UNIVERSITÀ DEGLI STUDI DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE

ICT International Doctoral School
CYCLE XXX

MACHINE LEARNING METHODS FOR URBAN COMPUTING

GIANNI BARLACCHI

Advisors:

Bruno Lepri, *Fondazione Bruno Kessler*
Alessandro Moschitti, *University of Trento*

Industrial Advisor:

Michele Vescovi, *TIM-Telecom Italia (Joint Open Lab SKIL of Trento)*

ACADEMIC YEAR 2018/2019

ABSTRACT

World population is increasingly moving from rural areas to urban centers, making large cities densely populated. In urban areas, there is greater access to work, a wide variety of options for education and training, ease of transport and the abundance of attractive places within a few kilometers. Across huge cities, people tend to move more and have to do it faster than in the past. On the other hand, heavy traffic (e.g., traffic jams), overbuilding and changes in the urban lifestyle can cause several new problems such as noise, atmospheric pollution (i.e., smog) and severe traffic congestions. However, the rise of novel data sources and machine learning techniques can help to tackle such problems and improve the quality of life of citizens. Indeed, in a smart city environment, the huge amount of data generated daily can be captured by sensors, actuators, and mobile devices. It goes without saying that using such data opens the door to several applications, including forecasting of urban displacements, land use classification and event detection in an urban environment. Motived by these opportunities, Urban Computing (UC) leverages on heterogeneous data sources and applies machine learning techniques to tackle these big challenges that modern cities are facing. In this perspective, one of the core questions when designing UC systems is how to enable models to learn from different urban data sources and thus how to represent urban spaces. The mainstream approach is to represent input objects as feature vectors that encode several aspects of the urban environment such as presence of people, density of urban activities, and mobility flows. However, this tedious approach of manually feature engineering can be extremely complex, time consuming and domain-specific dependent. Additionally, it can become even more complex when aggregating multiple geographical data sources such as point-of-interests, administrative boundaries and mobility data. A valid alternative to feature-based methods is using kernels, which are non-linear functions that map input examples into some high dimensional space allowing for learning more powerful discriminative decision functions. Given a representation of the input object, kernels map it into some high-dimensional space where implicitly a large number of features are generated, allowing for learning robust discriminative functions. In this way the effort for the feature engineering process can be greatly reduced.

Kernel methods have been widely applied in Natural Language Processing on tasks such as question answering, semantic role labeling and even for solving linguistic games. Taking inspiration from this successful cases, in this thesis we adapt kernel learning for solving novel tasks in UC. First, we focus on the problem of aggregating multiple urban data sources to provide datasets that fuse knowledge from a wide variety of data sources. Next, we focus on the problem of designing input structure that are representative for urban space. In particular, we propose to model urban areas with tree structures that are fed to tree kernel functions for automatically generate expressive features. We propose several urban space representations that demonstrated to be very effecting in solving novel urban computing task such as land use classification and next location prediction in human mobility. Then, by applying a mining algorithm we enabled the interpretation of urban zones, providing help in the difficult problem of understanding the high-level urban characteristics of a city. In fact, our mined substructures provide help in identifying the different urban nature of cities. Finally, we explore the application of machine learning models to novel urban data sources by solving solve innovative tasks such as predicting the future presence of influenza-like symptoms looking at the people's mobility behaviors.

PUBLICATIONS

This PhD was supported by a fellowship grant from TIM-Telecom Italia (at the Joint Open Lab SKIL of Trento).

The contents of this thesis are based on ideas and figures presented in the following publications:

[1] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of milan and the province of trentino. *Scientific data*, 2:150055, 2015.

[2] Gianni Barlacchi, Alberto Rossi, Bruno Lepri, and Alessandro Moschitti. Structural semantic models for automatic analysis of urban areas. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 279–291. Springer, 2017.

[3] Gianni Barlacchi, Bruno Lepri, and Alessandro Moschitti. Predicting land use of italian cities using structural semantic models. *CLiC-it 2017 11-12 December 2017, Rome*, page 7, 2017.

[4] Alberto Rossi, Gianni Barlacchi, Monica Bianchini, and Bruno Lepri. Modeling taxi drivers' behaviour for the next destination prediction. *arXiv preprint arXiv:1807.08173*, 2018.

[5] Mohammed N Ahmed, Gianni Barlacchi, Stefano Braghin, Francesco Calabrese, Michele Ferretti, Vincent Lonij, Rahul Nair, Rana Novack, Jurij Paraszczak, and Andeep S Toor. A multi-scale approach to data-driven mass migration analysis. In *SoGood@ ECML-PKDD*, 2016.

[6] Gianni Barlacchi, Christos Perentis, Abhinav Mehrotra, Mirco Musolesi, and Bruno Lepri. Are you getting sick? predicting influenza-like symptoms using human mobility behaviors. *EPJ Data Science*, 6(1):27, 2017.

Some of the topics I studied for this thesis have not been included in the presented work, but they are available in the following publications:

[7] Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. Learning to rank answer candidates for automatic resolution of crossword puzzles. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 39–48, 2014.

[8] Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. Sacry: Syntax-based automatic crossword puzzle resolution system. *ACL-IJCNLP 2015*, page 79, 2015.

[9] Aliaksei Severyn, Massimo Nicosia, Gianni Barlacchi, and Alessandro Moschitti. Distributional neural networks for automatic resolution of crossword puzzles. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 199–204, 2015.

[10] Massimo Nicosia, Gianni Barlacchi, and Alessandro Moschitti. Learning to rank aggregated answers for crossword puzzles. In *European Conference on Information Retrieval*, pages 556–561. Springer, 2015.

[11] Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. A retrieval model for automatic resolution of crossword puzzles in italian language. In *The First Italian Conference on Computational Linguistics CLiC-it*, page 33, 2014.

[12] Olga Uryupina, Barbara Plank, Gianni Barlacchi, Francisco J Valverde-Albacete, Manos Tsagkias, Antonio Uva, and Alessandro Moschitti. Limosine pipeline: Multilingual uima-based nlp platform. *Proceedings of ACL-2016 System Demonstrations*, pages 157–162, 2016.

[13] Andrei Catalin Coman, Giacomo Zara, Yaroslav Nechaev, Gianni Barlacchi, and Alessandro Moschitti. Exploiting deep neural networks for tweet-based emoji prediction. In *International Workshop on Semantic Evaluation*, volume 4, page 1, 2018.

[14] Michele Ferretti, Gianni Barlacchi, Luca Pappalardo, Lorenzo Lucchini, and Bruno Lepri. Weak nodes detection in urban transport systems: Planning for resilience in singapore. *arXiv preprint arXiv:1809.07839*, 2018.

[15] Gianni Barlacchi, Azad Abad, Emanuele Rossinelli, and Alessandro Moschitti. Appetitoso: A search engine for restaurant retrieval based on dishes. *CLiC it*, page 46, 2016.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 URBAN COMPUTING

Latest demographic projections estimate an increasingly urban world population. As a result of this trend, by 2030 9% of the world population will be concentrated in just 41 *mega-cities*, each one with more than 10M inhabitants[37]. New and important challenges will then affect our society, from policy makers, urban planners and other experts to lay citizens. While cities are in fact praised as crucibles for innovation, due to their capacity to foster flows of ideas, tolerance and economic growth, they might also produce negative externalities such as environmental pollution, criminality and social inequalities [43]. Tackling these problems seemed impossible till some years ago, mostly due to lack of data and computability resources. However, this trend has started to change thanks to the massive adoption of sensing-technologies and large-scale computing infrastructures that enabled companies and researchers to explore novel methodologies to solve these challenges. Indeed, the growing amount of urban data generated, daily and the need for building more intelligent cities opened the door to new interesting questions. One characteristic of such problems is the need of aggregating multiple data sources. For instance, air quality prediction in urban spaces requires at least the aggregation of three data sources such as air quality, mobility, and weather. Another example is given by the applications integrated with Google Maps, the mapping service launched by Google in 2005. Among the others, it offers real-time traffic situation but also a very accurate route planning service with the estimation of arrival times and the prediction of traffic conditions. This emerging research field is known as Urban Computing (UC) [38] and aims at collecting, integrating and analyzing heterogeneous and large-scale urban data sources such as sensors, vehicles, buildings, humans, to solve the major issues that modern and future cites have to face.

The whole Urban Computing process is organized in four steps:

1. Urban sensing: it concerns the data acquisition phase where people's daily life data are collected and stored through (i) traditional urban sen-

sors and measurements like weather stations and official surveys; (ii) passive crowd sensing with taxis, public transportation and wireless cellular networks; and (iii) participatory sensing with data shared by citizenship and generated from sensors embedded in their devices.

2. Urban Data Management Techniques: it includes all the aspects useful for storing and managing spatial data, i.e., indexing structures for spatiotemporal information and additional data to support efficient data analytics.

3. Learning from Heterogenous Urban Data Sources: it is the step that includes all the learning approaches, e.g. supervised learning, that can fuse the knowledge and learn to solve urban computing tasks from heterogeneous data sources.

4. Providing Urban Applications: it delivers the results of the process through the deployment of systems, e.g., web applications, or with the visualization of the outcome inferred by the machine learning models.

A large class of Urban Computing tasks is solved with supervised learning algorithms, which learn a function that maps an input to an output based on some labeled training data. Each instance in the training set is a pair composed by an input object, e.g., set of measurements of air quality, GPS trajectory, the density of people in a city, and the desired output value, e.g., Air Quality Index (AQI), next visited location, urban zone land use class. Thus, the goal in supervised learning is to infer a decision function that can map new input examples to their correct output labels. This process is called *training* and it aims at minimizing the error (loss) function computed on a labeled training set. Once the model is trained, the accuracy of the learned function is evaluated on how well it can generalize (predict) with previously unseen examples.

In general, when solving any UC task with a supervised learning approach, it is required to:

- decide the type of data sources to use, e.g., mobility, environment, location-based social network, and provide a spatial aggregation of them. Solving urban challenges involves a wide range of factors whose common characteristic is the spatial component. Thus, when deciding the data sources to use it is important to evaluate if all of them provide the same level of spatial granularity.

- define a model for the spatial representation that will be fed to a learning algorithm for inferring a decision function from the training set. The input representation is fundamental for learning an appropriate decision function. Typically, the input object is transformed into a feature vector, e.g., the measure of CO_2, number of people moving between two urban zones, which is a vector of numbers that encodes certain characteristics of the input object;
- determine the learning algorithm depending on the input representation and the nature of the output labels.
- train the model and test it on previously unseen examples to asses the model accuracy.

In Urban computing, given the usage of a wide variety of data sources, it is likely to have a large number of features that can lead to several difficulties in learning, i.e., the curse of dimensionality problem. Additionally, when considering location data it is important to take into consideration the spatial dimension of the data. While there are many aspects to consider when tackling supervised learning tasks in urban computing, e.g., choice of an efficient learning algorithm, parameters optimization, etc., a key one is on how to represent the input objects, which calls for an effective spatial aggregation of all the knowledge for each location. In fact, the representation of a simple raw location, i.e., latitude/longitude, should model multiple dimensions of a given geographical area by aggregating heterogeneous urban data sources.

A large effort in the design of Urban Computing systems is devoted to the process of manual feature engineering required by feature-based methods. However, this step is largely an empirical process as it requires a high level of domain-specific knowledge, experience, and intuition. Additionally, when dealing with complex data structures, e.g., trees and graphs, such process can become very complicated. A valid alternative to the manual feature engineering approach is using kernel. A kernel is a non-linear function that maps input vectors in a high dimensionality feature space where the inner product is more efficiently calculated. Thus, given their ability to automatically generate a very large number of features, kernel functions can be designed to alleviate the feature engineering problem and reduce the required manual effort.

A successful field where structural kernel have been effectively employed is Natural Language Processing (NLP), a sub-field of Artificial Intelligence (AI)

that aims at enabling machines to understand and process human languages. The input text can be modeled with syntactic structural representations enriched with additional semantic information. For instance, structural kernels have been used in NLP for question answering [24, 25], opinion mining [26] and eventually to solve linguistic games [9, 129, 130].

