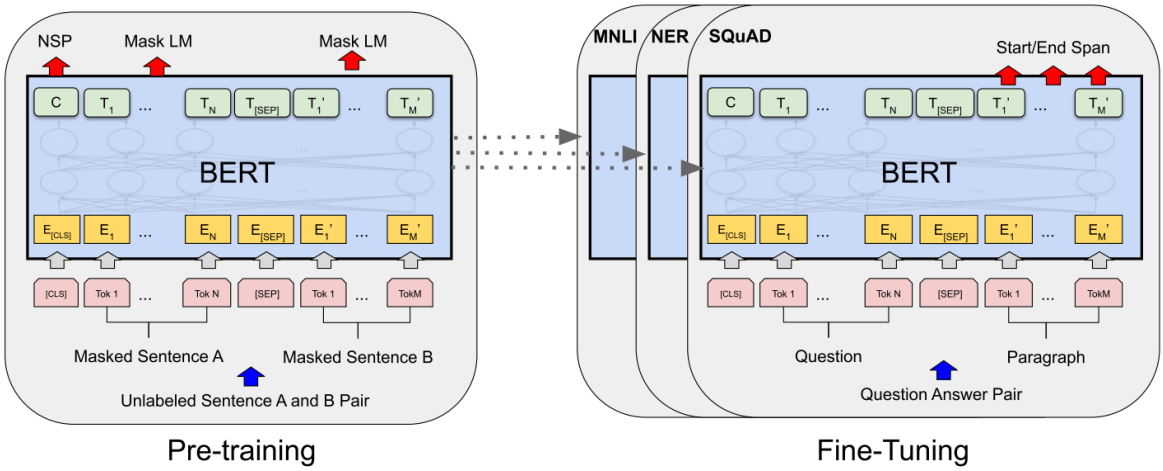


Figure 21 – Pre-training and fine-tuning procedures for BERT. Reprinted from (DEVLIN *et al.*, 2018).

(BERT) proposed by (DEVLIN *et al.*, 2018) is a multi-layer transformer encoder that mitigates the unidirectionality constraint of its previous language models based on transformers. Unidirectionality implies that the model can access only the previous tokens in the self-attention layers. Contrary to unidirectionality, BERT creates an embedding space of words considering the left and right context. BERT is designed to pre-train representations from unlabeled data (originally text data). Apart from the output layer, BERT architecture also allows the fine-tuning of representations for a specific task.

Figure 21 shows the BERT architecture for pre-training and fine-tuning representations. The pre-trained word representations of BERT can be obtained by using two different tasks, a masked language model (MLM) or a next sentence prediction. For both, the model receives a pair of unlabeled sentences A and B. The MLM model randomly masks a percentage of the input tokens in sequence A. During training, the goal is to predict the masked tokens on sequence A based on their left and right contexts. The next sentence prediction jointly pre-trains text-pair representations. The next sentence aims to predict sentence B given the previous sentence A. The result is the pre-trained representations for the tokens in a specific vocabulary.

In fine-tuning, BERT can model many different tasks, such as Named Entity Recognition (NER), question answering (QA), and possible continuation prediction. To specify BERT for a specific task, the model is initialized with the same parameters as the end-task model and the appropriate input and outputs are plugged into the model.

5.1.4 EST Prediction Model for the Extrinsic Evaluation

In our extrinsic evaluation of trajectory embedding vectors, we propose to use EST prediction as the main task. Our architecture is a recurrent-based model that uses the Long Short-Term Memory Network (LSTM).

As presented in Chapter 4, the Recurrent Neural Networks (RNN) (CONNOR *et al.*, 1994) compose a class of neural networks that work on sequences of arbitrary lengths, such as trajectories. On the computation of gradients by the BPTT algorithm for long sequences, the vanilla RNNs suffer from vanishing or exploding gradient problems. The Long Short-Term Memory Network (LSTM) model was proposed by (HOCHREITER; SCHMIDHUBER, 1997) to overcome this issue. An LSTM comprises several memory cells, each memory cell represents a hidden layer and has a recurrent edge with a value associated with it, called cell state. The LSTM cells have three components, named input gate, output gate, and forget gate, for controlling the error flow from the memory unit and its connections, and so, prevent the vanishing and exploding gradients.

Figure 22 illustrates the flow of information on a LSTM cell. Each box applies a linear combination on their inputs, and then, one activation function on the result; $\mathbf{C}^{(t-1)}$ is the cell state from the previous time step; $\mathbf{h}^{(t-1)}$ is the output of the previous hidden unit; σ and *Tanh* are the **Sigmoid** and **hiperbolic tangent** activation functions, respectively; $\odot$ is the element-wise product and $\oplus$ is the element-wise summation. The output of gates is processed by $\odot$. The forget gate **f** is used to decide whether to reset or not the previous cell state. The input gate **i** and the input node **g** are responsible to update the currently cell state, which is done by $\oplus$ operation. Finally, the output gate **o** updates the values of hidden units.

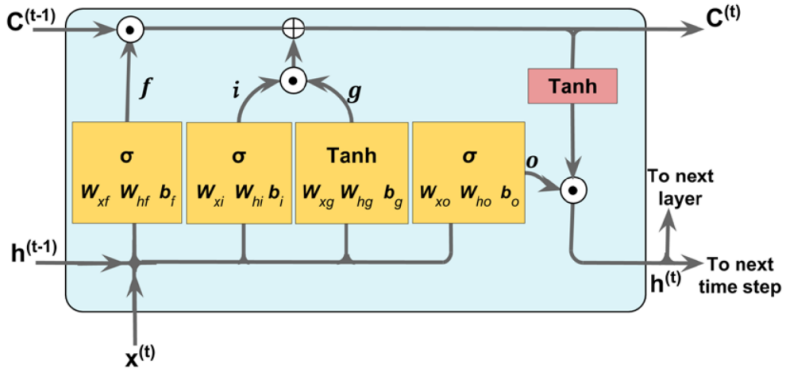


Figure 22 – Structure of an LSTM cell. Reprinted from (RASCHKA; MIRJALILI, 2017).

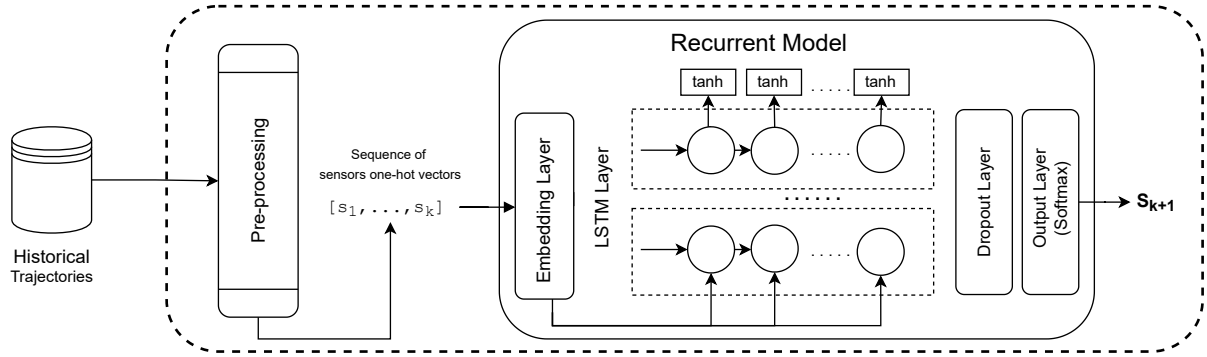


Figure 23 – LSTM EST prediction architecture.

The EST prediction model investigated, which was also based on our papers (CRUZ *et al.*, 2019b; CRUZ *et al.*, 2021; NETO *et al.*, 2021) is a recurrent based model. The first layer (1) is an **Embedding Layer**, which applies a linear transformation on the high-dimensional input vectors to reduce their dimensionality while trying to preserve the similarity between instances from the original space of features in the new space of features. The original representation of the input in our problem is the one-hot encoding representation, in which the dimensionality of the vectors is the number of existent sensors. A one-hot encoding represents categorical values as binary vectors, where all values are zero except the index of the categorical input, which is marked as one. The parameters of the embedding layer can be trained (or tuned) during the training of the other parameters in the neural network for the next location prediction or can be pre-trained using a representation learning model. The output of the embedding layer is sent to (2) a **LSTM Layer**, which is able to learn the moving patterns from the sequence of sensors. In sequence, we have (3) a **Dropout Layer** help to prevent the overfitting problems by hiding some input units randomly at each iteration of the training phase. Moreover, (4) an **Output Layer** is a fully connected layer. The activation function of the output layer is the **Softmax**. The Softmax function converts the output into a vector that indicates the probability of each sensor being the target. Figure 23 shows the EST prediction architecture utilized in this evaluation.

To train the model, we transform trajectories into fixed-length sequences of sensor labels. A sliding window scans each sequence. The window moves forward by one position until it reaches the end. We define the points inside the window as features, and the position immediately outside is used as a label.

5.1.5 Metrics for the Intrinsic Evaluation

Our intrinsic evaluation is based on the Reciprocal Rank (RR) measure. In Information Retrieval, the RR calculates the rank at which the first relevant document is retrieved. If the relevant document was retrieved at rank r, then RR is $1/r$ (CRASWELL, 2009). We adapt the definition of RR measure for our particular goal (Definitions 5 and 6).

Definition 5 (Reciprocal Rank) *Given a reference distance d_{ref}, a query distance d_{query}, a query object o and a set of objects O, the reciprocal rank of o with respect to the reference distance d_{ref} and the query distance d_{query} for the set O is $1/r$, if the closest object to o, $o^* \in O$ according by d_{ref} is at rank r according by d_{query}.*

Definition 6 (Mean Reciprocal Rank) *The Mean Reciprocal Rank (MRR) is the average of RR across a set of query objects.*

In our experiments, for the evaluation of sensor embedding vectors, the reference distance d_{ref} was set to be a spatial distance (Euclidean and road distances), while the query distance d_{query} was the cosine distance between embedding vectors.

The trajectory distances used in our experimental intrinsic evaluation are the Dynamic Time Warping (DTW) and the Edit (ED). DTW is one of the most popular trajectory distance measures. It searches for all possible alignment points between two trajectories to find the one with minimal distance. ED quantifies the dissimilarity between two sequences of strings by counting the number of operations to transform one string into another.

Given two trajectories T_1 and T_2, consider p_i as the first sensor of trajectory T_i and $tail(T_i)$ is the sub-trajectory that starts at the second point of T_i until its end, or is empty if the T_i has length equals to one. The DTW and Edit distances are calculated in a recursive approach. Equation 5.1 presents the DTW distance, and Equations 5.2 and 5.3 presents the Edit distance.

$$d_{DTW}(T_1,T_2) = \begin{cases} 0, & if\ n = 0 \text{ and } m = 0 \\ \infty, & if\ n = 0 \text{ or } m = 0 \\ d_{Euclidean}(p_1,p_2) + min\{d_{DTW}(T_1,tail(T_2)), & \\ \quad d_{DTW}(tail(T_1),T_2), & \\ \quad d_{DTW}(tail(T_1),tail(T_2))\ \} & otherwise \end{cases} \tag{5.1}$$

$$d_{ED}(T_1,T_2) = \begin{cases} n, & if\ m = 0 \\ m, & if\ n = 0 \\ min\{d_{ED}(tail(T_1),tail(T_2),+subcost(p_1,p_2), & \\ \quad d_{ED}(tail(T_1),T_2)+1, & \\ \quad d_{ED}(T_1,tail(T_2))+1\}\ , & otherwise \end{cases} \tag{5.2}$$

$$subcost(p_1,p_2) = \begin{cases} 0, & p_1 = p_2 \\ 1, & otherwise \end{cases} \tag{5.3}$$

For more details about trajectory similarity functions, we refer to the survey (SU *et al.*, 2020).

5.2 Experimental Evaluation

From here, we conduct our research by splitting it into three sets of experiments: i) Analysis of Location Prediction - Investigate whether NLP models have the potential of modeling the vectorial space of features for location prediction; ii) Analysis of Sensor Embeddings - Investigate whether sensors embeddings obtained from NLP models can capture the relationship between sensors locations; iii) Analysis of Trajectory Embeddings - Investigate whether the trajectories embeddings obtained from NLP models can capture the behavior of the whole trajectory.