Starting from the aforementioned problems, in this thesis we focused on designing rich spatial representation of urban spaces. This problem calls for having heterogenous urban data sources that allow for the modeling of multiple dimensions of a given geographical area. Thus, we start from the exploration of methods that can create multi-source urban life datasets in which heterogenous geo-located data sources, e.g., social media, mobile phone, cellular network, governments and administrative, are aggregated to form a rich resource to describe cities. Next, we address the problem of designing input structures that combine the spatial dimension of a given geographical area together with its semantics, i.e., its urban functionality. We show that by feeding tree kernel functions with such structural representation we can automatically generate a large set of tree-fragment features that are very effective in solving urban computing tasks. The core idea is to inject semantic information about places directly into the structures, e.g., a tree, and let the structural kernel to generate rich feature spaces. We show that previously used feature-based models are often a poor choice that limits the accuracy achievable by supervised learning approaches. Finally, we explore the usage of heterogeneous urban data sources and structural representation of urban spaces to solve novel urban computing tasks related to healthcare, migration, and transportation. To motivate the proposed methodologies and leveraging on multiple data sources, in Section 5 we present some methods we designed to solve novel urban computing tasks, e.g., modeling the next location prediction problem with geographical information and neural networks.

In the following we provide an overview of the thesis structure highlighting its contributions.

1.2 CONTRIBUTIONS AND STRUCTURE OF THE THESIS

In Chapter 2, we introduce the reader to general concepts that we use in this thesis. We provide an overview over Support Vector Machines (SVMs), i.e., structural kernels, which are at the core of the supervised learning framework

in most of our contribution. By presenting a set of supervised learning problems in urban computing, we also give a formal definition of two supervised learning tasks that we tackle in this thesis: classification and regression.

CONTRIBUTION 1 (CHAPTER 3). *A multi-source dataset of urban life.*

In this chapter, we first provide an extensive description of several urban data sources. Then, we present our work on the richest open multi-source dataset ever released [1]. The dataset provides heterogeneous data sources, spatially aggregated, such as telecommunications, weather, news, social networks and electricity data from the city of Milan and the Province of Trentino.

CONTRIBUTION 2 (CHAPTER 4). *Novel structural semantics models for modelling urban spaces with hierarchical location-based social network data.*

The main drawback of feature-based models resides in the tedious feature engineering process. Additionally, extracting features from complex structures, e.g., trees, requires to define further procedures to encode the information contained in the substructures. In this chapter, we present the work done in [2] where different types of structural representations are presented and used to solve supervised learning tasks in Urban Computing. In particular, the proposed model features the following: (i) it encodes geo-located textual information taking into account its spatial component; (ii) uses tree kernels to automatically generate features from the input representation; (iii) combines both structural features, i.e., tree fragments, and additional feature vector representations into a single model; (iv) outperforms strong feature-based models when tested on urban computing tasks, e.g., land use classification.

CONTRIBUTION 3 (CHAPTER 4). *Novel kernel designed to weigh the tree fragments contribution relying on additional information attached to each node.*

We propose different kernel-based models exploiting new hierarchical representation for urban spaces. Tree kernels designed for urban computing tasks are presented as well as their combinations with standard vectors. In particular, we define a new kernel function, which encodes the frequency and, more in general, the weight of point-of-interests in a specific area.

CONTRIBUTION 4 (CHAPTER 4). *Reverse kernel engineering for the automatic characterization of urban zones.*

Structural kernels alleviate the manual feature engineering process thanks to their ability to represent rich features sets. However, the implicit nature of such space makes the model interpretation extremely difficult, e.g., finding which subtree is more useful than others. In this chapter, we propose a novel approach to explicitly extract structural features for urban computing tasks. In particular, by applying a mining algorithm [42] for structural kernels we create an explicit set of structural features that enable the possibility to learn linear models and thus, to use any other state-of-the-art machine learning algorithm based on such representation.

CONTRIBUTION 5 (CHAPTER 5). *Learning to solve urban computing tasks with heterogeneous urban data sources.*
In this chapter, we propose different methodologies, based on urban data, for the resolution of novel urban computing tasks. We present our works on:

- Classifying land use with structural semantic models, a novel methodology we proposed in [2, 3]. Our mined substructure can provide valuable evidence of the different urban nature of cities. Additionally, effective structural features can be used in other feature-based machine learning models, e.g., XGBoost, which often achieve the state-of-the-art in urban computing.
- Predicting the future presence of influenza-like symptoms by using mobility behaviors [6];
- A multi-scale approach on mass migration prediction [133], where by combing different data sources we propose a model to predict the arrivals and displacements of refugees in Europe;
- Predicting next taxi destination, which refers to our work presented in [4]. We present a Recurrent Neural Network (RNN) approach that models the taxi drivers' behaviour and encodes the semantics of visited locations by using geographical information from Location-Based Social Networks (LBSNs). In particular, RNNs are trained to predict the exact coordinates of the next destination, overcoming the problem of producing, in output, a limited set of locations, seen during the training phase.

2

LEARNING WITH URBAN DATA

In this chapter we introduce the main concepts that are used throughout this thesis. First, we define the general problem of supervised learning with respect to classification and regression problems. Then, we define the SVM learning algorithm and its kernelization. In particular, we formally define and discuss structural kernels, which is a recurrent topic in this thesis.

2.1 SUPERVISED LEARNING

Supervised learning is the machine learning task that maps input data into some output. Given a labeled training set $= \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ the goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ that maps an input $x \in \mathcal{X}$ into an output $y \in \mathcal{Y}$. The input x is a D-dimensional vector of numbers, also called *feature vector*. The output y could be both a categorical variable from some finite set $y \in \{1, ..., k\}$, in case of classification, or a real valued scalar in case of regression.

In the classification case, given an input instance $\mathbf{x}$ and a target class $\mathbf{y}$, the decision function is defined as in the following:

$$h(\mathbf{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \tag{1}$$

where $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a discriminative function that assigns a numerical score to the input-output variables pair. Such function is parametrized by a vector of weights $\mathbf{w}$, which is learned during the training. Typically, the function f is chosen to be linear in the weights vector $\mathbf{w}$:

$$f_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \left\langle \mathbf{w}, \psi(\mathbf{x}, \mathbf{y}) \right\rangle \tag{2}$$

where $\left\langle \cdot, \cdot \right\rangle$ is a dot product, and $\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ computes a joint feature representation of the input instance $(\mathbf{x}, \mathbf{y})$ into some feature space $\mathbb{R}^d$.

2.1.1 *Empirical Risk Minimization*

Given a training set $\mathcal{D}$ and the hypothesis space $\mathcal{H}$ of discriminative functions, the Empirical Risk Minimization search for the function that better fits the training data, i.e., minimizes the risk over the data distribution. Empirical Risk Minimization is a way for choosing the function f, which is determined by the models parameter $\mathbf{w}$ that have to be estimated. The Empirical Risk Minimization principle formalize this process by minimizing a task-dependent loss function L computed over the training instances:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} R(f) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(h(\mathbf{x}_i, \mathbf{y}_i)) \tag{3}$$

where L is a task-dependent loss function that measures how good the model is in terms of being able to output the expected outcome. To prevent overfitting, the risk R can be regularized to penalize complex models:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} R(f) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(h(\mathbf{x}_i, \mathbf{y}_i)) + \lambda \Omega(\mathbf{w}) \tag{4}$$

The term Ω is the regularization term. A common choice for the regularization term is the squared L_2-norm:

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2 \tag{5}$$

2.1.2 *Loss Function*

A loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ defines how much incorrect predictions have to be penalized. The cost is computed by evaluating the difference between the prediction from the model and the desired output value. A common choice used for training *maximum-margin* classifiers, i.e., SVM, is the hinge loss function, which represent the closest convex approximation of the standard zero-one loss used in classification. Its most general form is defined in the following:

$$l_{hinge}(t, y) = \max(0, \max_{y \neq t}(\Delta(y, t) + \langle \mathbf{w}, \psi(\mathbf{x}, \mathbf{y}) \rangle) - \langle \mathbf{w}, \psi(\mathbf{x}, \mathbf{y}) \rangle) \tag{6}$$

where $\Delta(\cdot, \cdot)$ measures the difference between the predicted output y and the correct output t. Because of this definition, it is worth noticing that the correct output t should have a higher score than any other predict y by at least $\Delta(y, t)$.

Given these concepts, to define a supervised learning problem it is required to define:

1. An input space where a feature map $\psi(\cdot,\cdot)$ is applied to map all the instances of train and test into some feature space $\mathbb{R}^d$. This is a feature engineering process and the design of such mapping is of paramount importance as the expressivity of the features directly affect the accuracy of the learned function;
2. An output space $\mathcal{Y}$ and its cardinality, which is task-dependent. For instance, $y \in \{-1, 1\}$ in case of binary classification;
3. An hypothesis space $\mathcal{H}$, a loss function L and the form of the regularization term $\Omega(\cdot)$ if present;
4. An inference process to assign output labels $\mathbf{y}$ to input $\mathbf{x}$ (Equation 1);
5. An optimization algorithm to learn the parameters that define the model by minimizing the empirical risk from Eq. 3 or Equation 4.

Some of these steps include choices that are task-dependent and can lead to different learning algorithms. In the following, we introduce supervised learning problems in urban computing focusing on two specific tasks that are directly related to this thesis: classification and regression.

2.2 SUPERVISED LEARNING PROBLEMS IN URBAN COMPUTING

In this section, we introduce some common urban computing tasks. Although the wide range of possible tasks, we discuss in the following those two tackled in this thesis: classification and regression.

2.2.1 *Classification*

In Urban Computing, some problems can be modeled as classification problems. The general goal is to map an input instance $\mathbf{x}$ into an output class $y \in \mathcal{Y}$, where $\mathcal{Y} = \{1, ..., K\}$ is the set of possible classes that can be assigned to input instances. The size K of the output domain is defined by the type of problem. In the case of binary classification $K = 2$ and $\mathcal{Y} = \{-1, 1\}$. while when $K > 2$, we have a multi-class classification problem with an output space $\mathcal{Y} = \{1, ..., K\}$. A variant of the classification task is for example where the function f outputs a probability distribution over the classes.

LAND USE CLASSIFICATION Cities are usually divided into areas depending on land use, which is a trait of urban life and a good proxies for the characterization of urban environments. In *land use classification* the goal is to infer the land use of a certain region given a feature representation of such region and a target label for that. This task can be modeled as a multi-class classification problem where the output domain is $\mathcal{Y} = \{1, ..., K\}$ for K land use classes. We consider this problem in Section 5.1. We encode geo-social data from Location-Based Social Networks (LBSNs) into different structural representations. Then, we used such representations in kernel machines, which can thus perform accurate classification exploiting hierarchical substructure of concepts as features (Chapter 4).

2.2.2 *Regression*

A wide range of problems in urban computing concerns the prediction of real valued numbers, and thus they can be seen as regression problems. In this type of task, to the algorithm is asked to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ that given an input instance, it outputs a numerical value.

NEXT LOCATION PREDICTION Predicting the next location is an extensively studied problem in human mobility, which finds several applications in real-world scenarios, from optimizing the efficiency of electronic dispatching systems to predicting and reducing the traffic jam. This task is normally modeled as a multi-class classification problem, where the goal is to select, among a set of already known locations, the next taxi destination. However, this setting assumes that the next visited location is among those ones already seen in the training set, which introduces a strong limit in the accuracy of the final prediction. Instead, regression is a more difficult but natural way to model this task. In this case, the algorithm is asked to learn to predict the exact coordinate values of the next visited location. In Section 5.4 we address this problem proposing to model the geography of visiting locations and use such information within a neural network whose goal is to output the coordinates of next visited location.

2.3 SUPPORT VECTOR MACHINES

SMVs are supervised learning models and the best known member of the family of kernel methods. The core idea of SMVs is to act as a linear classifier by finding an optimal hyperplane that divided the space and thus allow for categorizing new examples. However, if the linear separation between point is not possible in the input space, kernel functions enable SVMs to operate in a high-dimensional space, via the *kernel-trick*, where such separating operation might be possible.