5.2.1 Analysis of Location Prediction

In this section, we conduct an extrinsic evaluation to examine the research question ***RQ5:*** *Could NLP embedding models, more specifically, language models and word embeddings, be used to represent the vector space of trajectories in location prediction tasks using recurrent architectures?*

To extract the embedding representation of sensors, we evaluate both BERT MLM and Word2Vec. The embedding of words from NLP models was used as the embedding layer on the top of the LSTM-based model (as shown in Figure 23) for predicting the next location. The LSTM-based model was evaluated with and without fine-tuning on the embedding layer. The fine-tuning adjusts the weights of the embedding layer on the training step for one specific

task (location prediction in our case). We also evaluated the pure LSTM model as a baseline, i.e., LSTM with the embedding layer trained from zero.

In BERT MLM, part of trajectories sensors is masked, and the model is trained to predict the masked sensors considering the not-masked ones. We propose applying a simple strategy of data augmentation that works with a masked model. We replied to the training dataset a number of *n_replication* times. As in (DEVLIN *et al.*, 2018), we randomly chose the masked 15% of sensors on each replicated sample. In the case of Word2Vec, we applied CBOW architecture. The variation of hyper-parameters of BERT MLM and Word2Vec are shown in Table 11. For the Word2Vec model, we trained an LSTM using each parameter configuration and reported the result of the configuration with the best accuracy. For BERT MLM, we masked the last sensor in the validation data and collected the accuracy for each BERT MLM configuration. We trained an LSTM using the configuration which reported the best accuracy of the BERT MLM.

Table 11 – Word2Vec and BERT MLM Parameters.

Word2Vec	**BERT MLM**
embedding size = [16,32,64,128]	embedding size = [16,32,64,128]
n-grams = [4, 6, 8, 10]	sequence size = [32, 64]
	intermediate sizes = [64]
	hidden dims = [64, 128, 256]
	num hidden layers, num attention heads = [8]

Using the pre-trained embeddings, we train LSTM models to predict the next sensor given the previous m (we varied m according to the set [5, 7, 15, 31]. When $m = 5$, the model outperforms others) observed sensors on the same trajectory. The models were evaluated using *ACC@N* and *closeness_error* metrics. The *ACC@N* measures the percentage of instances where the correct prediction is among the N most probable outputs according to the estimation given by the model, using the softmax activation. The *closeness_error* is the road network distance between the predicted sensor and the expected one.

Models. For what concerns the models of our work, we consider the configurations: i) LSTM, the embedding layer was trained from scratch on the prediction task; ii) LBERT, the LSTM using as embedding layer the pre-trained embeddings of BERT MLM; iii) LBERT-FT, the pre-trained BERT embedding layer was fine-tuned under the next sensor prediction task; ii) LW2V, the LSTM using as embedding layer the pre-trained embeddings of Word2Vec; iii) LW2V-FT, the pre-trained Word2Vec embedding layer was fine-tuned under the next sensor

prediction task. Furthermore, we report only the results of the better parameters configuration.

Table 12 presents the results for the best model configuration of each strategy. The best accuracy for each different window size was reported in Appendix C. The LSTM predictor with the BERT embedding layer fine-tuned achieves the best result. The fine-tuning of the BERT embedding layer has improved the accuracy *ACC*@1 up to 7%, the same behavior was found for *ACC*@2 and *ACC*@3. We believe BERT represents the feature space of sensors for trajectories better because it learns representations by adjusting the weight of the context. In addition, the positional encoder of BERT captures sequential connexions from the road network. Also, the masked language model of BERT simulates and learns under the situation of missing data, where the sensor does not capture the passage of a moving object. The LSTM predictor using Word2Vec embedding reached the lowest accuracy, 52.14%, followed by its fine-tuned version. The fining tuning of the Word2Vec embedding layer has not improved the predictor significantly. One possible reason for this could be the fact that Word2Vec models do not learn sequential patterns, but only the surrounding context. The baseline LSTM achieves 66% of accuracy, these results are consistent with the ones obtained by (NETO *et al.*, 2021).

Table 12 – Accuracy of prediction models.

	ACC@1	**ACC@2**	**ACC@3**
LSTM	64.10	74.96	80.32
LBERT	66.84	79.70	85.05
LBERT-FT	**73.71**	**85.35**	**90.01**
LW2V	52.14	63.09	72.38
LW2V-FT	53.33	66.90	72.14

Table 13 reports the mean and percentiles of closeness error obtained by the models. The medians and lower percentiles were omitted since, for all models, their values were zero. The mean closeness error of LBERT-FT was the best one, with a value of 0.79km, followed by LBERT with 0.97km and LSTM with 1km. LW2V and LW2V-FT found the worst results, in this order. LBERT-FT obtained 80% of all predictions with errors under 1.3km. LBERT reached 70% of closeness error lower than 0.5km and 80% lower than 1.6km. We can conclude that BERT representation helped not only to increase accuracy but also to increase the proximity between predicted and expected sensors when the predictor fails. However, with fine-tuning for location prediction tasks, BERT representations could reach better results.

Table 13 – Mean and percentiles of the closeness error for predictions (in kilometers).

	Mean	**60**	**70**	**80**	**90**
LSTM	1.0	0.0	0.85	1.99	3.49
LBERT	0.97	0.0	0.56	1.67	**2.78**
LBERT-FT	**0.79**	**0.0**	**0.0**	**1.36**	**2.78**
LW2V	1.44	0.73	1.4	2.44	4.09
LW2V-FT	1.36	0.65	1.31	2.46	4.09

5.2.2 Analysis of Sensor Embeddings

These experiments are guided by the research question ***RQ6:*** *Are the representations of sensors/locations from representation models in NLP able to capture their context, i.e. the closest sensors/locations, both in terms of road distance and connectivity?*

Our goal is to evaluate if the spatial relationship impacts the degree of similarity of sensor embeddings. We evaluate the relationship between spatial and embedding distances and investigate how similar embeddings reflect the spatial proximity or connectivity in the road network. To achieve this goal, we use BERT MLM that leverages the best next location predictor from the experiments explained in the previous section (Section 5.2.1). BERT MLM is used to generate the embedding space of the sensors. Figure 24 exemplifies embedding vectors of sensors on space. The points (in red) are sensors, and the markers (in black) connected by a line represent a pair of nearest sensors according to the cosine distance between their embedding vectors.

In spite of the nearest embedding sensors vectors from BERT MLM is not directly the closest in space, we can see in Figure 24 that they are on roads with some connectivity and also in their spatial neighborhood; which suggests that BERT MLM could capture some spatial relationship between sensors.

To better understand the impact of spatial proximity on embedding similarity, we analyze the MRR (Definition 6) of sensors concerning these metrics. In these experiments, the reference distance d_{ref} was set to be a spatial distance (Euclidean and road distances), while the query distance d_{query} was the cosine distance between embedding vectors.

First, we collect the MRR using the Euclidean distance as d_{ref}, and Cosine distance as d_{query}. In that case, the MRR was 0.20, which means the nearest sensor according to Euclidean distance is around the $5th$ most similar on the embedding space, on average. Using the Road Network distance as d_{ref} and cosine distance as d_{query}, MRR was 0.17, so the nearest sensor

(a) Top-1 is at the same road, but it is not the closest in space.

(b) Top-1 is the closest in space and at the same road.

(c) Top-1 is not at the same road.

Figure 24 – Example of nearest sensors according to cosine distance between BERT MLM embedding vectors.

in the road network is, on average, around the $6th$ most similar on the embedding space. We believe that, on average, being in the top five or six is acceptable as we are learning a new space (embeddings) to represent the sensors, which is not a trivial task.

We also evaluate the variation of MRR for sensors inside a neighborhood using the following methodology. We calculate the RR for each sensor considering a spatial distance (both the Euclidean and the Road Network distances) as reference distance and the cosine distance between the embedding vectors as the query distance. The set of sensors to be ranked (O) were filtered out to have only the ones spatially close to the sensor considered as the query object, i.e., the ones that fall inside the distance range. We vary the range distance and collect the MRR according to the filtered sensors.

Figure 25 presents the result of this analysis for both Euclidean and road network distances. One can observe that among pairs of sensors far around 1.5km, the *MRR* is about 0.35, which means the closest sensor in space is at rank 2 or 3; when the filter of maximal distance increases, the *MRR* decreases. Even when the maximal distance is around 4km for Euclidean space, the *MRR* is more significant than 0.25, and for the road distance of approximately 4km, the *MRR* is 0.20. In other words, the rank given by the embeddings of the closest sensor is around 4 and 5 for euclidean and road distances, respectively. The embedding representation reached a rank slightly similar to the spatial rank for the set of sensors that are not spatially so far. We argue that an efficient model for sensor representation does not necessarily reflect only the spatial proximity but also the connectivity in terms of road connectivity and frequent paths. Overall, we note from the previous research question that BERT sensor embeddings achieved better results for the next location prediction task. Furthermore, the BERT embeddings tend to reflect more spatial similarity when considering the sensor's neighborhood limited to a distance.

5.2.3 Analysis of Trajectory Embeddings

In this section, we investigate the research question **RQ7:** *Could trajectory representations from NLP embedding models adequately capture trajectories similarity?* by evaluating the similarity between trajectory embeddings regarding the similarity between raw trajectories.

We define trajectory embedding as the average vector of the embedding of sensors comprising the trajectory. As in Section 5.2.2, we use BERT MLM to generate the embedding representation of sensors. Figure 26 exemplifies different pairs of the most similar trajectories in the test set according to the cosine distance of their embedding vectors. Consider the trajectory

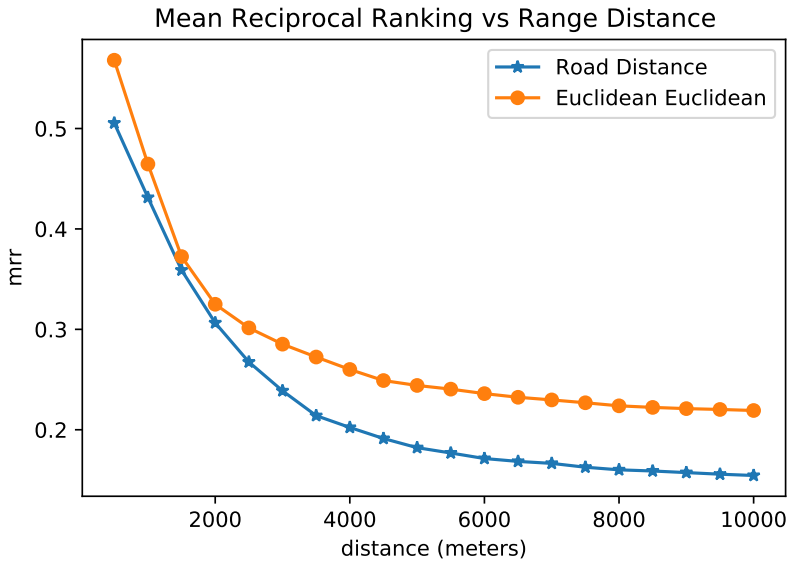
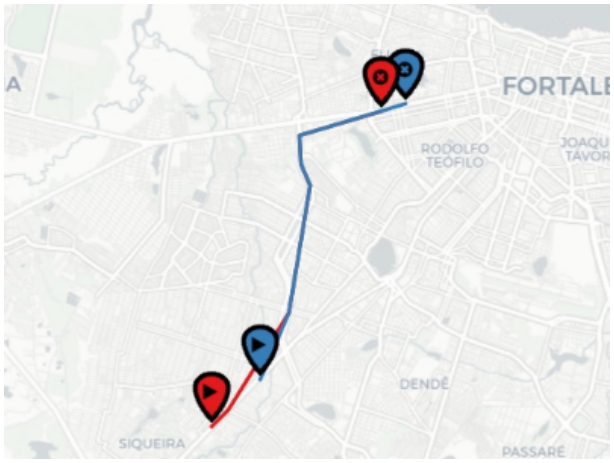


Figure 25 – Comparison of the mean reciprocal rank of embeddings among sensors inside a neighborhood in space.

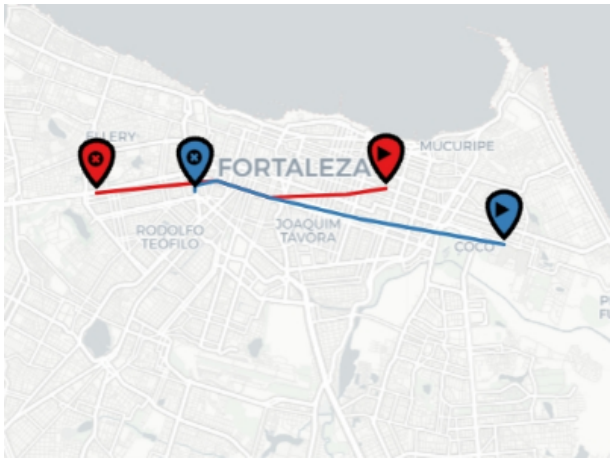
source, the marker with a triangle, and the target, the other marker. We divide the figure into three cases: when trajectories present high, medium, and low spatial similarity. Our goal is to show that the trajectory embeddings can capture the spatial likeness for some cases but not much for others. One further direction to improve the embedding quality is training with more data or investigating other language models.

We evaluate the Reciprocal Rank and the Mean Reciprocal Rank (Definitions 5 and 6) between raw EST trajectories and their embeddings using the Dynamic Time Warping (DTW) and the Edit (ED) distances between raw trajectories and also the cosine distance between embedding vectors. DTW is one of the most popular trajectory distance measures. It searches for all possible alignment points between two trajectories to find the one with minimal distance. For measuring the distance between sensors, we have used the Euclidean distance. ED quantifies the dissimilarity between two sequences of strings by counting the number of operations to transform one string into another. For more details about trajectory similarity functions, we refer to the survey (SU *et al.*, 2020).

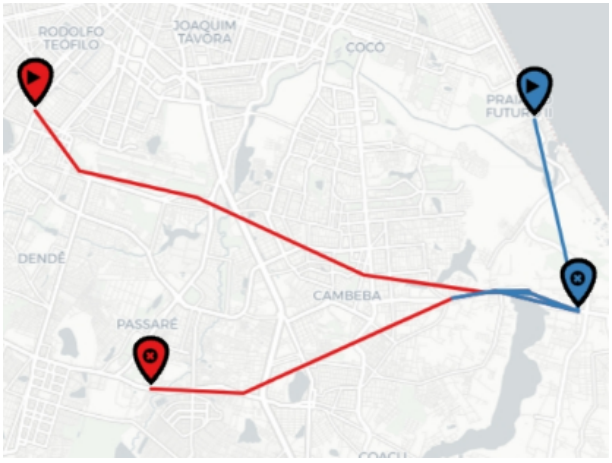
The experiment considers the cosine distance between trajectory embeddings as the query distance. We perform two experiments: compute the MRR using DTW as a reference distance and another one using ED as a reference distance. By using, DTW as the reference distance, MRR obtained the best results, with a value of 0.44. In other words, the rank of embedding for the most similar trajectory according to DTW is between 1 and 3, on average. The MRR using ED as the reference distance was 0.27. We believe that being on the top from



(a) High spatial similarity.



(b) Medium spatial similarity.



(c) Low spatial similarity.

Figure 26 – Example of most similar trajectories according to cosine distance between their embedding vectors.

one to three on average is a good result as we are learning a new embedding space to represent trajectories, which is a complex task due to the nature of these trajectory data.

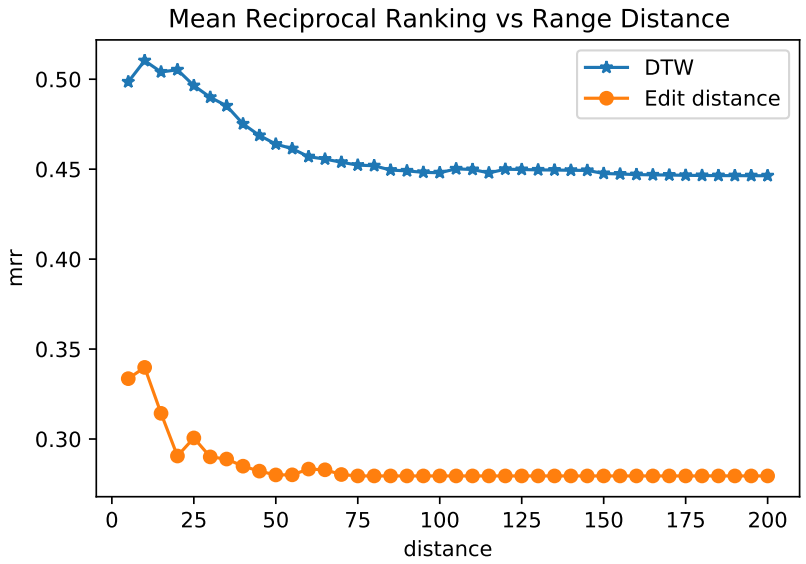


Figure 27 – Comparison of the mean reciprocal rank of the embeddings among trajectories pairs under a maximal distance.

Another experiment we performed was filtering out the set of trajectories objects O to only those with the reference distance under a maximal value. Similar to the analysis of sensor embeddings, we evaluate how the MRR performs on a subset of trajectories when the neighborhood increases. Figure 27 shows the results. For the most similar trajectories pairs (DTW distance of approximately 5), the MRR is close to 5. For trajectories on the neighborhood filtered by DTW equal to 85, the MRR has reached the minimum value (0.44) and maintains almost constant when the neighborhood increases. This result implies that BERT MLM embeddings could capture spatial similarity and connectivity when the sensors of similar trajectories do not exactly match. The highest value of MRR using ED was 0.33, at the point where ED is at a maximum of 10. This result implies that BERT embedding can represent well the sequence of discrete location labels in a trajectory (measured by edit distance). With these experiments, we have achieved our goal of using an NLP model and evaluating its quality to represent the trajectories and capture their spatial similarity.

5.3 Discussion

In this chapter, we investigated the use of NLP models to generate the embedding space of features for external sensors trajectories. We analyze the quality of the embedding trajectories by using extrinsic and intrinsic strategies. The extrinsic approach is concerned with evaluating a trajectory prediction architecture using different embedding vectors. In the intrinsic evaluation, we analyzed the reciprocal rank of embedding vectors by using well-known distance measures for locations (sensors, in our case) and trajectories. The experimental evaluation has shown the applicability of BERT MLM to extract embedding vectors to represent trajectories. BERT MLM has demonstrated the best result for location prediction tasks, mainly under fine-tuning the embedding layer. The reciprocal rank analysis has shown that embeddings can represent spatial similarity for locations in a restricted region surrounding the area. However, when the neighborhood expands, the embedding distance among sensors does not maintain the correlation with spatial distance (see Figure 25). Additionally, when the query range of trajectories increases, the correlation between trajectory similarity and the embedding distance does not decrease considerably (see Figure 27). In future work, we propose to investigate distinct approaches to compare and evaluate trajectories embeddings under different tasks and metrics.

6 CONCLUSION AND FUTURE DIRECTIONS

In this thesis, we presented and evaluated a schema based on recurrent neural networks that includes data imputation and clustering for location prediction from external sensors trajectories. We initially performed an analysis of the trajectory data to understand the behavior of trajectories collected by external sensors, which evidenced the sparsity and missing data on external sensor trajectories.

Based on this data analysis, we proposed approaches to deal with sparsity and incompleteness coming from external sensor trajectory data. We proposed single and multi-task learning models for the EST prediction problem, named Basic with Clustering, Next Imputation, Next Imputation with Clustering, Full Imputation, Full Imputation with Clustering, Multi-Task Basic, Multi-Task Basic with Clustering, Multi-Task with Next Imputation, Multi-Task with Next Imputation and Clustering, Multi-Task with Full Imputation, Multi-Task with Full Imputation and Clustering. We compare these methods with a simplified version of the state-of-art machine learning approach that fits our problem. The experimental evaluation has shown that our schema using multi-task learning and clustering could improve the accuracy and reduces the distance between the predicted location and the expected one.

We also study how the representation of trajectories can impact the performance of location predictors. More specifically, we investigate if the representation learning models from the state-of-art in NLP can generate the embedding space of features for on-road external sensors and EST. We evaluate the embedding representation intrinsically and extrinsically. The intrinsic evaluation access the quality of representation independent of specific tasks, directly evaluating the relationships between trajectories and sensors. The extrinsic evaluation of trajectory vectors has evaluated the embedding vectors using the external sensor trajectory prediction task.

In future works, we want to improve the quality of prediction approaches using different representation learning strategies. We want to investigate representation learning for trajectories enriched with external and semantic information (e.g., the number of crimes in the surrounding area of the sensor, type of road, day of the week, etc.). More specifically, we plan to study how to combine attributes of different domains in a unique embedding vector. Another future direction is to propose solutions for dealing with class imbalance and concept drift problems. Furthermore, we will study how to propose local and global models and integrate other temporal features. We can also investigate how to learn representation for different groups of users with different behavior.

REFERENCES

AKBIK, A.; BERGMANN, T.; BLYTHE, D.; RASUL, K.; SCHWETER, S.; VOLLGRAF, R. Flair: An easy-to-use framework for state-of-the-art nlp. In: **Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)**. [*S. l.*: *s. n.*], 2019. p. 54–59.

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**. [*S. l.*: *s. n.*], 2017.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 35, n. 8, p. 1798–1828, 2013.

BRANCO, P.; TORGO, L.; RIBEIRO, R. P. A survey of predictive modeling on imbalanced domains. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 49, n. 2, p. 1–50, 2016.

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**. New York, NY, USA: ACM, 2017. (PredictGIS'17), p. 2:1–2:4. ISBN 978-1-4503-5501-8.

CAO, H.; XU, F.; SANKARANARAYANAN, J.; LI, Y.; SAMET, H. Habit2vec: Trajectory semantic embedding for living pattern recognition in population. **IEEE Transactions on Mobile Computing**, IEEE, v. 19, n. 5, p. 1096–1108, 2019.

CHOLLET, F. *et al.* **Keras**. 2015. https://keras.io.

CONNOR, J. T.; MARTIN, R. D.; ATLAS, L. E. Recurrent neural networks and robust time series prediction. **IEEE transactions on neural networks**, IEEE, v. 5, n. 2, p. 240–254, 1994.

CRASWELL, N. Mean reciprocal rank. In: LIU, L.; ÖZSU, M. T. (Ed.). **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 1703–1703. ISBN 978-0-387-39940-9.

CRIVELLARI, A.; RESCH, B.; SHI, Y. Tracebert—a feasibility study on reconstructing spatial–temporal gaps from incomplete motion trajectories via bert training process on discrete location sequences. **Sensors**, MDPI, v. 22, n. 4, p. 1682, 2022.

CRUZ, L. A.; Zeitouni, K.; DE Macedo, J. A. F.; RAMALHO Brilhante, I. Trajsense: Trajectory prediction from sparse and missing external sensor data. In: **2019 20th IEEE International Conference on Mobile Data Management (MDM)**. [*S. l.*: *s. n.*], 2019. p. 365–366.

CRUZ, L. A.; ZEITOUNI, K.; MACEDO, J. A. F. de. Trajectory prediction from a mass of sparse and missing external sensor data. In: IEEE. **2019 20th IEEE International Conference on Mobile Data Management (MDM)**. [*S. l.*], 2019. p. 310–319.

CRUZ, L. A.; ZEITOUNI, K.; SILVA, T. L. C. da; MACEDO, J. A. F. de; SILVA, J. S. d. Location prediction: a deep spatiotemporal learning from external sensors data. **Distributed and Parallel Databases**, Springer, v. 39, n. 1, p. 259–280, 2021.

CUI, Y.; JIA, M.; LIN, T.-Y.; SONG, Y.; BELONGIE, S. Class-balanced loss based on effective number of samples. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2019. p. 9268–9277.

DAI, A. M.; OLAH, C.; LE, Q. V. Document embedding with paragraph vectors. **arXiv preprint arXiv:1507.07998**, 2015.

DAMIANI, M. L.; ACQUAVIVA, A.; HACHEM, F.; ROSSINI, M. Learning behavioral representations of human mobility. In: **Proceedings of the 28th International Conference on Advances in Geographic Information Systems**. [*S. l.*: *s. n.*], 2020. p. 367–376.

DERAKHSHAN, B.; MAHDIRAJI, A. R.; RABL, T.; MARKL, V. Continuous deployment of machine learning pipelines. In: **EDBT**. [*S. l.*: *s. n.*], 2019. p. 397–408.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. *et al.* A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. [*S. l.*: *s. n.*], 1996. v. 96, n. 34, p. 226–231.