2.3.1 *Primal Formulation*

Support Vector Machines (SVMs) support classification, regression and ranking. When used in the binary classification setting, the output space is $\mathcal{Y} = \{-1, 1\}$, and the joint feature map is simply $\psi(\mathbf{x}) = y\Phi(\mathbf{x})$ with $\Phi : \mathcal{X} \to \mathbb{R}^d$ defining an arbitrary feature representation of inputs. When considering the case of linear SVMs, where the input feature map is an identity function $\psi(\mathbf{x}) = \mathbf{x}$, the discriminant function can be defined as in the following:

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle \tag{7}$$

In this case, to assign the label we can simply consider the inference procedure $\hat{y} = sign(\langle \mathbf{w}, \mathbf{x} \rangle)$. Among the possible hyperplanes that can be used to separate the classes, a reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. In SVMs the hyperplane is selected accordingly with the maximum-margin principle, which selects the hyperplane that represents the largest separation, or confidence margin, between the points of the two classes. However, with meaningful positive rescaling we can obtain infinite equivalent formulation. Thus, we request the hyperplane to have $\langle \mathbf{w}, \mathbf{x} \rangle = 1$ for the closest points on one side [1], and $\langle \mathbf{w}, \mathbf{x} \rangle = -1$ for the closest points on the other side. The resulting hyperplanes have a confidence margin equal to 1 and they are called *canonical hyperplanes*. It results that each correctly classified point has $\langle \mathbf{w}, \mathbf{x} \rangle \geq 1$. The margin band size is set to $2/\|\mathbf{w}\|^2$, which is equal to two times the canonical hyperplane geometric margin. Selecting the best hyperplane is equivalent to maximize such margin, while correctly classifying all the training instances:

1 The bias term is omitted as it can be included in the feature vector $\mathbf{x}$

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i\langle \mathbf{w}, \mathbf{x}\rangle \geq 1 \end{aligned}$$

The points that satisfy at the equality the constraint are called *support vectors* and they are the only points that contribute to the decision function. All the other points are not used during the training and thus they can be discarded. Under this conditions we refers to the Hard Margin SVM as it requires that all the points respect the constraints, i.e., no points are allowed to be inside the margin band. This version is highly sensitive to outliers and thus not suitable to be used with noisy datasets. An alternative is given by the Soft Margin SVM that relax previous constraints, allowing training data to lie inside the margin band, or even on the wrong side of the hyperplane. In this case, in order to take in consideration those errors during the learning, non-negative slack variables are introduced. Thus, the objective tries to minimize the sum of the errors in addition to $\|\mathbf{w}\|^2$:

$$\begin{aligned} \underset{w,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{subject to} \quad & y_i\langle \mathbf{w}, \mathbf{x}\rangle \geq 1 - \xi_i, \xi_i \geq 0 \end{aligned}$$

For $0 < \xi_i \leq 1$ the constraint on the soft margin is satisfied, while $\xi_i > 1$ is a non-zero training error. The slack variable ξ_i represents the penalty assigned to the example x_i that does not satisfy the constraint. The term C is the regularization parameter and trades-off the margin size and data fitting. An high value for C leads to having few errors, but increase the possibility of overfitting. For $C \to \infty$, the cost to pay for each error is unaffordable, and it brings back to have the hard margin case. Contrary, when $C \to 0$ all the solutions are equals since we have $\mathbf{w} = 0$.

2.3.2 *Dual Formulation*

To solve the SVM optimization problem we can swap to the dual formulation, which allows for using kernels in the algorithm. Accordingly with the Representer Theorem [16], the $\mathbf{w}$ parameter vector can be obtained by a linear combination of the training instances:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{8}$$

By substituting $\mathbf{w}$ in the Equation 7 we obtain:

$$f_\alpha(\mathbf{x}) = \langle \sum_i \alpha_i y_i \mathbf{x}_i, \mathbf{x} \rangle = \sum_i \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle \tag{9}$$

In the following, we omit the derivation of the dual formulation from the primal, and we directly report the equation for optimization problem:

$$\begin{aligned} &\underset{\alpha \geq 0}{\text{maximize}} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \\ &\text{subject to} \quad 0 \leq \alpha_i \leq C \end{aligned}$$

From both primal and dual formulation, there are few observation we can make:

- We replaced $\mathbf{w}$, whose dimensionality is the same as the input space, with dual variables α_i that are independent of the dimensionality of the input space. Hence, instead of learning d parameters in the dual form there is a parameter α to learn for each of the N training examples;
- In $N << d$, it is more efficient to solve for α than for d;
- Examples with $\alpha_i \geq 0$ are called support vectors. They are the only examples used to classify new instances. Typically, a large portion of the α_i are zeros, making the number of support vectors smaller than the number of training instances.
- The decision function has a linear form. We can make it non-linear by transforming the input examples from their original feature space into another space via a non-linear feature map. Indeed, the dot product in the dual formulation can be substituted wit a kernel function.

2.3.3 *The kernel trick*

The SVMs operations are expressed in terms of dot products, making possible to swap to a non-linear form via feature map: $\phi(\mathbb{R}^d \rightarrow \mathbb{R}^{d'})$:

$$f_\alpha(\mathbf{x}) = \sum_i \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle \tag{10}$$

where the non-linear map $\phi(\cdot)$ does not have to be computed explicitly. Indeed, instead of mapping our data and then compute the inner product, we can do it in one single operation, leaving the mapping completely implicit.

This follows as a results of the Mercer's Theorem [17] that states that a symmetric function $K(\mathbf{x}_i, \mathbf{x}_j)$ can be expressed as an inner product $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, for some $\phi(\cdot)$ if and only if $K(\mathbf{x}_i, \mathbf{x}_j)$ is positive semidefinite, i.e. $\sum_i \sum_j K(\mathbf{x}_i, \mathbf{x}_j) c_i c_j \geq 0 \forall \mathbf{c}$.

Thus, all we need to know is how to compute the dot product, without requiring an explicit form of $\phi(\cdot)$. A function that (i) satisfies the Mercer's Theorem conditions; and (ii) given two input objects outputs an inner product in some feature space, it is called *kernel function*.

The resulting kernelized SVMs can be expressed as:

$$f_\alpha(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \tag{11}$$

Depending on the input data and the nature of the task, we can choose different kernel functions. In Table 1, we list some common kernel functions that can be applied on vectors of fixed dimensionality.

Polynomial of degree p	$K(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^p$
Sigmoid	$K(\mathbf{x}, \mathbf{y}) = tanh(a\langle \mathbf{x}, \mathbf{y} \rangle + b)$
Gaussian RBF	$K(\mathbf{x}, \mathbf{y}) = exp(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2)$

Table 1: Some examples of widely used kernel functions.

The kernel function provides an inner product between two input objects in some implicit feature space, and it can be applied to other types of input objects such as trees and graphs. In the next section we define kernels that are designed to operate with such structured objects.

2.4 STRUCTURAL KERNELS

Structural kernels have been successfully applied in machine learning in all the domain where computing the objects similarity is a function of the similarity of their subparts. This includes bioinformatics, datamining, urban computing and mostly, Natural Language Processing. The ability of automatically generates features, makes structural kernels very appealing for all those tasks where no expert knowledge is available.

In this section we introduce the Convolution Tree Kernels, a set of structural kernels used throughout this thesis: the Syntactic Tree Kernels (STKs) [88],

the Partial Tree Kernels (PTKs) [63] and the Smoothed Partial Tree Kernels (SPTKs) [27].

2.4.1 *Convolution Tree Kernels*

In the majority of machine learning approaches, data examples are transformed in feature vectors, which in turn are used in dot products for carrying out both learning and classification steps. Kernel Machines (KMs) allow for replacing the dot product with kernel functions, which compute the dot product directly from examples (i.e., they avoid the transformation of examples in vectors). As showed in Figure 1, Tree Kernels (TKs) are functions designed to compute the similarity of tree structures by counting the number of common subparts. What makes a tree kernel different from another is the way in which the subparts, i.e., the tree fragments composing the fragment space, are generated.

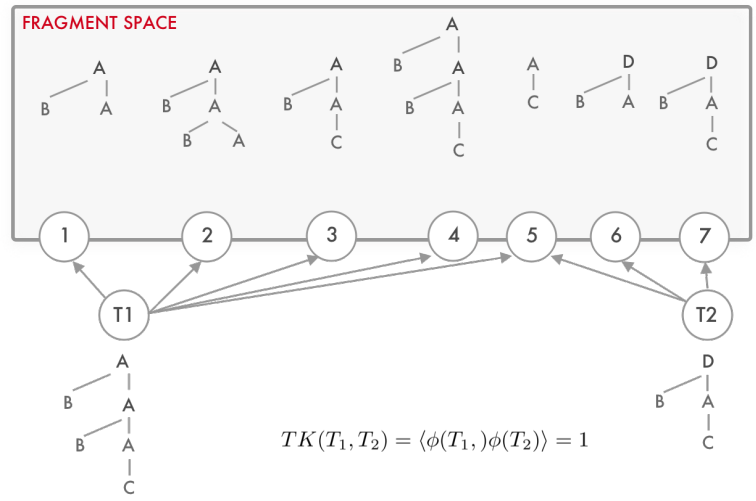


Figure 1: General idea of tree kernels computation. Given two tree, the possible fragments are generated in order to form the fragment space. Then, each tree is represent in terms of its subfragments and a dot product is applied to compute the similarity.

Given two input trees, TKs evaluate the number of substructures, also called fragments, that they have in common. More formally, let $\mathcal{F} = \{f_1, f_2, \ldots..f_{\mathcal{F}}\}$ be the space of all possible tree fragments and $\chi_i(n)$ an indicator function

such that it is equal to 1 if the target f_i is rooted in n, equal to 0 otherwise. TKs over T_1 and T_2 are defined by the following function:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \tag{12}$$

where N_{T_1} e N_{T_2} are the set of nodes of T_1 and T_2 and

$$\Delta(n_1, n_2) = \sum_{i=1}^{\mathcal{F}} \chi_i(n_1)\chi_i(n_2) \tag{13}$$

represents the number of common fragments rooted at nodes n_1 and n_2. The number and the type of fragments generated depends on the type of the used tree kernel functions, which, in turn, depends on $\Delta(n_1, n_2)$.

2.4.1.1 *Syntactic Tree Kernels (STK)*

The Syntactic Tree Kernel (STK) [88] computes the number of common substructures by using $\Delta_{STK}(n_1, n_2)$ in Eq. 13 , which is defined as follows [2]:

1. if the productions at n_1 and n_2 are different: $\Delta_{STK}(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children then: $\Delta_{STK}(n_1, n_2) = \lambda$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not preterminals then:

$$\Delta_{STK}(n_1, n_2) = \lambda \prod_{j=1}^{l(n_1)} (1 + \Delta_{STK}(c_{n_1}^j, c_{n_2}^j)) \tag{14}$$

where $l(n_1)$ is the number of children of n_1 and c_n^j is the j-th child of the node n.

The steps (2) and (3) can be modified to introduce a decay factor:

5. $\Delta_{STK}(n_1, n_2) = \lambda$;
6. $\Delta_{STK}(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta_{STK}(c_{n_1}^j, c_{n_2}^j))$.

Finally, by adding the following step:

1. if the nodes n_1 and n_2 are the same then: $\Delta_{STK}(n_1, n_2) = \lambda$.

2 In a syntactic tree, each node can be associated with a production rule

also the individual nodes will be counted by Δ_{STK}. We call this kernel STK_b.

Note that, since the productions are the same, $l(n_1) = l(n_2)$ and the computational complexity of STK is $O(|N_{T_1}||N_{T_2}|)$ but the average running time tends to be linear, i.e., $O(|N_{T_1}| + |N_{T_2}|)$, for natural language syntactic trees [63]. The main characteristic of STK is that production rules of the grammar used to generate the tree will not be broken, meaning that siblings separation is not allowed.

2.4.1.2 *The Partial Tree Kernel (PTK)*

[63] generalizes a large class of tree kernels[3] as it computes one of the most general tree substructure spaces. Indeed, PTK allows the children of a node to be separated to generate new fragments. It turns out that the number of fragments generated is greater than with STK, making PTK more expressive.

Given two trees, PTK considers any connected subset of nodes as possible features of the substructure space. Its computation is carried out by Eq. 13 using the following Δ_{PTK} function:

1. if the labels of n_1 and n_2 are different: $\Delta_{PTK}(n_1, n_2) = 0$;
2. else:

$$\Delta_{PTK}(n_1, n_2) = 1 + \Big(\sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_{PTK}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j})) \Big) \quad (15)$$

where $\vec{I}_1$ and $\vec{I}_2$ are two sequences of indices, which index subsequences of children u, $\vec{I} = (i_1, ..., i_{|u|})$, in sequences of children s, $1 \leq i_1 < ... < i_{|u|} \leq |s|$, i.e., such that $u = s_{i_1}..s_{i_{|u|}}$, and $d(\vec{I}) = i_{|u|} - i_1 + 1$ is the distance between the first and last child.

As with STK, Equation 14 can be modified to add two decay factors: μ for the depth of the tree and λ for the length of the child subsequence with respect to the original sequence. Hence, the derived λ_{PTK} function is defined as follows:

$$\Delta_{PTK}(n_1, n_2) = \mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_{PTK}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big) \quad (16)$$

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11} + 1$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21} + 1$. This ensure that both larger trees and child subsequences with gaps are penalized.

3 It also generalizes string/sequence kernels, i.e., when the tree is just a root with a sequence of children.

The PTK computational complexity is $O(p\rho^2|N_{T_1}||N_{T_2}|)$ [63], where p is the largest subsequence of children that we want to consider and ρ is the maximal outdegree observed in the two trees. However the average running time tends to be linear for natural language syntactic trees [63].