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**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018. (WWW '18), p. 1459–1468. ISBN 978-1-4503-5639-8.

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: SPRINGER. **Joint European conference on machine learning and knowledge discovery in databases**. [*S. l.*], 2016. p. 36–40.

FU, K.; JI, T.; ZHAO, L.; LU, C.-T. Titan: A spatiotemporal feature learning framework for traffic incident duration prediction. In: ACM. **Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems**. [*S. l.*], 2019. p. 329–338.

FU, T.-Y.; LEE, W.-C. Trembr: Exploring road networks for trajectory representation learning. **ACM Trans. Intell. Syst. Technol.**, Association for Computing Machinery, New York, NY, USA, v. 11, n. 1, feb 2020. ISSN 2157-6904.

HAIXIANG, G.; YIJING, L.; SHANG, J.; MINGYUN, G.; YUANYUE, H.; BING, G. Learning from class-imbalanced data: Review of methods and applications. **Expert Systems with Applications**, Elsevier, v. 73, p. 220–239, 2017.

HASAN, S.; UKKUSURI, S. V. Reconstructing activity location sequences from incomplete check-in data: a semi-markov continuous-time bayesian network model. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 19, n. 3, p. 687–698, 2017.

HE, H.; CHEN, S.; LI, K.; XU, X. Incremental learning from stream data. **IEEE Transactions on Neural Networks**, IEEE, v. 22, n. 12, p. 1901–1914, 2011.

HENDAWI, A. M.; ALI, M.; MOKBEL, M. F. Panda∗ : A generic and scalable framework for predictive spatio-temporal queries. **GeoInformatica**, Springer, v. 21, n. 2, p. 175–208, 2017.

HENDAWI, A. M.; BAO, J.; MOKBEL, M. F. iroad: a framework for scalable predictive query processing on road networks. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 6, n. 12, p. 1262–1265, 2013.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HU, H.; KANTARDZIC, M.; SETHI, T. S. No free lunch theorem for concept drift detection in streaming data classification: A review. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 10, n. 2, p. e1327, 2020.

JANSEN, L.; LUIJTEN HANS, B.; BAKKER, N. Vincent van gogh - the letters. In: ____. Amsterdam The Hague: Van Gogh Museum Huygens ING, 2009. p. 638. Disponível em: https://vangoghletters.org.

JI, Y.; WANG, L.; WU, W.; SHAO, H.; FENG, Y. A method for LSTM-based trajectory modeling and abnormal trajectory detection. **IEEE Access**, IEEE, v. 8, p. 104063–104073, 2020.

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**. New York, NY, USA: ACM, 2018. (SIGSPATIAL '18), p. 528–531. ISBN 978-1-4503-5889-7.

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**. [*S. l.*: *s. n.*], 2018. p. 528–531.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, abs/1412.6980, 2014. Disponível em: http://arxiv.org/abs/1412.6980.

KUMAR, S.; ZHANG, X.; LESKOVEC, J. Predicting dynamic embedding trajectory in temporal interaction networks. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [*S. l.*: *s. n.*], 2019. p. 1269–1278.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: PMLR. **International conference on machine learning**. [*S. l.*], 2014. p. 1188–1196.

LI, X.; ZHAO, K.; CONG, G.; JENSEN, C. S.; WEI, W. Deep representation learning for trajectory similarity computation. In: IEEE. **2018 IEEE 34th international conference on data engineering (ICDE)**. [*S. l.*], 2018. p. 617–628.

LI, Y.; FU, K.; WANG, Z.; SHAHABI, C.; YE, J.; LIU, Y. Multi-task representation learning for travel time estimation. In: ACM. **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [*S. l.*], 2018. p. 1695–1704.

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**. [*S. l.*]: AAAI Press, 2016. (AAAI'16), p. 194–200.

LOSING, V.; HAMMER, B.; WERSING, H. Incremental on-line learning: A review and comparison of state of the art algorithms. **Neurocomputing**, Elsevier, v. 275, p. 1261–1274, 2018.

LU, J.; LIU, A.; DONG, F.; GU, F.; GAMA, J.; ZHANG, G. Learning under concept drift: A review. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 31, n. 12, p. 2346–2363, 2018.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

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. (Ed.). **Advances in Neural Information Processing Systems 26**. [*S. l.*]: Curran Associates, Inc., 2013. p. 3111–3119.

MIKOLOV, T.; YIH, W.-t.; ZWEIG, G. Linguistic regularities in continuous space word representations. In: **Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies**. [*S. l.*: *s. n.*], 2013. p. 746–751.

NASERIAN, E.; WANG, X.; DAHAL, K.; WANG, Z.; WANG, Z. Personalized location prediction for group travellers from spatial–temporal trajectories. **Future Generation Computer Systems**, v. 83, p. 278–292, 2018. ISSN 0167-739X.

NETO, J. S. D. S.; SILVA, T. L. C. D.; CRUZ, L. A.; LIRA, V. M. de; MACÊDO, J. A. F. de; MAGALHÃES, R. P.; PERES, L. G. Predicting the next location for trajectories from stolen vehicles. In: IEEE. **2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)**. [*S. l.*], 2021. p. 452–456.

NGUYEN, D.; LUO, W.; NGUYEN, T. D.; VENKATESH, S.; PHUNG, D. Sqn2vec: Learning sequence representation via sequential patterns with a gap constraint. In: BERLINGERIO, M.; BONCHI, F.; GÄRTNER, T.; HURLEY, N.; IFRIM, G. (Ed.). **Machine Learning and Knowledge Discovery in Databases**. Cham: Springer International Publishing, 2019. p. 569–584.

OZA, N. C. Online bagging and boosting. In: IEEE. **2005 IEEE international conference on systems, man and cybernetics**. [*S. l.*], 2005. v. 3, p. 2340–2345.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [*S. l.*: *s. n.*], 2014. p. 1532–1543.

RASCHKA, S.; MIRJALILI, V. **Python machine learning**. [*S. l.*]: Packt Publishing Ltd, 2017.

ROCHA, C. L.; BRILHANTE, I. R.; LETTICH, F.; MACEDO, J. A. F. D.; RAFFAETà, A.; ANDRADE, R.; ORLANDO, S. Tpred: A spatio-temporal location predictor framework. In: **Proceedings of the 20th International Database Engineering; Applications Symposium**. New York, NY, USA: ACM, 2016. (IDEAS '16), p. 34–42. ISBN 978-1-4503-4118-9.

RONG, X. word2vec parameter learning explained. **arXiv preprint arXiv:1411.2738**, 2014.

SU, H.; LIU, S.; ZHENG, B.; ZHOU, X.; ZHENG, K. A survey of trajectory distance measures and performance evaluation. **The VLDB Journal**, Springer, v. 29, n. 1, p. 3–32, 2020.

SUN, Y.; TANG, K.; MINKU, L. L.; WANG, S.; YAO, X. Online ensemble learning of data streams with gradually evolved classes. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 28, n. 6, p. 1532–1545, 2016.

SUN, Y.; TANG, K.; ZHU, Z.; YAO, X. Concept drift adaptation by exploiting historical knowledge. **IEEE Transactions on Neural Networks and Learning Systems**, v. 29, n. 10, p. 4822–4832, 2018.

TRASARTI, R.; GUIDOTTI, R.; MONREALE, A.; GIANNOTTI, F. MyWay: Location prediction via mobility profiling. **Information Systems**, v. 64, p. 350–367, 2017. ISSN 03064379.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

WANG, S.; MINKU, L. L.; YAO, X. A learning framework for online class imbalance learning. In: IEEE. **2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)**. [*S. l.*], 2013. p. 36–45.

WANG, S.; MINKU, L. L.; YAO, X. Resampling-based ensemble methods for online class imbalance learning. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 27, n. 5, p. 1356–1368, 2014.

WANG, S.; MINKU, L. L.; YAO, X. A systematic study of online class imbalance learning with concept drift. **IEEE transactions on neural networks and learning systems**, IEEE, v. 29, n. 10, p. 4802–4821, 2018.

WU, F.; FU, K.; WANG, Y.; XIAO, Z.; FU, X. A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. **Algorithms**, v. 10, n. 2, 2017. ISSN 19994893.

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**. [*S. l.*]: AAAI Press, 2017. (IJCAI'17), p. 3083–3090. ISBN 978-0-9992411-0-3.

WU, H.; CHEN, Z.; SUN, W.; ZHENG, B.; WANG, W. Modeling trajectories with recurrent neural networks. In: **Proceedings of the 26th IJCAI**. [*S. l.*: *s. n.*], 2017. p. 3083–3090.

WU, H.; MAO, J.; SUN, W.; ZHENG, B.; ZHANG, H.; CHEN, Z.; WANG, W. Probabilistic robust route recovery with spatio-temporal dynamics. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [*S. l.*: *s. n.*], 2016. p. 1915–1924.

YAO, D.; ZHANG, C.; HUANG, J.; BI, J. Serm: A recurrent model for next location prediction in semantic trajectories. In: ACM. **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management**. [*S. l.*], 2017. p. 2411–2414.

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 CIKM**. [*S. l.*: *s. n.*], 2017. p. 2411–2414.

ZHANG, C.; HAN, J.; SHOU, L.; LU, J.; PORTA, T. L. Splitter: Mining fine-grained sequential patterns in semantic trajectories. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 7, n. 9, p. 769–780, 2014.

ZHANG, C.; ZHANG, K.; YUAN, Q.; ZHANG, L.; HANRATTY, T.; HAN, J. GMove : Group-Level Mobility Modeling Using Geo-Tagged Social Media. **Kdd**, p. 1305–1314, 2016. ISSN 2154-817X (Print).

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. **Knowledge and Information Systems**, Springer, v. 56, n. 3, p. 559–579, 2018.

APPENDIX A – TRAJSENSE DEMONSTRATION TOOL

We implemented a tool named TrajSense (CRUZ *et al.*, 2019a) which demonstrates the predictions and compares the methods proposed in (CRUZ *et al.*, 2019b). The scenarios of the demonstration are described below.

Query trajectories: A map with the sensors is shown; we query a set of trajectories that pass by a set of sensors by selecting an area and a time interval. Then, we specify the maximum number of trajectories to be retrieved. The system plots the result on the map, and the user selects one trajectory for the analysis. The graphical interface for this step is shown in Figure 28.

Figure 28 – Query trajectories scenario of demonstration tool.

Pre-process: Select a window of k sensors (highlighted on the table) on the trajectory chosen in the last step by choosing the latest sensor on the window. These sensors will be used as input for data imputation. On the next screen, we show how the pre-processing completes the transitions according to a specific approach. The graphical interface for this step is shown in Figure 29.

Plot and compare predictions: The selected sensors are submitted to the models. We evaluate and compare the strategies. The system shows the window of k sensors (yellow), the predicted sensor (orange), and the expected one (green) as well as the error in terms of road distance. In case of some sensor is imputed, this sensor is plotted too. For the clustering-based

Figure 29 – Pre-process scenario of demonstration tool.

approaches, the system shows the maximum distance in the cluster. The graphical interface for this step is shown in Figure 30.

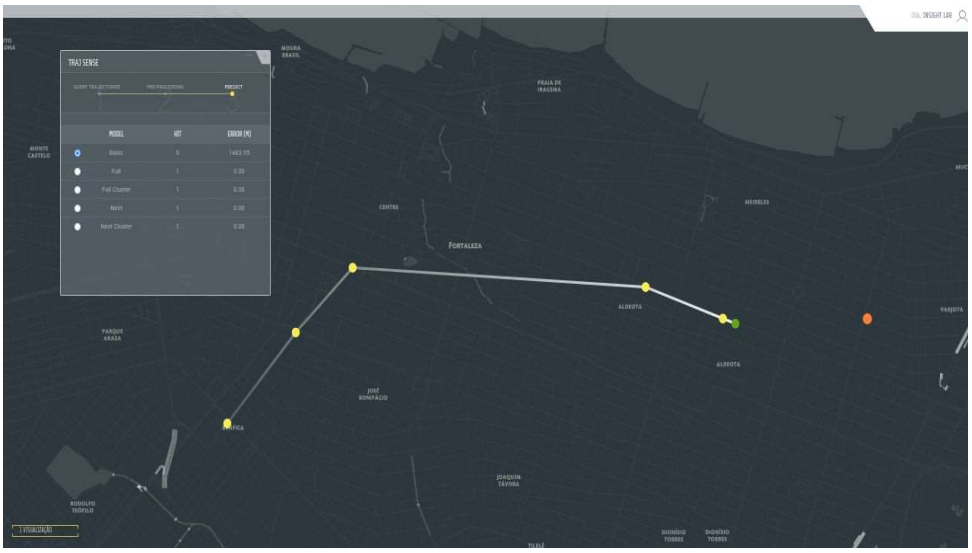


Figure 30 – Prediction scenario of demonstration tool.

APPENDIX B – EST PREDICTION UNDER CONCEPT DRIFT

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. Also, the distribution of vehicles is not spatial uniform, and sensors are not equally represented in data. Motivated by this, in this chapter, concept drift and imbalanced class problems are discussed as two relevant aspects that could be tacked into account on location prediction from EST. We also present complementary experiments to evaluate some basic solutions for these problems.

Concept Drift

In the data stream applications, concept drift means a change in the underlying distribution of that data over time that could impact the performance of machine learning tasks. When concept drift happens, past data could be not relevant to the new data, which implies poor predictions (LU *et al.*, 2018). Concept drift requires the learning model to be adaptive to changes. Classical machine learning approaches do not integrate new information into the existing trained models. Instead of that, they reconstruct the model from scratch which is very time-consuming.

Given a supervised classification task, where (x_t, y) is a sample arriving at time t, x_t is the set of descriptors and y is the label; the concept drift may be classified into three types: change in prior probability $P(y)$, changes in the class conditional probability function $P(x \mid y)$ and changes in the posterior probability $P(y \mid x)$. Concept drift of the last type, changes in $P(y \mid x)$, affects the decision boundary of the classifier, which implies the need to adapt the learning model to the new data distribution (WANG *et al.*, 2018). Figure 31 illustrates a change in the posterior probability $P(y \mid x)$ and how these changes modify the decision boundary of a linear classifier.

The research on concept drift learning adds to the conventional methods of machine learning new components (LU *et al.*, 2018): drift detection, drift understanding, and drift adaptation. Drift detection research aims to develop techniques to identify whether or not drift happens; drift understanding aims to identify when drift occurs, its degree, and the region of drift; drift adaptation research develops strategies to update learning models according to the drift.

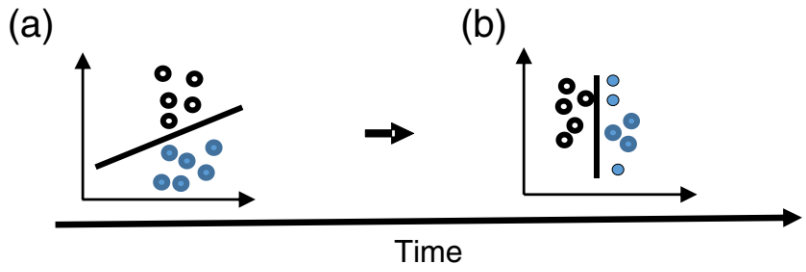


Figure 31 – Exemplifying concept drift of a linear classifier: classification bound before (a) and after (b) concept drift. Reprinted from (HU *et al.*, 2020).

The survey (HU *et al.*, 2020) presents the main three components of a general stream classification framework (Figure 32). Firstly, Concept Drift Detection and Concept Drift Classification Units are trained using previously collected data. Then, when new data samples arrive, the Concept Drift Detection Unit receives the data. If concept drift is detected, the Classifier Unit applies an adaptation strategy using the most recent data. If there is no concept drift detected, the samples are sent to the Classification Unit.

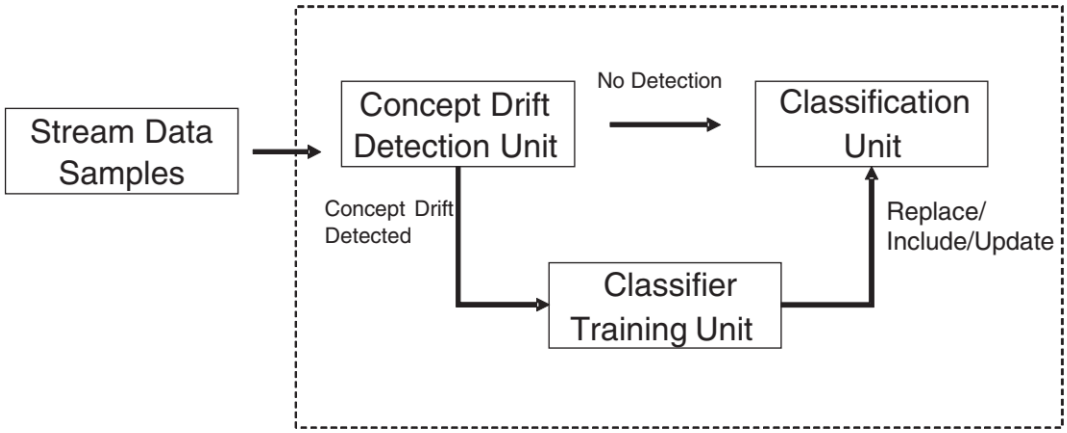


Figure 32 – General workflow data for stream classification under concept drift. Reprinted from (HU *et al.*, 2020).

In terms of drift adaptation, *incremental* and *online* approaches continuously incorporate new information into their model on the same time that aims for minimal processing time and space, these kinds of solution fits well to update learning models under concept drift (LOSING *et al.*, 2018). According to He *et al.* (2011), there are two mainly types of *incremental* approaches. The first one applies a data accumulation strategy. When a chunk of data arrives, one simply trains a new model with the data accumulated data and discards the previously trained model. This strategy usually needs to store all historical data. The second one utilizes ensemble

methods. When a new data chuck arrives, a new model or a set of models are trained based on the new data. Then, these new models are integrated to reach the final precision. This approach does not need to access all previous data. Incremental approaches update the model triggered by certain events, which could be a drop in the quality of the model or such amount of time since the last retraining (DERAKHSHAN *et al.*, 2019).

Class Imbalance

Class imbalance happens when the classes are not equally represented, in other words, there is at least one category with a number of occurrences much less than the other categories. On learning tasks guided by global performance, class imbalanced data create a bias towards the majority class and rare minority examples may be treated as noise by some machine learning approaches.

Some basic strategies to address class imbalanced in classification task are *resamplig* and *cost-sensitive learning* (HAIXIANG *et al.*, 2017). *Resamplig* strategies have the goal to rebalance the sample data to reduce the effects of class imbalance. This can be done by *over-sampling* methods that generate new instances of minority classes, by *under-sampling* that reduce the number of instances of majority classes, and also by *hybrid* methods that combine over-sampling and under-sampling. *Cost-sensitive* strategies assign a cost to the misclassification of each class and consider a higher cost to the wrong classification of minority classes when compared to majority ones.

Class imbalance also requires to adapt evaluation metrics to access the performance of the learning model on imbalanced data. In such problems, standard evaluation criteria focus on the most frequent cases. Many metrics were proposed for imbalanced class classification problems. Branco *et al.* (2016) shows a discussion about those metrics. The authors highlight that most of the metrics were proposed for two-class problems and there is still a gap for those measures in multi-class imbalanced domains.

Hu *et al.* (2020) presents a survey of classification methods under concept drift. They divide the concept drift into major categories: (i) concept drift with novel classes and novel features; (ii) concept drift with constant features and classes. For the last one, they divide the sub-types according to the detection scope (single drift and drift sequence), criteria (speed, distribution, recurrence, and time interval), and type (abrupt, gradual, incremental, fixed space, non-fixed space, recurrent non-recurrent, periodic and regular). They also present a comparison

of some approaches to drift detection, including performance-based, data-based, and ensemble.

Derakhshan *et al.* (2019) propose a platform for continuous update of machine learning models using historical and incoming training data. Instead of periodical retraining, the platform performs regular updates based on samples of data. The platform processes the incoming training data and transforms the data into small chunks; it prepossesses the raw data in the chunks and transforms the raw data into features. The framework applies a sampling strategy to select chunks used to update the model, and proactive training that continuously updates the model using mini-batch stochastic gradient descent to update the model. A scheduler component is responsible for scheduling proactive training. The work (SUN *et al.*, 2018) proposes an approach to deal with non-stationary distributions. The ensemble method uses preserved historical models as initial models for searching/training new models when new data chunk arrives. The newly obtained models are combined to form the new ensemble. The approach employs a decision tree as the base learner.

In (WANG *et al.*, 2018), the authors discuss the challenges of learning with concept drift when the data stream is imbalanced. According to them, this challenge arises because one problem (concept drift or class unbalance) can affect the other. They present some methods to tackle both class imbalance and concept drift, most of them are chunk-based approaches when a batch of data is received at each time. To deal with class imbalance, OOB and UOB algorithms (WANG *et al.*, 2013) apply oversampling and under-sampling to Online Bagging (OZA, 2005). OOB and UOB dynamically estimate imbalance and decides the resampling rate in each time step. The resampling in these approaches is based on the parameter λ of Poisson distribution. The work of (WANG *et al.*, 2014) proposes an extension of OOB and UOB by improving the parameter setting strategy. (SUN *et al.*, 2016) presents CBCE, a class-based ensemble for class evolution. The drift tacked into account by CBCE is class evolution, including class appearing and disappearing. The approach maintains a base learner for each class that has already appeared. Each model implements a *one vs all* strategy to classify one class. To address the class imbalance issue, they propose an under-sampling strategy included in each class-based model.

Concept Drift and Class Imbalance on EST

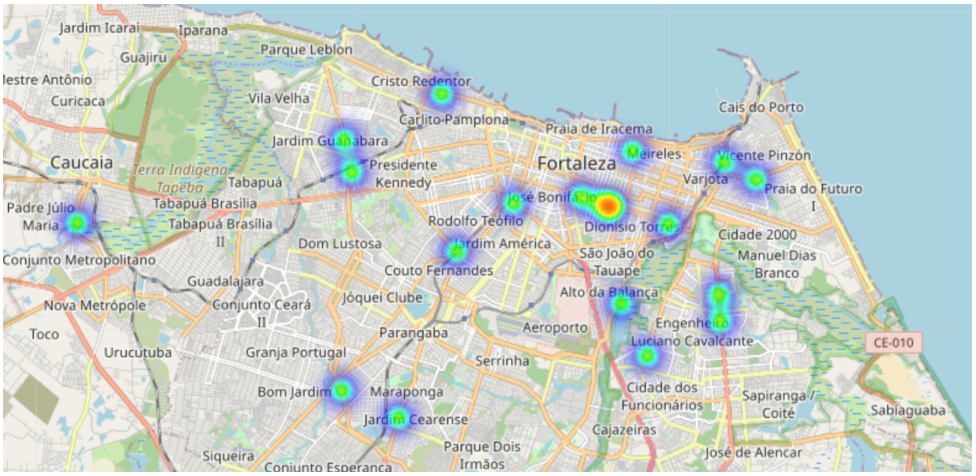
Class imbalance and concept drift are crucial questions in machine learning applications in the real world. The problem becomes challenging when they occur at the same time (WANG *et al.*, 2018). This section presents a study on the EST prediction that identifies both

problems, concept drift, and class imbalance. On EST data, concept drift can appear in many ways, for example, the spatial distribution of vehicles passages over the city may change on time, new regions may start to be visited while some regions may reduce the number of visitors; preferred paths may change depending on the day of the week and so on. Also, the number of observations in each sensor varies and the majority of sensors have much more observations than the minority ones, which characterizes class imbalance. For the purpose of the analysis, it was considered the same trajectory dataset described in Chapter 3 and the solution presented in Chapter 4.