2.4.1.3 *Smoothed Partial Tree Kernel (SPTK)*

Tree Kernels such as STK and PTK require and hard matching to the node, which in case of similar words leads to a loosing of relevant information. The Smoothed Partial Tree Kernel [27] is a convolution kernel for dependency structures that aims at jointly modeling syntactic and lexical semantic similarity. More in detail, this kernel can measure the similarity of structural similar trees whose nodes are associated with different but related lexicals.

The SPTK computation is carried out by using $\Delta_\sigma(n_1, n_2)$ in Eq. 13 defined as follows:

1. If n_1 and n_2 are leaves than: $\Delta_\sigma(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)$;
2. else:

$$\Delta_\sigma(n_1, n_2) = \mu\sigma(n_1, n_2) \times \left(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_{sigma}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\right) \tag{17}$$

where σ is a similarity function between nodes, e.g. between their lexical labels, and the other variables are the same of PTK.

3

URBAN DATA

3.1 DATA DEFINITION AND PREPROCESSING

3.1.1 *Definition*

Urban data provides information about a physical object in the urban environment that can be represented by numerical values in a Coordinate Reference System (CRS). A common choice for CRS is the Geographical Coordinate System (GCS) WGS84, which is a coordinate system that enables representing every location on Earth by a set of coordinates, i.e., longitude, latitude and elevation. An object in the physical space can be described by means of points, lines, and polygons, e.g., the shape of a city, a river, or a point-of-interest. While coordinates are descriptive enough for objects identified by a point, to represent more complex structures, e.g., by a such as city boundaries, shape of the buildings or road networks, a different approach is necessary. A possibility is to use the widely adopted ESRI Shapefile format, the *de facto* standard for spatial analysis and data exchange. It stores the data as primitive geometric shapes like points, lines, and polygons, which together with the associated data attributes create the representation of the geographic data. Those attributes add semantics to the object and can be represented with textual or image data.

3.1.2 *Spatial Aggregation*

Given the heterogeneity of currently available sources, the availability of comparable spatial data is a notorious obstacle when reconstructing characterizations of urban environments across geographical domains. Geographic data is regularly captured and stored in different formats as point data and polygon data, which means that it is necessary to join these files in order to use them.

Once all the points are represented with the same CRS, a common approach to solve this problem consist of aggregating them in a tessellation built over the original polygonal shape, e.g., the polygon that contains all the points of the datasets. In some tasks, such as next location prediction in human mobil-

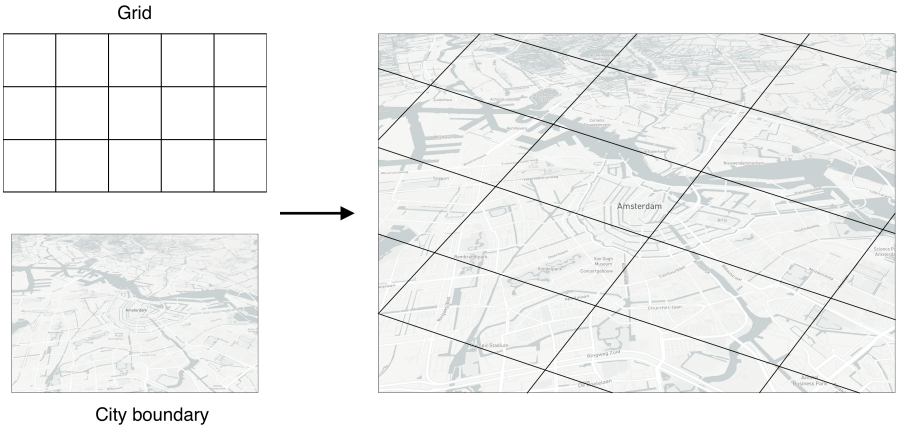


Figure 2: Building process of a spatial tessellation over the city boundary of Amsterdam.

ity, spatial aggregation is also applied to reduce the complexity of the problem and discretize the space by mapping the coordinates to such tessellation. In more detail, the spatial aggregation process is divided in two phases: the first phase aims at creating the tessellation covering the shape that contains all the points in the dataset. Fig.2 shows an example with a regular tessellation with square cells covering the shape representing the city boundaries. Let C be the polygon composed by q vertices $v_1, ..., v_q$ that define the city boundaries. We define the grid G as the tessellation covering C with $m \times n$ squared cells: $G = \{c_{ij}\}_{i=1,..,m;j=1,...,n}$, where c_{ij} is the squared cell (i, j) defined by two vertices representing the top-left and bottom-right coordinates. Depending on the nature of the problem, such tessellation can be formed with any triangle or quadrilateral tile or, as in the case of Voronoi tessellation, with tile defined as the set of points closest to one of the points in a discrete set of defining points. During the second phase, a spatial join operator is applied between the tessellation and all the points contained in the dataset. In a spatial join, two geometry objects are merged based on their spatial relationship to one another. Thus, in the output each geometric location is assigned to an area.

3.2 URBAN DATA SOURCES

Urban spaces are a physical manifestation of continuous processes of human interaction and of the weaving and knitting of social relations unfolding through space and stratifying through time. Cities are inherently hard to understand; their interlacing facets and tangled assemblages render them difficult to frame

[18, 22]. In this section we review the frequently used data sources about urban data together with representative example applications.

3.2.1 *Location-Based Social Networks (LBSNs)*

For urban study purposes, it is very important to have an up-to-date image of the status as all the urban phenomena and changes over time may occur. A help in this direction is given by sensing technologies and Location-based Social Networks (LBSNs), e.g., Foursquare[1], which have increased the production and availability of geographical data. Foursquare is a well-known LBSN, which provides information about key social, cultural, and infrastructural components of an area by capturing its Point-Of-Interests (POIs). Moreover, Foursquare allows users to register their presence in a place by means of a *check-in* action, which collects their latitude and longitude coordinates and additional metadata such as the check-in venue, comments and photos. A POI is usually characterized by a location (i.e., latitude and longitude), textual information (e.g., a description of the activity in that place) and a hierarchical categorization that provides different levels of detail about the activity of the place (e.g., *Food*, *Asian Restaurant*, *Chinese Restaurant*). For example, for the activity, *Chinese Restaurant*, we have the path *Food* → *Asian Restaurant* → *Chinese Restaurant*.

An example where LBSN data can be used it the automatic analysis of land use, which enables the possibility of better administrating a city in terms of resources and provided services. However, such analysis requires specific information, which is often not available for privacy concerns. For instance, in [92] used the *check-in* information available in Foursquare to classify the land use in New York City (NYC). In particular, they applied extensive feature engineering, mainly based on spatio-temporal data of check-in actions, and they built a Gold Standard from the data provided by the NYC Department of City Planning 2013, mapped on a tessellation with squared cells of 200×200 meters that covers the whole NYC area.

3.2.1.1 *Traffic*

World population is increasingly moving from rural areas to urban centers. Actually, approximately 54% of people worldwide live in cities, especially in metropolises, which offer better working and leisure opportunities. For instance, in urban areas there is greater access to work, a wide variety of op-

1 https://foursquare.com

tions for education and training, ease of transport and abundance of attractive places within a few kilometers. Across huge cities people tend to move more and have to do it faster than in the past. On the other hand, heavy traffic (e.g., traffic jams, severe traffic congestions, etc.) can cause noise and atmospheric pollution (i.e., smog, hydrocarbon concentration, and exhaust gas emissions). Optimizing the public transportation system can therefore help in improving the quality of citizens' lives, both by facilitating their mobility and ensuring their health.

The research in this field can also support transport companies to provide a better service, in terms of waiting times for the customers, fuel saving, traffic reduction, and ease of mobility.

In our work [14] we propose Achilles, a web-based application which provides an easy-to-use interface to explore the mobility fluxes and the connectivity of urban zones in a city, as well as to visualize changes in the transport system resulting from the addition or removal of transport modes, urban zones and single stops. Achilles is based on a multiplex network representation of mobility data [19] for the city of Singapore. Each layer describes people's movements with a given transport mode, e.g. buses, metros, taxis. A node in a layer represents an urban zone of the city, edges indicate routes between zones and edge weights indicate the amount of people moving between two nodes in a given time window. Achilles exploits this network representation and allows the user to visualize changes in the transport system resulting from the addition or removal of transport modes, urban zones and single stops. Notably, Achilles allows to compute the *Heel-ness* of a zone, a measure introduced in this paper which indicates the probability of a zone to disconnect the network when edges adjacent to it are removed. The computation of the *Heel-ness* allows Achilles to state that the transport system has an *Urban Achilles Heel* if there is at least one urban zone with a non-zero *Heel-ness*.

3.2.2 *Mobile Phone*

There are almost 6 billion mobile phone users worldwide. The world coverage has raised from 12% of the world population in 2000 up to 96% in 2014 [1], and this number even reaches 100% of population in the developed countries. These devices generate an incredible amount of data on how we use our mobile phone daily and how we interact with other people. Furthermore, they contain location data (e.g. from where a person calls) that make people's movements

easily traceable through the antennas to which they are connected or, even better, with ad hoc applications that register the GPS tracks.

The availability of these data is indeed defining a novel area of research that exploits CDRs to extract human mobility patterns [65, 64] and social interactions [67, 66], estimates population densities [55], models citiy structures [68], predicts socio-economic indicators and outcomes of territories [69, 70, 71], and models the spread of diseases [68, 91, 72, 89] (See Blondel *et al.* [90] for a comprehensive review of recent advances in studies using mobile phone datasets).

3.2.3 *Environment*

As in other scientific disciplines, also in environmental science data-driven approaches have been successfully applied to key problems such as climate change and air quality monitoring. The resolution of such tasks provides new understanding of the complex natural system and its mechanisms that contribute to climate change, such as the growing intensity of hurricanes [33], weather prediction [32], and the occurrence of extreme weather events that results in unprecedented environmental and socioeconomics disasters. Environmental data demonstrated to be effective also in other field such as social science.

3.2.4 *Social Media*

The emergence of social media user-generated data, such as text, photos and video, introduces further opportunities for researchers to quantitatively inspect different aspects of human behaviour such as the social well-being of individuals and communities [73] and socio-economic status of geographical regions [74]. By adding the location to such data records we can, for instance, model people movements and activities. In [107] they propose to predict the taxi pick-up zones in New York City (NYC), based on start/end points of trajectories and Twitter data, while Frias *et al.* [58] used tweets as input data to predict the land use of a certain areas of Manhattan.

Even more promising is the study of datasets combining social media data with CDRs and other economic and demographic indicators [83, 75].

3.2.5 *Governments and Municipalities*

The Census dataset represents an interesting source of information that can be linked to other data, for example, to understand and predict the socio-economic well-being of a given territorial area. Census data is a periodical collection where sociodemographic and economic information about household members are collected by means of survey directly filled by the householders. An interesting part of Census data is the one concerning location habits, such as workplace and previous residence, of the household members. For instance, Census data was used by Simini *et al.* [31] to test the radiation model, a human mobility model used to describe flows of people between different locations.

Another example of administrative dataset is represented by land use, which assign to each urban zone a specific use class, e.g., Residential, Commercial, Manufacturing. An interesting combination of different administrative data is the one proposed by De Nadai *et al.* in [53]. They empirically investigated the urban conditions that makes a city's area lively, by combining CDRs, social media data (i.e. Foursquare's Point of Interests), Open Street Map data, and census and land use data.

3.2.6 *People's Mobility*

The availability of data that describe human mobility (e.g., GPS traces, mobile phone records, social media records) is a trend that will grow in the near future. In particular, this will happen when the shift from traditional vehicles to autonomous self-driving car, will transform our society, the economy and the environment.

Mobility behaviors have been captured mainly by (i) Call Detail Records (CDRs) (Sec. 3.2.2), (ii) social media networks (See Section 3.2.4), and by (ii) smartphone applications that collect user movements by leveraging on Global Positioning System (GPS) data. Indeed, it is worth noticing that both CDRs are based on the cell towers of a provider, thus resulting in a coarser spatial granularity with respect to the GPS data. In addition, CDRs and social media data suffer from low temporal resolution since they are event-driven (i.e. records are created by a call/SMS trigger or through the user check-in), while the GPS overcome this since they are generated independently of the phone usage [52].

3.3 A MULTI-SOURCE URBAN LIFE DATASET

Unfortunately the availability of communications and social media data is usually restricted to a few research teams that sign non-disclosure agreements (NDAs) and research contracts with telecommunication and other private companies. Thus, the lack of open datasets limits the number of potential studies and creates issues in the process of validation and reproducibility needed by the scientific community. In this context, research challenges that provide access to a large number of research teams to the same dataset are becoming a truly valuable framework to advance the state of the art in the field. A prototypical example is offered by Orange's "Data for Development" (D4D) initiative in 2013 [82] and 2014-2015 [81]. Analogously, Telecom Italia organized the *Telecom Italia Big Data Challenge*, providing various geo-referenced and anonymized datasets. In the 2014 edition they provided data of two Italian areas: the city of Milan and the Province of Trentino. More than 650 teams from more than 100 universities have participated in this Challenge. In addition, the data pertaining to the challenge have been released to the research teams under the Open Database License (ODbL), thus triggering a long tail of follow on research work based on these data [80, 79, 78, 30, 77].