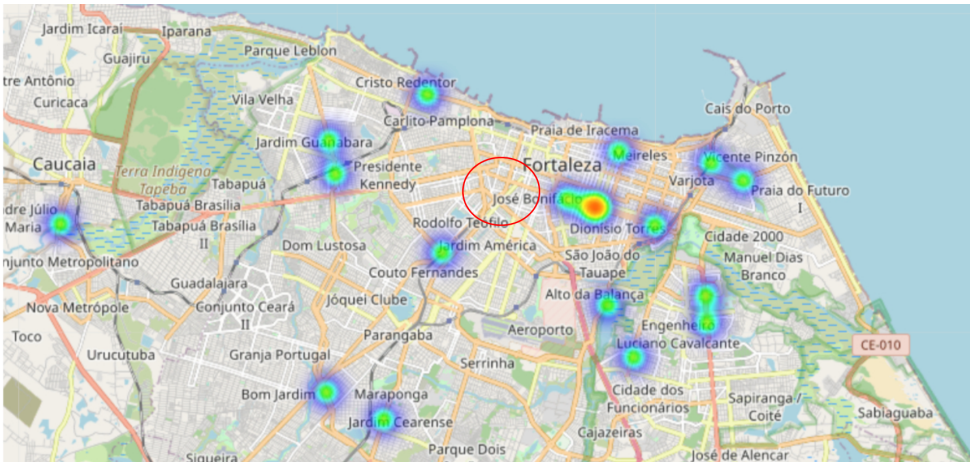
Figure 33 presents heat maps of vehicle passages by the sensors at days (a) 15, (b) 16 and (c) 20. The circle highlight the zones with some visual difference in sensors observations distribution among the days. We can note that some new zones appeared, while other zones disappeared, which shows how the spatial distribution of trajectories changes over time.

To visually analyze the class imbalance, as the sensors are the target of the EST prediction task, we selected the top-10 most frequent sensors on one day and visualize their frequency on the other days. Figure 34 presents this result, where each sensor is represented by a different color. The axis x is the day and the axis y is the support of occurrences or their frequency. We can observe that the most frequent target sensor appeared about 4000 times on day 15, while the third appeared less than 3000, and the $10th$ sensor in this rank appeared about 2000 times. These values indicate that class imbalance occurs in the data. We can also observe how the support of the sensors changes over time. Especially, on some days the support has been considerably reduced for all these sensors. For example, on day 20, all of them presented support of less than 1500, which also indicates concept drift. Following, we show in our experiments that day 20 presented the lower accuracy by the models, which indicates that the change in the data distribution has impacted the performance of the learning model.

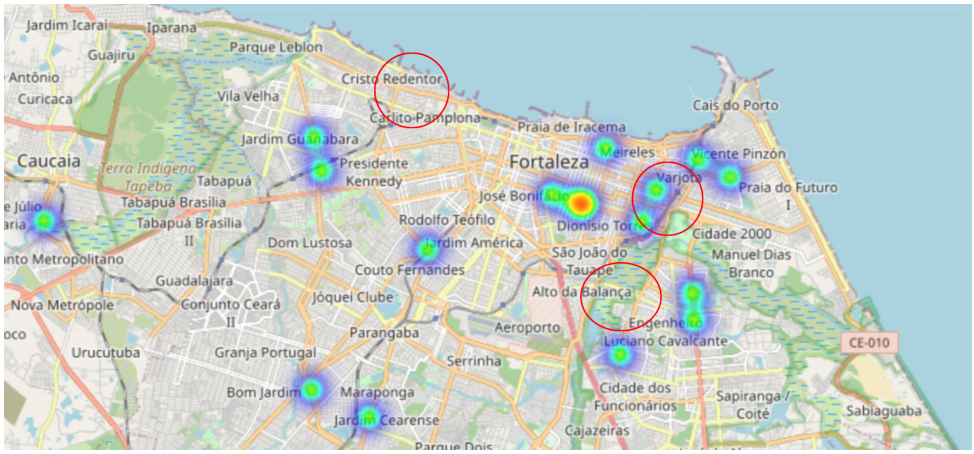
Figure 35 presents the difference in support between the most frequent (majority) and the less frequent (minority) sensors for each day. In this analysis, it was considered only the sensors with support greater than zero each day. We can observe how this difference varies over time. Initially, the majority and minority sensors have differences in support of about 4000, highlighting the class imbalance problem. This number decreases until day 20 and then, it turns to increase again.



(a) Heat map in July 15, 2017.



(b) Heat map in July 16, 2017.



(c) Heat map in July 20, 2017.

Figure 33 – Heat map of sensors passages reflects spatial Concept drift.

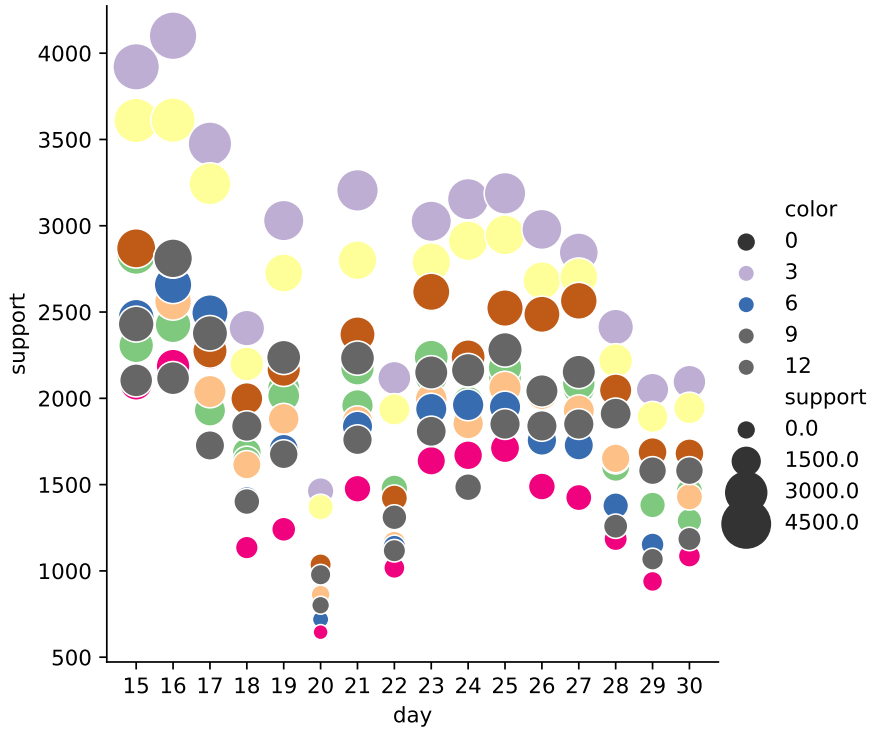


Figure 34 – Support of top-10 more frequent sensors (fixed at the 16th day) over time.

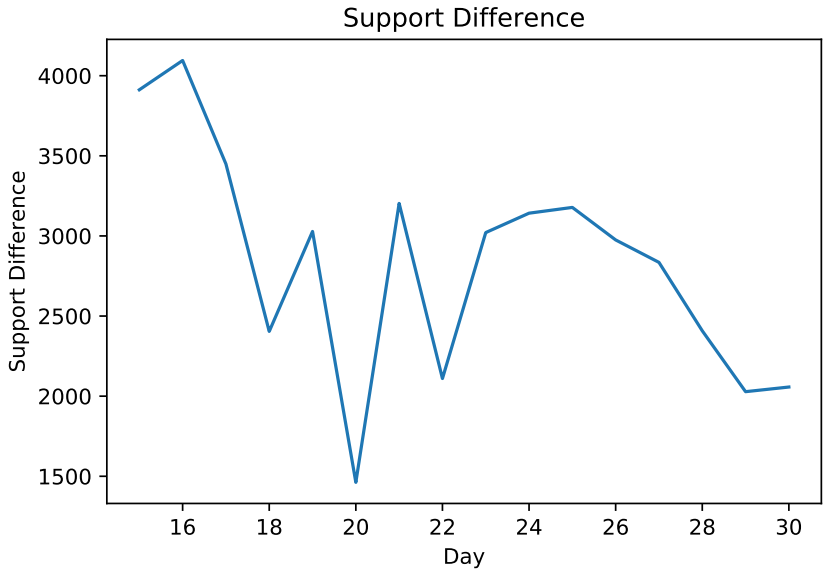


Figure 35 – Difference between support of the majority and minority sensors.

Evaluation of EST prediction framework under concept drift and class imbalance

In the following, we present an experimental evaluation for the models introduced in Chapter 4. In general, the objective of this investigation is to analyze the impact of some basic strategies to deal with concept drift and class imbalance.

The metrics considered were **accuracy** and **macro-averaged accuracy**, detailed

following.

Let us say the test set contains n instances of sub trajectories, each one having the target sensor $s \in S$. Each test case i has s_i as the true target sensor and $\overline{s_i}$ as the predicted sensor. $I(.)$ is a function that returns 1 if the argument is true, otherwise, it returns 0.

In multi-class problems, the **accuracy** is given by the formula

$$accuracy = \frac{\sum_{i=1}^{n} I(s_i = \overline{s_i})}{n} \tag{B.1}$$

However, **accuracy** is not appropriated for unbalanced domains (HU *et al.*, 2020). An extension of accuracy for unbalanced domains is the **macro-averaged accuracy (MAvA)** (BRANCO *et al.*, 2016). It is given by the average of **recall** of each class.

$$recall(s) = \frac{TP_s}{TP_s + FN_s} \tag{B.2}$$

Where TP is the number of true positives and FN is the number of false negatives of for the class s. Thus, the macro-averaged accuracy is given by Equation B.3.

$$MAvA = \frac{\sum_{s \in S} recall(s)}{\mid S \mid} \tag{B.3}$$

In that way, we evaluate the accuracy and macro-averaged accuracy to compare the different results of each measure.

Incremental update

The goal of this experiment is to evaluate an incremental strategy to update the model using the new data collected on the last day.

***Methodology*:** We evaluate the multitask models named MT-Basic and MT-FIC trained using the trajectories generated from the first fourteen days (September 1st to September $14th$). Then, for the data generated from September $15th$ to September $30th$, the trajectories of each day were used as a test set. To test the models, two strategies were performed: **MT-B (no update)** and **MT-FIC (no update)**, that simple applies data set from that day as test set; **MT-B (update)** and **MT-FIC (update)**, after evaluate the model with data of day t, before test with data in day $t+1$, the model was updated using the data of day t.

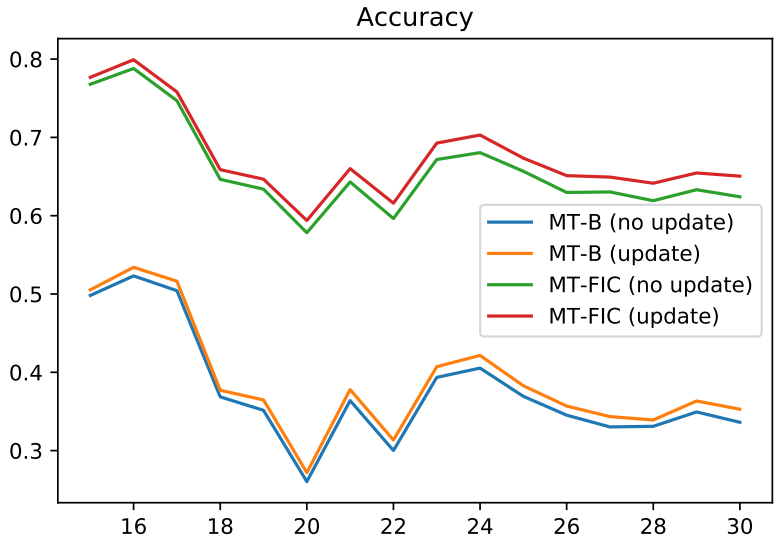


Figure 36 – Accuracy of models over days, with and without update strategy.

Figure 36 presents the accuracy of models. As in the experiments presented in Chapter 3, MT-FIC models outperformed the MT-Basic, in this case with and without incremental update strategy. Incrementally updating the models over the days has slightly improved the accuracy. The accuracy curve had peaks similar to the support presented in Figure 35. One can note that when the support of the top-10 most frequent sensors decreases (Figure 35 on day 20), the accuracy also decreases. Also, the best accuracy was found when the support of those sensors was the highest (Figure 35 on day 16), which indicates the distribution of data on that day is more similar to the distribution of training data. To better visualize the impact on the performance of updated and not updated models, we calculate the difference between the accuracy of **no update** and **update** strategies over the days. Figure 37 shows this result. MT-Basic with update has reached up to 1.7% of gain when compared with the model without an update. MT-FIC has shown an improvement of 2.5% with the update strategy.