In this section, we describe our work on the richest open multi-source dataset ever released on two geographical areas [1]. The dataset is composed of most of the urban data introduced in this chapter: telecommunication, weather, news, social networks and electricity data from the city of Milan and the Province of Trentino (see Figures 2 and 3). The unique multi-source composition of the dataset makes it an ideal testbed for methodologies and approaches aimed at tackling a wide range of problems including energy consumption, mobility planning, tourist and migrant flows, urban structures and interactions, event detection, urban well-being and many others.

The multi-source nature of the current dataset permits the modeling of multiple dimensions of a given geographical area and to address a variety of problems and scientific issues that range from the classic human mobility and traffic analysis studies to energy consumption and linguistic studies. The dataset has been released to the whole research community and here we provide a detailed description of the data records' structure, and present the methodology used in the data collection/aggregation process. This dataset represent an unique resource for the research community that widely is using it in many interesting studies and applications. Among the others, this dataset has been

used to estimate population [30], relationship between urban communication and happiness [29], energy consuption forecasting [28] and network demand prediction [35].

The datasets are released under the Open Database License (ODbL) and are publicly available in the Harvard Dataverse [2].

Dataset type	Issuer	Area
Grid	Telecom Italia	Milan, Trentino
Social Pulse	SpazioDati, DEIB	Milan, Trentino
Telecommunications	Telecom Italia	Milan, Trentino
Precipitations	Meteotrentino, ARPA	Milan, Trentino
Weather	ARPA	Milan, Trentino
Electricity	SET Distribuzione SPA	Trentino
News	Citynews	Milan, Trentino

Table 2: Dataset types and issuers.

3.3.1 *Data Records*

Since the datasets come from various companies which have adopted different standards, their spatial distribution irregularity is aggregated in a grid with square cells by applying the methodology presented in Section 3.1.2. This allows comparisons between different areas and eases the geographical management of the data. Thus, the area of Milan is composed of a grid overlay of 10000 squares with size of about 235×235 meters and Trentino is composed of a grid overlay of 6575 squares (see Figure 4). This grid is projected with the WGS84 (EPSG:4326) standard.

3.3.1.1 *Telecommunications activity and interactions*

The Call Detail Records (CDRs) are provided by TIM[3]. Every time a user engages a telecommunication interaction, a Radio Base Station (RBS) is assigned by the operator and delivers the communication through the network. Then, a new CDR is created recording the time of the interaction and the RBS which

2 https://dataverse.harvard.edu/dataverse/bigdatachallenge
3 https://www.tim.it/

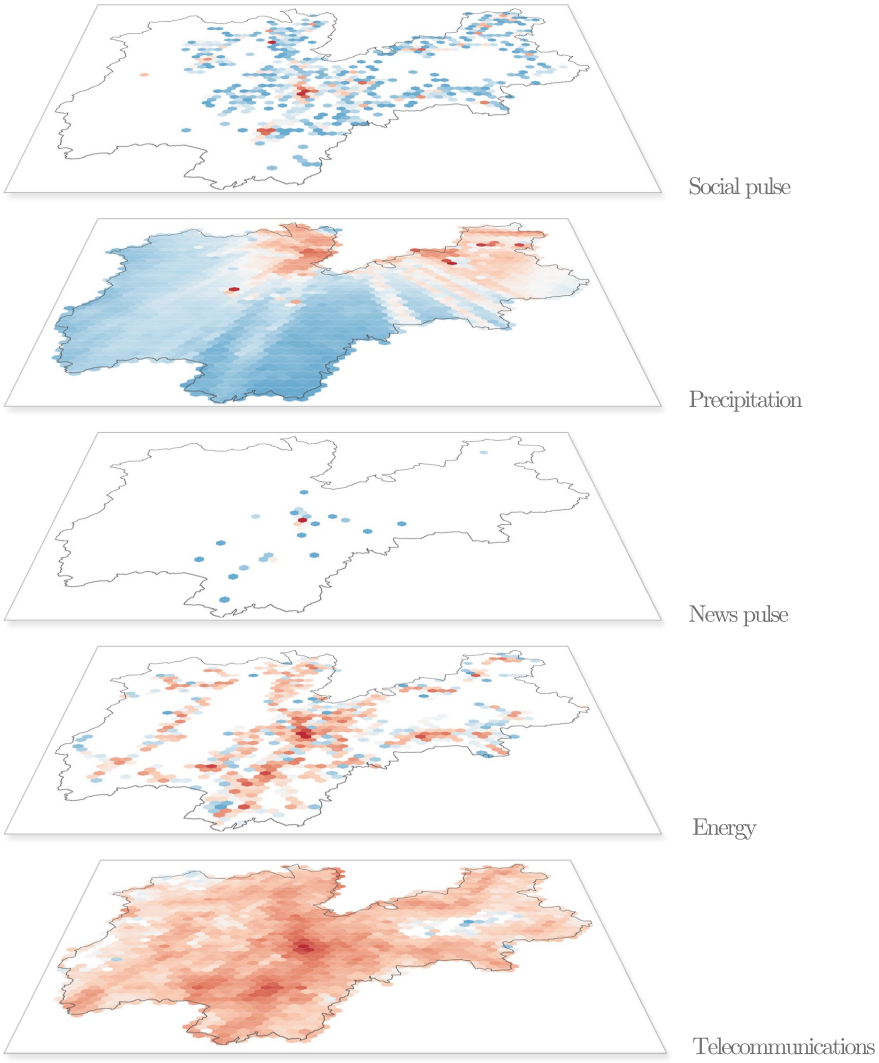


Figure 3: Hexbin map with logaritmic color scale of the Province of Trentino. Each layer represents a specific dataset. In the energy layer the red color represents the sum of consumed electricity. In the precipitation layer colors go from blue (minimum mean intensity of precipitations) to red (the maximum one). In the other layers the blue color represents the minimum number of events (e.g. connections, tweets, news), while the red the maximum number of events. The News pulse map is generated from the News dataset, which is only available for Trento while the Social Pulse map shows the high concentration of Tweets in the biggest cities of Trentino.

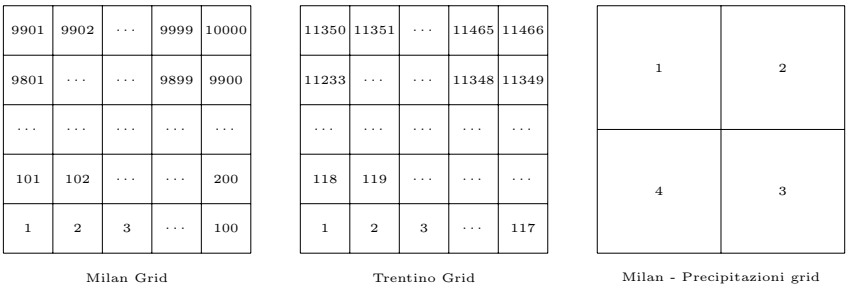


Figure 4: The various grid systems employed in this project.

handled it. From the RBS it is possible to obtain an *indication* of the user's geographical location, thanks to the coverage maps C_{map} which associates each RBS to the portion of territory which it serves (AKA coverage area, Figure 5).

In order to spatially aggregate the CDRs inside the grid, each interaction is associated with the coverage area v of the RBS which handled it. Hence, the number of records $S_i(t)$ in a grid square i at time t is computed as follows:

$$S_i(t) = \sum_{v \in C_{map}} R_v(t) \frac{A_{v \cap i}}{A_v} \tag{18}$$

where $R_{v,j}(t)$ is the number of records in the coverage area v at time t, A_v is the surface of the coverage area v and $A_{v \cap i}$ is the surface of the spatial intersection between v and the square i.

There are many types of CDRs and Telecom Italia has recorded the following activities:

RECEIVED SMS a CDR is generated each time a user receives an SMS

SENT SMS a CDR is generated each time a user sends an SMS

INCOMING CALL a CDR is generated each time a user receives a call

OUTGOING CALL a CDR is generated each time a user issues a call

INTERNET a CDR is generated each time a user starts an Internet connection or ends an Internet connection. During the same connection a CDR is generated if the connection lasts for more than 15 minutes or the user transferred more than 5 MB.

The shared datasets were created combining all this anonymous information, with a temporal aggregation of time slots of ten minutes. The number of records in the datasets $S_i'(t)$ follows the rule:

$$S_i'(t) = S_i(t)k \tag{19}$$

where k is a constant defined by Telecom Italia, which hides the true number of calls, SMS and connections.

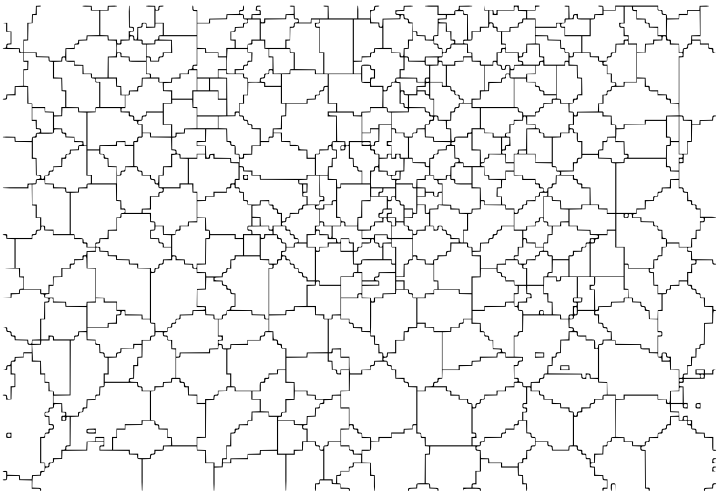


Figure 5: An example of coverage map of Milan.

The first type of dataset built by using CDRs represents the activity of Trentino and Milan, showing all the aforementioned telecommunication events which took place within these areas. The data provides information of TIM's customers interacting with the network and of other people using it while roaming.

Two types of CDR datasets were also produced to measure the interaction intensity between different locations: one from a particular area (Trentino/Milan) to any of the Italian provinces and one quantifying the interactions within the city/province (e.g. Milan to Milan). Since TIM only possesses the data of its own customers, the computed interactions are only between them. This means that (at most) 34% of population's data is collected, due to Telecom Italia's market share [4]. Moreover there is no information about missed calls.

4 http://www.agcom.it/documents/10179/1734740/Studio-Ricerca+24-07-2014/5541e017-3c7a-42ff-b82f-66b460175f68?version=1.0, date of access 06/08/2014

3.3.1.2 *Social Pulse*

The Social Pulse dataset is composed of geo-located tweets that were posted by users from Trentino and Milan between November 1, 2013 and December 31, 2013. The stream was gathered through the Twitter Streaming API (https://dev.twitter.com/docs/streaming-apis) which is a free service allowing the extraction of ~ 1% of the total Twitter feed through a set of filterers provided by the user. This process saves the author username, the tweet content and the time-stamp when the tweet has been written. In order to ensure the privacy of the original users, their username has been obfuscated and the text of the tweet has been replaced with a list of entities extracted by the *dataTXT-NEX* tool (https://dandelion.eu/products/datatxt/), a tool to identify meaningful sequences of one or more terms, and then to link them to the most appropriate Wikipedia page. The obfuscation of the username has been done using the hash function SHA-1, and two random generated strings (SALT1 and SALT2):

$$username_{new} = sha1(SALT1 + username + SALT2) \tag{20}$$

3.3.1.3 *Weather station data*

The weather data describes meteorological phenomena type and intensity in Milan and Trentino. The data of Milan are collected by Agenzia Regionale per la Protezione dell'Ambiente (ARPA)[5] while Trentino's data are collected by Meteotrentino [6].

MILAN In Milan, the type and the intensity of the phenomena are continuously measured by different sensors located within the city limit. Each sensor has a unique ID, a type and a location. Different sensors can share the same location. The data are split into two datasets called *Legend dataset* and *Weather Phenomena*. Intuitively, the former provides the locations of the sensors and the unit of measurements, while the latter contains the measurement files for each sensor. The sensors can measure different meteorological phenomena: Wind Direction, Wind Speed, Temperature, Relative Humidity, Precipitation, Global Radiation, Atmospheric Pressure and Net Radiation. There is no spatial aggregation and the data is aggregated in 60 minute time-slots.

5 http://www.arpalombardia.it/siti/arpalombardia/meteo/richiesta-dati-misurati/Pagine/RichiestaDatiMisurati.aspx

6 http://www.meteotrentino.it

TRENTINO The dataset contains measurements about temperature, precipitation and wind speed/direction taken in 36 Weather Stations placed around the Province of Trentino. There is no spatial aggregation and the data are aggregated in timeslots of 15 minutes.

3.3.1.4 *Precipitation*

The precipitation datasets provide information about precipitation intensity and type over the geographical area. The data of Milan and Trentino are collected by ARPA (`http://www.arpa.piemonte.it/rischinaturali`) and by Meteotrentino respectively. Since they adopt different standards, we organized two sections to describe them.

MILAN This dataset is temporally aggregated every 10 minutes and spatially aggregated in four quadrants of equal size of 11.75×11.75 km, corresponding to 50 squares of the grid used for the aggregation.

The precipitation types are described as:

- *Absent (type 0)*: precipitation quantity equal to 0 mm/h;
- *Slight (type 1)*: precipitation quantity equal in $[0, 2]$ mm/h;
- *Moderate (type 2)*: precipitation quantity equal in $[2, 10]$ mm/h;
- *Heavy (type 3)*: precipitation quantity equal to in $[10, 100]$ mm/h.

while the precipitation intensity is characterized as *Absent* (type 0), *Rain* (type 1) and *Snow* (type 2).

TRENTINO The precipitation intensity values for Trentino are spatial aggregated over the Trentino grid and temporal aggregated every 10 minutes and they follow the standard described as:

- *very slight*: precipitation intensity defined $[1, 3]$ meaning an amount of $[0.20, 2.0]$ mm/hr;
- *slight*: precipitation intensity defined $[4, 6]$ meaning an amount of $[2.0, 7.0]$ mm/hr;
- *moderate*: precipitation intensity defined $[7, 9]$ meaning an amount of $[7.0, 16.0]$ mm/hr;
- *heavy*: precipitation intensity defined $[10, 12]$ meaning an amount of $[16.0, 30.0]$ mm/hr;

- *very heavy*: precipitation intensity defined $[13, 15]$ meaning an amount of $[30.0, 70.0]$ mm/hr;
- *extreme*: precipitation intensity defined $[16, 18]$ meaning an amount of more than 70 mm/hr;

The precipitation data collection is not continuous due to some technical issues such as the presence of snow over the sensor radar. For this reason, we issued the *data availability* dataset which indicates whether the data has been collected or not for a specific time interval.

3.3.1.5 *SET Electricity*

SET manages almost the entire electrical network over the Trentino territory. It uses around 180 primary distribution lines (medium voltage lines) to bring energy from the national grid to Trentino's consumers. To ensure the privacy of SET's customers, their locations and the geometry of the 180 primary distribution lines is not explicitly exposed. Consequently, the *Customer site dataset* shows the number of customer sites of each power line per grid square, while the *Line measurement dataset* indicates the amount of flowing energy through the lines at time t. Customer sites provide energy to different types of customers (e.g. houses, condominiums, business activities, industries etc.), which require different amount of electricity. For privacy reasons this information is hidden, meaning that in the dataset the energy flowing is uniformly distributed among the various types of customers.

Figure 6 shows the process we have done to transform the original dataset to the shared one. In the first layer we have the exact position of each customer site (e.g. some of them are industries, others are small houses) and the precise geometry of each line. In the second layer we lose the exact geometries of customer sites and power lines. However, this information is summarized in the *Customer site dataset* where for each square grid the number of customer sites is recorded along with the information about the power line they are connected to. In the third layer we know how the customer sites of a power line are distributed over the grid and the energy flowing through each power-line (from the *Line measurement dataset*). It is then possible to distribute the energy flowing through a powerline p over the grid in order to build a choropleth map of the energy consumption in each grid square (last layer in Figure 6).

The *Line measurement dataset* is temporal aggregated in time-slots of 10 minutes.

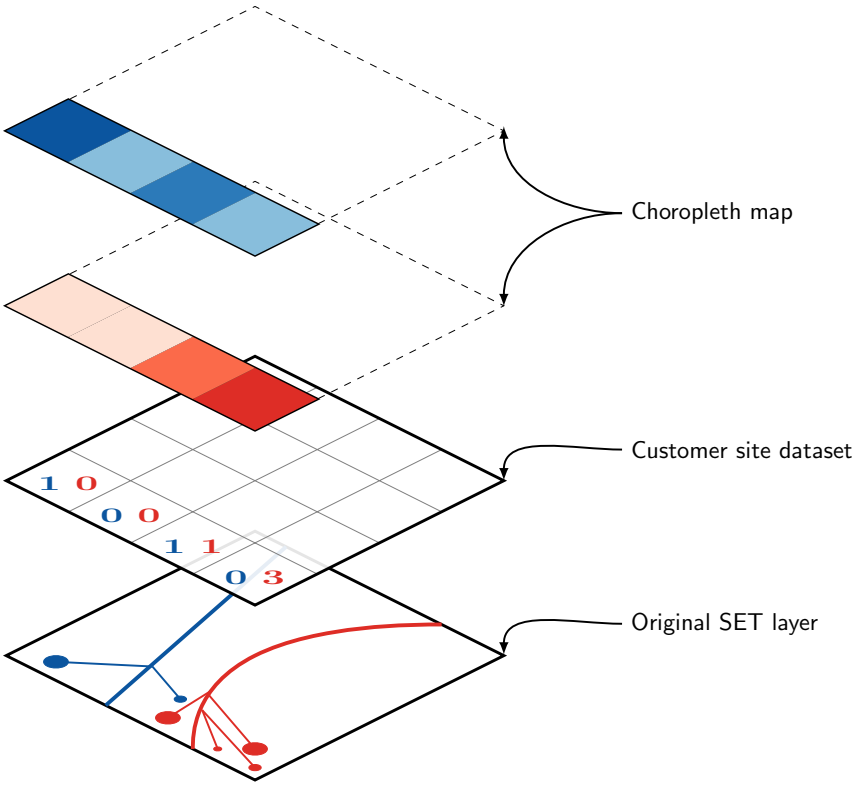


Figure 6: The SET customers are spatially aggregated into the grid squares and the energy consumption is uniformly divided among the customers, hiding their different type (e.g. houses, condominiums, business activities, industries).

3.3.1.6 *News*

The news datasets contain all the articles published on the websites Milano Today [7] and Trento Today[8]. Each news is referred to the geographical location where the event happened. All the news referring to the general area (the whole city of Milan or the whole Province of Trentino) are geo-tagged to its administrative centre.

3.3.1.7 *Grid*

Some of the datasets are spatially aggregated using a regular grid covering the territory. The Grid dataset provides the geographical reference of each square which composes the grid in the reference system: WGS 84 - EPSG:4326.

7 http://www.milanotoday.it
8 http://www.trentotoday.it

3.3.1.8 *Census Data*

The presented datasets can be enriched by using census data provided by the Italian National Institute of Statistics (ISTAT) [9], a public research organization and the main provider of official statistics in Italy. The census data have been released for 1999, 2001 and 2011. However, he resolution of the data is not uniform over the national territory. Urban areas have a resolution of 1:50.000, while areas with low population density have a resolution of 1:25.000.

The dataset[10], released in Italian, is composed of four parts: Territorial Bases (Basi Territoriali), Administrative Boundaries (Confini Amministrativi), Census Variables (Variabili Censuarie) and data about Toponymy (Dati Toponomastici). The first set contains the geographical shapefile data of all the Italian regional areas. The second set is composed of the administrative boundaries used in the last three censuses. The third contains census variables, divided into eight different groups: residential population, foreign population, families, education level, work status, commuting, accommodations info and building composition. The last set contains all the information about civic numbers and maps used in the census of 2011. The Census dataset represents an interesting source of information that can be linked to the data described in this paper to, for example, understand and predict the socio-economic well-being of a given territorial area.

For each information point I referring to a geographical area v (contained in the shapefile), we can calculate the proportion of data which belongs to each GRID's square g:

$$I_g = I_v \left(\frac{A_{v \cap g}}{A_v} \right) \tag{21}$$

where A_p is the area of a polygon p. After this process, the ISTAT data is correctly linked to the GRID.

3.3.2 *Statistical and Visual Characterization*

The technical quality validation of the datasets is limited due to the absence of similar datasets to compare our results with. Hence, we propose a statistical

9 http://www.istat.it/en/

10 http://www.istat.it/it/archivio/104317, date of access 09/09/2015

and visual characterization with the aim of supporting the naive correctness of the information provided.

3.3.2.1 *Temporal aspects*

Generally, people perform different activities during the day. Many of them are repeated on a daily basis (e.g. eating at noon, jogging in the evening etc.), others on a weekly basis (e.g. watching the favourite football team at the stadium). From Figures 7 and 8 it is possible to observe a strong daily seasonality which usually starts at 7:00, when people turn on their phones and probably commute to work and then slowly decreases in the evening when people return home and sleep. Moreover, there is also a weekly seasonality due to the work cycles behaviour of people (e.g. working days vs. weekends).

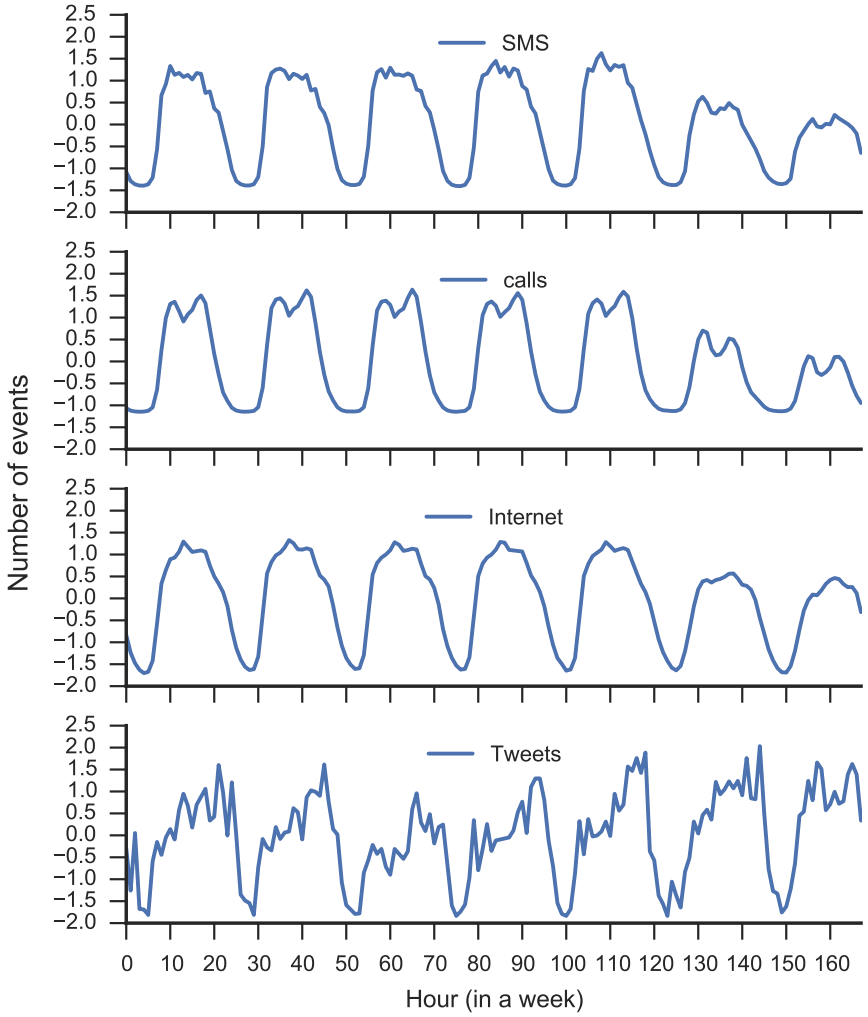


Figure 7: Weekly Z-scaled behavior of SMS, calls, Tweets and Internet CDRs in Milan.