Figure 38 presents the macro-averaged accuracy obtained by the models over the days. As expected, the macro-averaged accuracy values are lower than the accuracy values because it is calculated by considering the contribution of each class, including the minority classes. The MT-FIC model outperformed the MT-Basic again, with and without an incremental update. As has happened with accuracy, the curve of macro-averaged accuracy over days also follows the support difference, however, it is smoother when compared to support difference curve (35). This indicates the higher accuracy in some days could be the result of a similar distribution of those classes in the training set. Also, the macro-averaged accuracy lower than

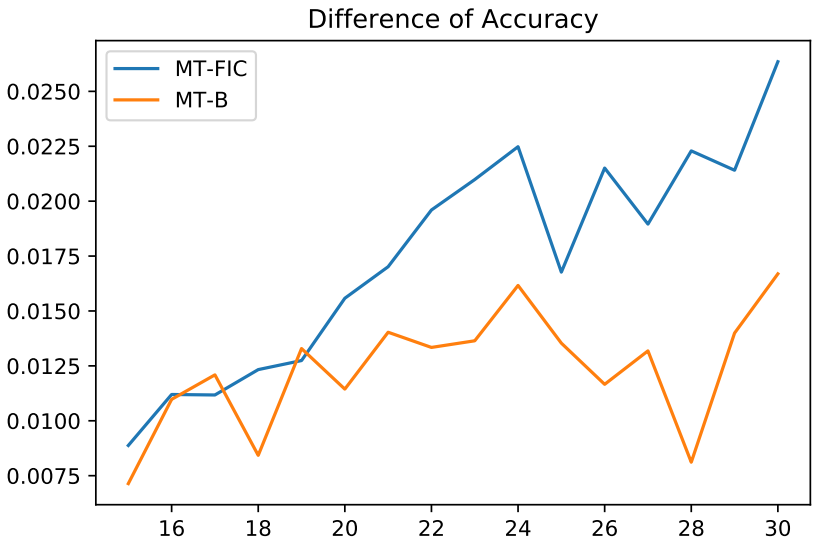


Figure 37 – Difference between the accuracy of models with and without updating.

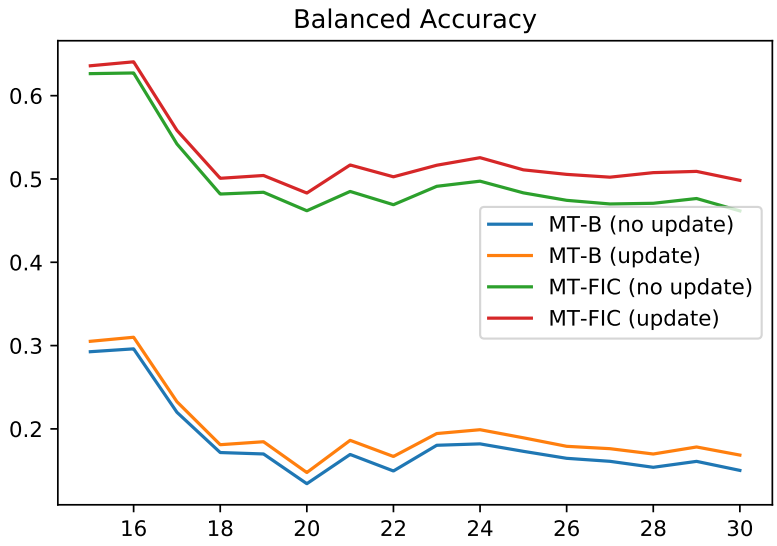


Figure 38 – Macro-averaged accuracy of models with and without updating.

accuracy is a consequence of the class imbalance.

Finally, we analyzed the gain after the incremental update strategy of both models in terms of macro-averaged accuracy. Figure 39 presents this result. We can observe the gain of MT-FIC was up to 3.6% and the gain of MT-Basic was up to 1.8%, which indicates the update strategy was more effective on MT-FIC when the number of classes is reduced.

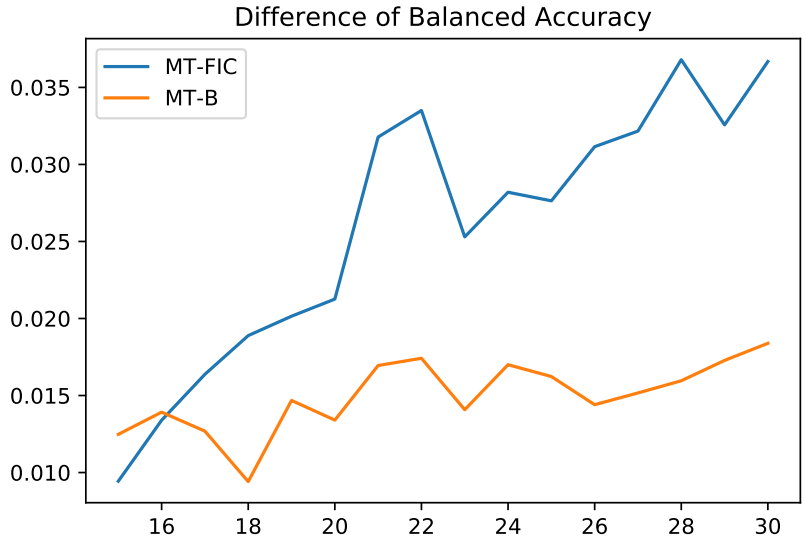


Figure 39 – Difference of macro-averaged accuracy with and without updating the model.

Resampling

Resampling is one of the strategies used to deal with the class imbalance problem. The goal of this experiment is to evaluate an incremental update approach using the new data collected on the last day together with a resampling strategy to deal with class imbalance.

Methodology: In this experiment, we evaluate only the best multitask model **MT-FIC** trained using the trajectories generated from the first fourteen days (September 1st to September 14th). Then, for the data generated from September 15th to September 30th, the trajectories of each day were used as a test set. In this experiment, we had applied the incremental daily update considering the following strategies. **MT-FIC (no update)**: simple uses data set from that day as a test set. **MT-FIC (daily update)**: the model was trained with original data, data of each day was used as a test set and daily updates were applied on the model using the data collected a day before. **MT-FIC (daily update and oversampling)**: the model was trained with original data with random oversampling, data of each day is used as a test set, random oversampling is applied to data and daily updates were applied to the model using the data collected a day before, after oversampling.

Figure 40 presents the accuracy of the three strategies compared in this experiment. Concerning accuracy, we can observe the *daily update* strategy has reached better results in terms of accuracy than the one without the update, which indicates the model could adapt a little better to new data even with a simple incremental update strategy. The use of random oversampling had

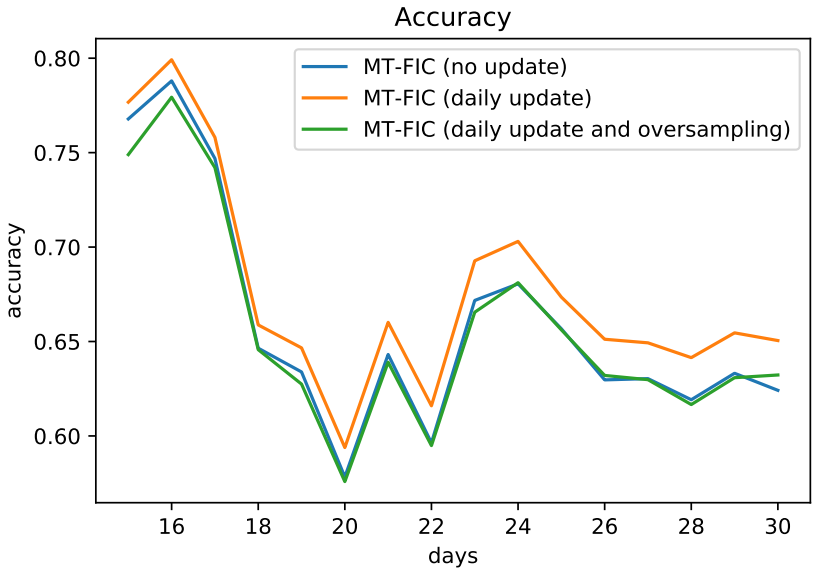


Figure 40 – Accuracy of MT-FIC model with no update, daily update and daily update with random sampling strategies.

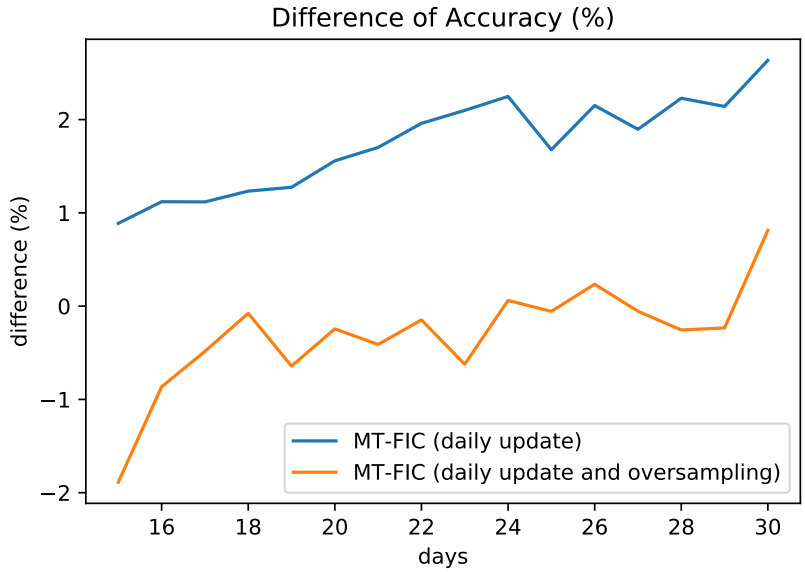


Figure 41 – Difference of accuracy of MT-FIC with daily update and MT-FIC with daily update and oversampling to basic version of MT-FIC without update.

not presented a significant difference to the no update strategy. Figure 41 presents the absolute gain of the accuracy of strategies with updates compared to no update. It is possible to observe the use of random oversampling turns the model worse to up 2% of cases.

Figure 42 presents the macro-averaged accuracy obtained by these strategies. The *daily update* strategy was the best one along all days. It is also possible to note a small benefit of the update with random oversampling when compared to *no update* strategy. However, the strategy with *daily update* only had performed better, which indicates that random resampling

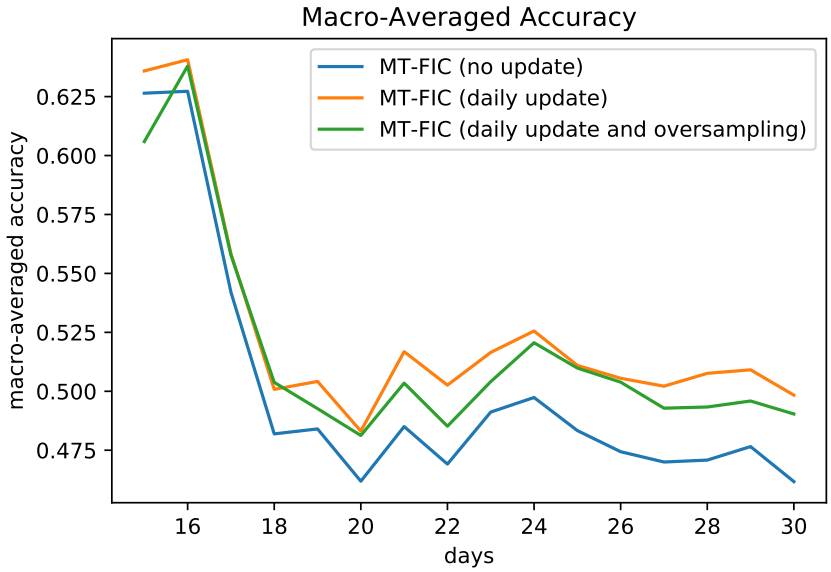


Figure 42 – Macro-averaged accuracy of MT-FIC model with no update, daily update and daily update with random sampling strategies.