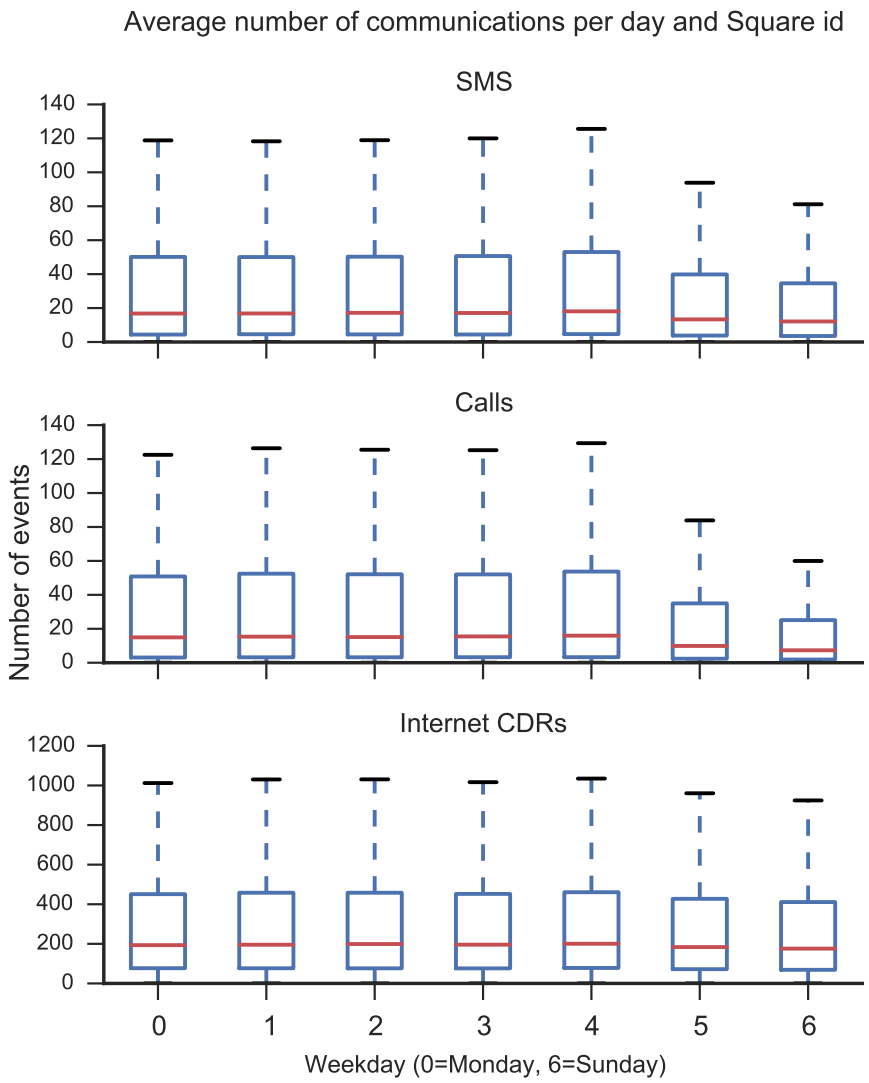


Figure 8: Box-plots showing the calls, SMS, and Internet CDRs distributions per weekday and per cell in Milan.

Similarly, Twitter data (see Figure 7) have a strong daily seasonal component which starts in the early morning and increases during the day, having a peak around 22:00. Nonetheless, there is also a soft weekly seasonality observable especially comparing Sunday and Monday. Instead, news stories exhibit a strong weekly seasonality which is probably due to work cycles, since Saturdays and Sundays less news are published (on the website) respectively to other days.

Since it is not possible to have a well-established ground truth for the data, some important events with expected high importance for Milan were selected to validate it. For example, we observed the stadium area of Milan and we noticed a steep increase in the number of communications in this area compared to other days. Similarly, this happened for the New Year eve in all areas of Mi-

lan and Trentino. This test suggests that the data correctly reflects the temporal human behavioural patterns for the two areas considered.

3.3.2.2 *Spatial aspects*

We compare some locations that we expect to have markedly different behavioural signatures. We select the following areas:

- Bocconi, one of the most famous Universities in Milan (*Square id*: 4259);
- Navigli district, one of the most famous nightlife places in Milan (*Square id*: 4456);
- Duomo, the city centre of Milan (*Square id*: 5060);
- Duomo, the city centre of Trento (*Square id*: 5200);
- Mesiano, the department of Engineering of the University of Trento (*Square id*: 5085);
- Bosco della città, a forest near Trento (*Square id*: 4703).

As depicted in the mobile phone usage plot (see Figure 9), the selected areas show very different behavioural patterns. As expected, Navigli is characterized by an increase in Internet connections during the evening, while Bocconi's connections drop off during the weekends. Moreover, Bocconi has less mobile phone activity than Duomo, which is the centre of the city and the most important tourist attraction.

Similarly, in Trentino we expect to have many connections in the city centre, a lower number of connections at Mesiano, which is outside the city centre and very few connections in Bosco della città, because of its position and function. The plot confirms our expectations.

This proves the consistency between the expected spatial behaviour and the observed one.

3.3.3 *Metrics*

In the Telecommunications and Social pulse datasets, we provided record level data which are not algorithmically aggregated on purpose. Indeed, we want to share data as close as possible to the raw data. The reason is that our goal is to give researchers the possibility both to extract known metrics and to design new ones. In the following we discuss some examples of metrics which can be extracted from the data.

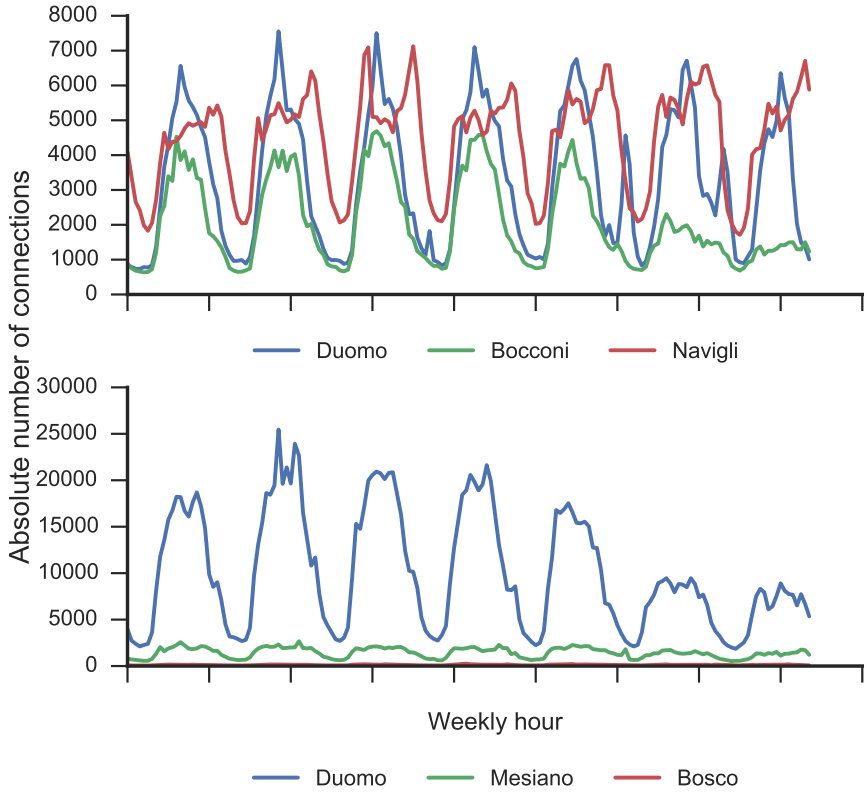


Figure 9: Weekly spatial behaviour of the six selected areas in Milan and Trentino.

3.3.3.1 *Hotspots*

The Telecommunications and Social pulse data make it possible to identify the *hotspots* of the city, defined as areas with high activity density with respect to the rest of the city. The simplest way to define *hotspots* is by choosing a threshold δ equal to the average of the city's activity, and considering *hotspots* as all the points with a density larger than that. However, this is only the minimal requirement. For this reason, it is possible to more restrictively define *hotspots* using the *Loubar* threshold introduced in [68]. The *Loubar* threshold is a time-dependent threshold that considers the inequality of the city, defined through the Lorenz curve of the density distribution of activity. From this definition, it is possible to study several behavioural aspects and cities' characteristics. For example, Louail *et al.* [68] designed various indexes to quantitatively define the typology of cities and their spatial structure, such as:

- number of *hotspots*, which scales with the population following a power law;

- *hotspots*' relative importance that evolves during the day. Hence, it is possible to capture the evolution observing *permanent hotspots* (places that are important all day), *intermittent* (with a lifespan of only few hours per day) and *intermediate* (with a lifespan ~ 12 hours). Consequently, researchers can study cities through the lens of *hotspots' stability*;
- the spatial structure of *hotspots* and their aforementioned categories can be studied to determine the typology of a city (e.g. mono-centric cities).

3.3.3.2 *Network based metrics*

From the Telecommunications interactions datasets (e.g. Milan to Milan), it is possible to create a virtual network of an area that describes a who-calls-whom network. Similarly to the physical network where people and goods move, the virtual network determines how information and knowledge moves. Clarke *et al.* [70] defined this network using the D4D Senegal data, and computed the following metrics:

- the *gravity residual*, meaning the difference between observed and expected flows of calls;
- the *network advantage*, which measure an area's interaction entropy;
- the *introversion* which compares the outgoing, incoming and total volume of traffic.

These metrics were also linked to socio-economical data in order to estimate poverty levels in a region.

4

STRUCTURAL SEMANTIC MODELS FOR URBAN ZONE REPRESENTATION

In recent years, the increasing availability of data from popular social networks, mobile phones, and other sensing devices has revealed many aspects of our cities and helped researchers to better characterize cities. A notable example is by [83], who proposed a model for predicting the most important activity of an urban area using Foursquare and mobile phone data. Along a similar line, [56] explored the attractiveness of different NYC places, combining mobile phone data and geo-located photos from Flickr. [40] proposed an online, semi-supervised, multimodal embedding method for geo-located information with space, time and text. The joint embeddings have been evaluated on cross-type retrieval task by retrieving (i) location for a given keywords and (ii) activities for a given location. Several works have also targeted land use classification and functional area detection. For example, [49] proposed a framework to classify functionalities of an area for the city of Beijing using trajectories of taxicab and POIs. They designed a topic model to represent each function of an area as a topic, the region as a document, and POIs and trajectories as metadata and words, respectively. The problem of detecting functional areas has been also studied including the temporal aspect. [50] proposed a spatio-temporal approach for the detection of functional regions. They exploited three different clustering algorithms to better track how the functionality of a city's region changes over time. They extracted features from Foursquare's POIs and check-in activities in the region of Manhattan (New York). In particular, their model included POIs' information by combining Latent Dirichlet Allocation and Dirichlet Multinomial Regression. The output was a vector representing the intensity of each topic (function) in the area, and could be used to aggregate regions with similar functions, applying k-means clustering.

In this chapter we focus on the design of semantic structures to model input urban zones by using LBSNs data, i.e., point-of-interests (POIs). First, we present new structural kernels that can automatically capture the most important patterns from the POI structure of city areas. These patterns are automatically engineered features that can be used in any urban related task such as the

classification of the land use of urban zones, crime prediction and human mobility. This chapter is based on our previous works [2][3] where we proposed to represent urban zones in three different ways: (i) as a bag-of-concepts (BOC) extracted from the Foursquare description of the area, e.g., *Arts and Entertainment*, *College and University*, *Event*, *Food*; and (ii) as the same concepts above organized in a hierarchical representation (HR), reflecting the hierarchical category structure of the LBSN activities; (iii) as a frequency hierarchical representation (FHR) that take into account also the multiple presence of the same POI category in the urban area.

Secondly, we designed kernels combining BOC vectors with Tree Kernels (TKs) applied to concept trees and used them in Support Vector Machines (SVMs). In particular, we apply standard Tree Kernel (TK) functions [61, 88, 62, 63], to HR to project them in the space of all their possible substructures such that each space dimension corresponds to a given semantic structural feature. However, this approach do not take into account the importance of different nodes, this mainly because it has been designed for natural language processing tasks, e.g., question answering [129]. In contrast, we assume that different POIs can assume different relevance because they repeat with different frequency in different areas. Thus, we define a new general Hierarchical POI Kernel (HPK), which can take the importance of different POIs into account by increasing the contribution of structure matches with POI weights, e.g., using the well-known $TF \times IDF$ heuristic.

Finally, we use a Support Vector Machine (SVM) with Tree Kernels to project structures in the space of all their possible substructures such that each dimension corresponds to a given semantic structural feature. We make these substructures explicit by means of a mining algorithm, which extracts the most relevant features (tree fragments) from the implicit TK space according to the weights the kernel machine assigned to them. Our approach can produce an explicit set of representative features that can be used to classify and characterize urban zones. This (i) improves the understanding of the target problem, (ii) highly speeds up both learning and classification as we can work in a linearized kernel space, and (iii) enables the use of structural features in other machine learning models, e.g., XGBoost.

The remainder of this chapter is structured as follows. Section 4.1 introduces our structured representation for urban zones, Section 4.2.1 is devoted to present our structural semantic models and, finally, Section 4.3 present our

mining algorithm to make an explicit set of features that can be used to classify and characterize urban zones.