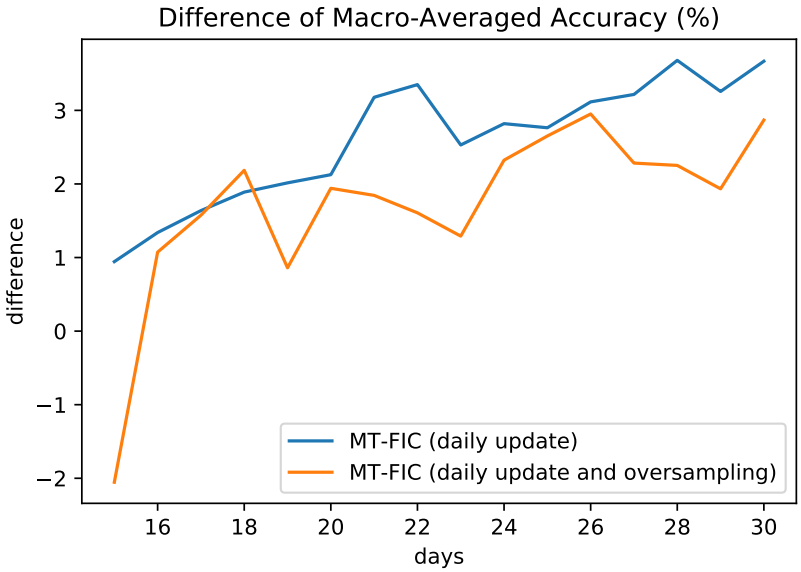


Figure 43 – Difference of macro-averaged accuracy of MT-FIC with daily update and MT-FIC with daily update and oversampling to basic version of MT-FIC without update.

generates some kind of noisy, instead of to improve its quality. Figure 43 shows the difference in terms of accuracy percentage strategies with an update to the basic one.

Balanced Loss Function

The goal of this experiment is to evaluate an incremental strategy to update the model using the new data collected on the last day together with a model-based strategy to deal with class imbalance.

Class-Balanced Loss: This experiment uses a scheme proposed by Cui *et al.* (2019) to re-weight samples of each class based on the effective number of samples. The *effective number* is define as following. by $E_n = (1-\beta^n)/(1-\beta)$, where $\beta = (N-1)/N$, N is the number of samples in each class and n is the number of samples. Based on the *effective number*, they proposed a general class-balanced loss function, that extends an generic loss function L. The Class-Balanced (*Class-Balanced* (KDE)) loss is defined as following.

$$CB(\mathbf{p}, y) = L(\mathbf{p}, y)\frac{1}{E_{n_y}} = L(\mathbf{p}, y)\frac{1-\beta}{1-\beta^{n_y}} \tag{B.4}$$

Methodology: In this experiment, we evaluate only the best multitask model **MT-FIC** using the daily update strategy trained using the trajectories generated from the first fourteen days (September 1st to September $14th$). Then, for the data generated from September $15th$ to September $30th$, the trajectories of each day were used as a test set. The **MT-FIC** was trained using different loss functions, the Class-Balanced Multi-class Cross-Entropy (*Class-Balanced Multi-Class Cross-Entropy* (CBMCE)) and Spatial-Temporal Class-Balanced Multi-class Cross-Entropy (*Spatial-Temporal Class-Balanced Multi-Class Cross-Entropy* (ST-CBMCE)). We named the strategies as **MT-FIC**: the original model was trained using **Cross-Entropy** as loss function; **MT-FIC (CBMCE)**: the model was trained using a balanced version of **Cross Entropy**, called here **Class Balanced Multi-Class Cross Entropy** only in the space output and the original Cross-Entropy in the time output; **MT-FIC (ST-CBMCE)**: the model was trained with **Class Balanced Multi-Class Cross-Entropy** as loss in both outputs (space and time).

Figure 44 presents the accuracy obtained by the three strategies. We can observe the ST-CBMCE version of MT-FIC was better when compared to the other versions and the major improvement was obtained in the peaks of drift (days 20, 21, and 22). However, MT-FIC (CBMCE) was not better than the original MT-FIC model. Figure 45 shows the difference in accuracy between each model obtained by changing in loss function and the original one. We can clearly show the importance to consider the unbalanced classes on training, including the temporal information used on multi-task models.

Figure 47 presents the values of macro-averaged accuracy obtained by the strategies. We can observe in this metric both models using a balanced version of loss function overperformed MT-FIC, and ST-CBMCE was the best one. Especially between days 18 and 24, the difference of the gain in macro-averaged accuracy obtained by MT-FIC (ST-CBMCE) (Figure 45

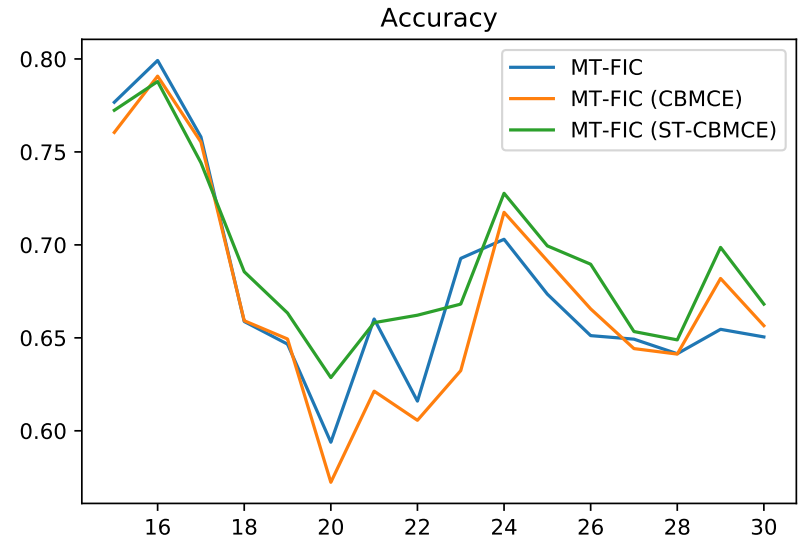


Figure 44 – Accuracy of version of MT-FIC with different loss functions.

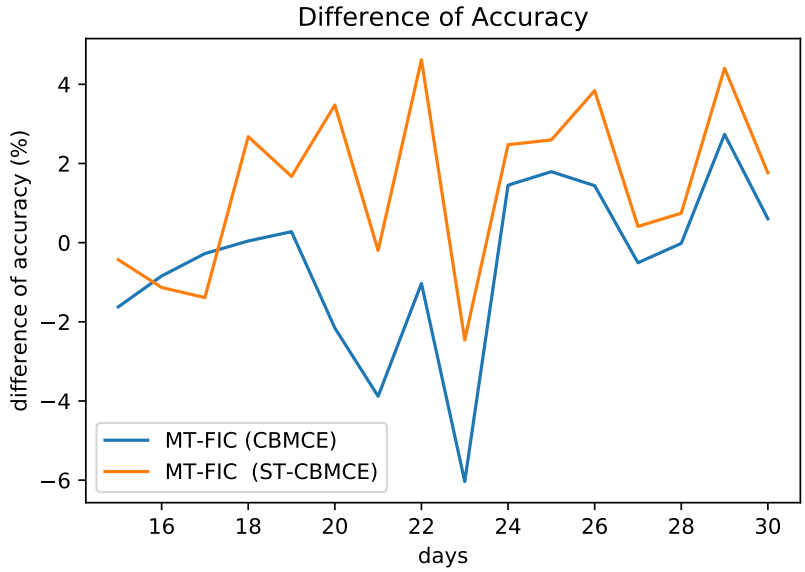


Figure 45 – Difference of accuracy of MT-FIC with balanced loss functions to the basic MT-FIC.

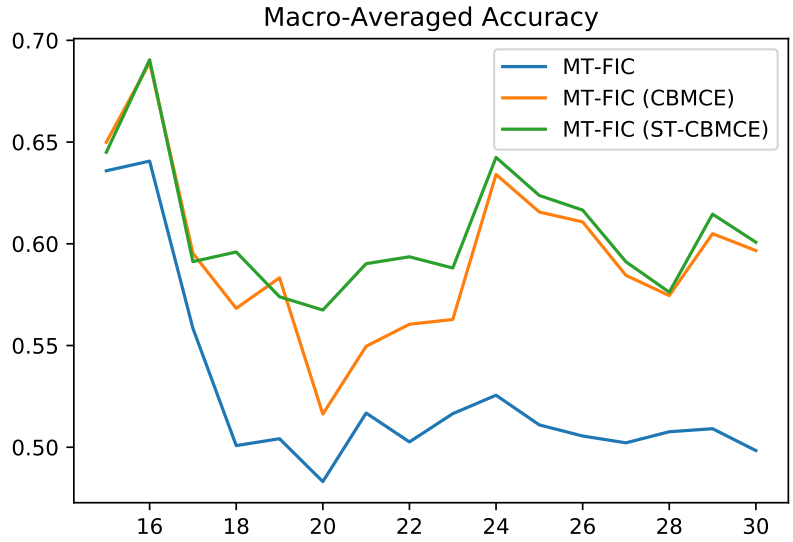


Figure 46 – Macro-averaged accuracy of version of MT-FIC with different loss functions.

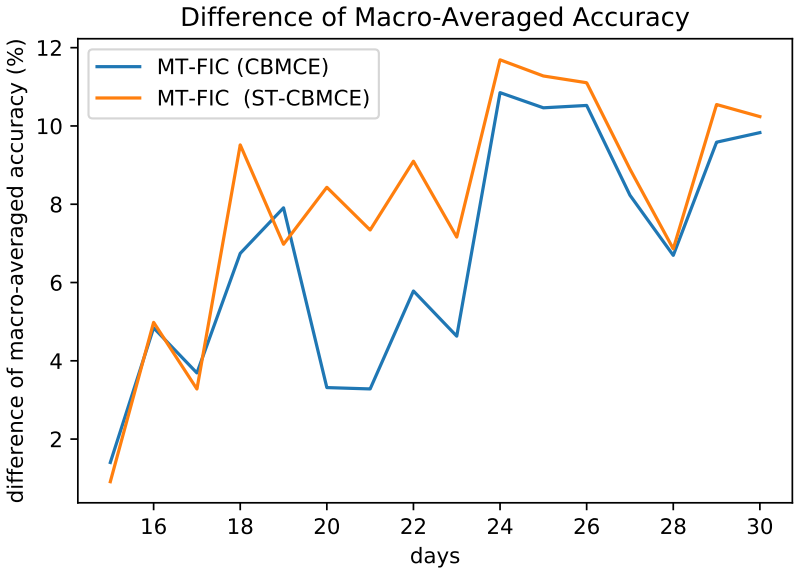


Figure 47 – Difference of macro-averaged accuracy of MT-FIC with balanced loss functions to the basic MT-FIC.

was more than the one obtained by MT-FIC (CBMCE). This indicates the model reaches better results when a balanced loss is used.

Discussion

This chapter discusses the problem of concept drift and class imbalance in EST prediction. We performed experiments including strategies of incremental learning update based on chunks of a day to deal with concept drift. Also, random oversampling and balanced loss

function were evaluated. The experiments show that, as expected, incremental updates had improved the quality of models. Random oversampling associated with the incremental update was not much more effective than the update only. The use of a balanced loss function had shown to be a good strategy.

APPENDIX C – EXTRINSIC EVALUATION WITH DIFFERENT WINDOW SIZES

In this section, we report the accuracy of the best prediction models considering the different window sizes. As explained in Section 5.2.1, the size of the window m is the number of the previous sensors observed in the trajectory, given as input for the model.

Table 14 – Results for the best LSTM model with different window sizes.

window size	**embedding size**	*ACC*@1	*ACC*@2	*ACC*@3
5	128	0.641	0.750	0.803
7	128	0.611	0.714	0.760
15	128	0.554	0.669	0.718
31	128	0.357	0.433	0.482

Table 15 – Results for the best LW2V model with different window size.

window size	**embedding size**	*ACC*@1	*ACC*@2	*ACC*@3
5	128	0.521	0.631	0.724
7	128	0.517	0.638	0.683
15	128	0.507	0.614	0.662
31	128	0.452	0.540	0.614

Table 16 – Results for the best LW2V-FT model with different window size.

window size	**embedding size**	*ACC*@1	*ACC*@2	*ACC*@3
5	128	0.533	0.669	0.721
7	64	0.526	0.650	0.695
15	32	0.514	0.624	0.664
31	128	0.460	0.550	0.614

Table 17 – Results for the best LBERT model with different window size.

window size	**embedding size**	*ACC*@1	*ACC*@2	*ACC*@3
5	128	0.668	0.797	0.851
7	128	0.587	0.748	0.814
15	128	0.405	0.579	0.654
31	128	0.480	0.659	0.783

Table 18 – Results for the best LBERT-FT model with different window size.

window size	**embedding size**	*ACC*@1	*ACC*@2	*ACC*@3
5	128	0.737	0.854	0.900
7	128	0.735	0.854	0.899
15	128	0.724	0.839	0.881
31	128	0.709	0.832	0.879