4.1 GEO-SPATIAL REPRESENTATIONS

In this section we present our structural representation aimed at describing urban zones by means of geo-located semantic information, i.e., textual category of POIs. Given the hierarchical nature of LBSN data, we use tree structures as our base representation. Among the advantages of using trees, they provide a sufficient flexibility in representation and allow for easier feature extraction than, for example, graph structures. In addition, when used within the tree kernel learning framework, tree structures allow for efficient and powerful automatic feature engineering.

4.1.1 *Bag-Of-Concepts*

In Natural Language Processing (NLP), the first and simple idea to model text is the so-called *bag-of-words*. This representation treats words as atomic units mapping them to a finite-set of corresponding numbers with the same size of the word vocabulary. The main draw-back of such approach stems to the fact that words order is completely ignored. For example, the sentence *"Today is a wonderful day"* has the same BOW representation than one of its shuffled variant *"Wonderful day is today a"*. Despite the fact that the order, and thus the grammar, are not taken into consideration, this representation turned to be very accurate for a wide range of NLP applications. Similarly to the NLP, the most straightforward way to represent a zone by means of LBSN data is to use its POIs. Every venue is hierarchically categorized (e.g. *Professional and Other Places* → *Medical Center* → *Doctor's office*) and the categories are used to produce an aggregated representation of the zone. Thus, given an urban zone u and its set POI_u, a simple feature representations can be built by aggregating all the venues together, namely we count the category (e.g. *Food*) in all the POIs present in the cell. That is, the Bag-Of-Concepts (BOC) representation is generated by counting the number of each activity under each category. However, this simple representation does not take into account by the hierarchical structure of categories, and thus provide a very shallow representation of the urban space.

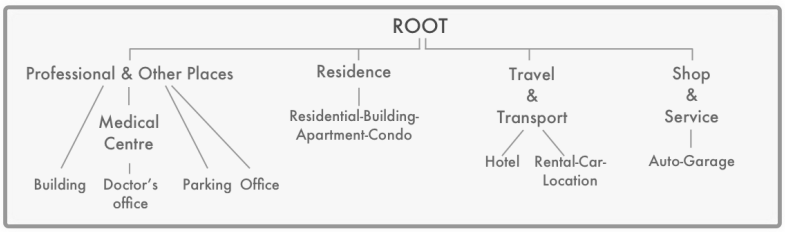


Figure 10: Example of GeoTree built from the hierarchical categorization of Foursquare venues.

4.1.2 *Geographical Tree (GeoTree)*

Every LBSN (e.g., Foursquare) has its own Hierarchical Category Tree (HCT), which is used to characterize each location and activity (e.g., restaurant or shops) in the database. Thus, each POI is associated with a hierarchical path, which semantically describes the type of location/activity (e.g., for *Chinese Restaurant* we have the path *Food* → *Asian Restaurant* → *Chinese Restaurant*).

The first representation we propose is the *Geo-Tree (GT)*, a tree structure where the nodes are LBSN categories and the edges among them are the same provided in its hierarchical category tree. Given an urban zone u, GeoTree is basically composed of all paths associated with the POIs that we find in the target zone u. Precisely, we connect all these paths in a new root node. This way, the first level of root children corresponds to the most general category in the list (e.g., *Arts & Entertainment*, *Event*, *Food*, etc.), the second level of our tree corresponds to the second level of the HCT, and so on. The terminal nodes are the finest-grained descriptions in terms of category about the area (e.g., *College Baseball Diamond* or *Southwestern French Restaurant*).

For example, Figure 10 illustrates the semantic structure of an urban zone obtained by combining all the categories' chains of each venue.

The advantage of the structured representation over the features-based models comes from its ability to encode powerful contextual features in the form of tree fragments. Additionally, this path is much more informative than just the target POI name, as it provides feature combinations following the structure and the node proximity information, e.g., *Food* & *Asian Restaurant* or *Asian Restaurant* & *Chinese Restaurant* are valid features whereas *Food* & *Chinese Restaurant* is not.

4.1.3 *Frequency Geographical Tree (F-GeoTree)*

Despite the informative structure of GeoTree, this representation suffers from the problem of not encoding the frequency of the different POIs in a specific area, which is also a disadvantage with respect to simple BOC. To overcome this problem, we explore a different approach by proposing an alternative structure that take into account the multiple presence of the same POI, e.g., two *Food* & Café, in a given urban zone.

Given an Hierarchical Category Tree, an urban zone u and its associated set of Point-Of-Interests POI_u, we define the *Frequency Geo-Tree (FGT)* as the tree composed with all paths from such HCT associated with the all the $POI \in POI_u$, increasing the frequency associated to each node every time that node appears in a path. Precisely, we connect all these paths in a new root node and we count the category presence at each level of the structure. This way, the first level of root children corresponds to the most general category in the list (e.g. *Arts & Entertainment*, *Event*, *Food*, etc.) and it will also contains nodes with the highest frequency. Then, the second level of the tree corresponds to the second level of the HCT, and so on. The terminal nodes are the finest-grained descriptions in terms of category about the area (e.g. *College Baseball Diamond* or *Southwestern French Restaurant*). In this way, each node will have not only the associated category, but also the frequency representing the number of times such category has been found in the POIs list. Additionally, for each tree level, its nodes can have at most the same frequency of parent nodes, and have a frequency equal to or greater than the child nodes.

For example, Figure 11 shows the semantic structure of an urban zone u obtained by combining all the categories' chains of each venue.

4.2 SEMANTIC STRUCTURAL MODELS

A large effort in previous work with urban data [47, 83, 57] was devoted to engineering features in an attempt to encode various aspects of input objects that can help to learn better discriminative functions. However, these feature vectors have in general some limitations as they (i) only encode small amount of information available for the target area and (ii) do not capture relations between different elements in the urban zone, i.e., POIs.

As introduced in Section 2.3, a valid alternative to the tedious manual feature engineering process is to use kernels function that automatically gener-

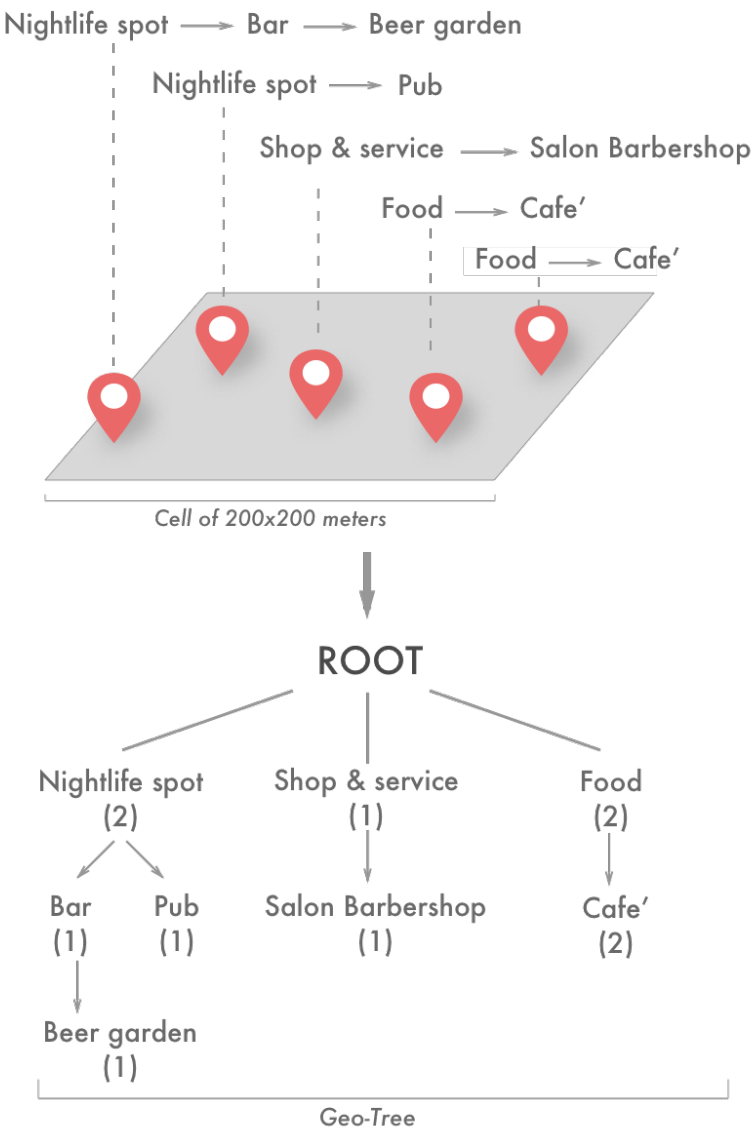


Figure 11: Example of Frequency Geo Tree built for a squared cell by using the hierarchical categorization of Foursquare venues.

ates a very large number of features representing the input objects. A class of learning methods building on kernels is represented by Kernel Machines (KMs), which allows for replacing the dot product with kernel functions directly applied to examples, i.e., they avoid mapping examples into vectors. The main advantage of KMs is a much lower computational complexity than the dot product as the kernel computation does not depend on the size of the feature space. Given a dataset $S = \{(\mathbf{x}_i, y_i) : i = 1, ..., n\}$ and a kernel function K, the classification function can be defined as:

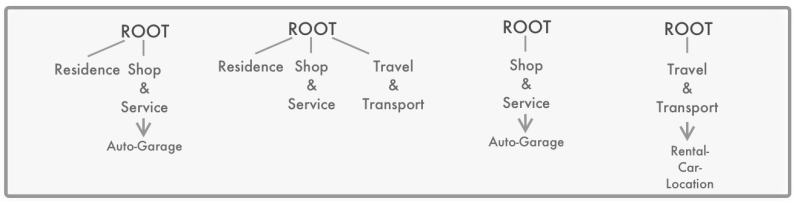


Figure 12: Some of the exponential fragment features from the tree of Figure 10 obtained by applying PTK.

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)\phi(\mathbf{x}) + b,$$

where $\mathbf{x}$ is a classifying example, $\mathbf{w}$ is the gradient of the separating hyperplane, and b its bias. The gradient is a linear combination of the training points $x_i \in \mathbb{R}^n$ multiplied by their labels $y_i \in \{-1, 1\}$ and their weights $\alpha_i \in \mathbb{R}^+$.

As discussed in Section 2.3, the scalar product can be replaced by a kernel function $K(o_i, o) = \phi(o_i)\phi(o)$ defined over a pair of objects $f(o) = \sum_i^n \alpha_i y_i K(o_i, o) + b$. Kernel functions implicitly define a mapping, $\phi : O \to \mathbb{R}^n$, from objects to vectors of the feature space.

As introduced in Section 2.4.1, Tree Kernels (TKs) measure a similarity between two trees, which a kernel machine can exploit to learn classification functions. Recalling the Equation 12, let $\mathcal{F} = \{f_1, f_2, \ldots..f_{\mathcal{F}}\}$ be the space of all possible tree fragments, and $\chi_i(n)$ an indicator function equals to 1 if the target f_i is rooted in n, equal to 0 otherwise. Tree Kernels over T_1 and T_2 are defined by the following function:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2),$$

where N_{T_1} e N_{T_2} are the set of nodes of T_1 and T_2 and

$$\Delta(n_1, n_2) = \sum_{i=1}^{\mathcal{F}} \chi_i(n_1)\chi_i(n_2)$$

represents the number of common fragments rooted at nodes n_1 and n_2. The difference in the generation of fragments depend on the definition of $\Delta(n_1, n_2)$.

This tree kernel learning framework allow us to automatically extract features for input tree structures. Indeed, tree kernel functions measure the similarity between GeoTrees objects in terms of the number of common substructures. Depending on the type of kernel function, objects are mapped in different spaces. We make use of three types of tree kernels applied to GeoTree structures: STK (Section 2.4.1.1), STK_b (Section 2.4.1.1) and PTK (Section 2.4.1.2).

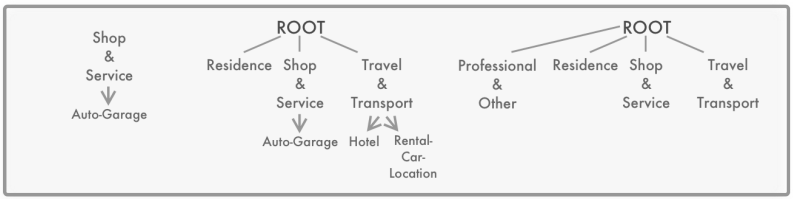


Figure 13: Some of the exponential fragment features from the tree of Figure 10 obtained by applying STK.

For example, PTK applied to GT of Figure 10 can generate informative fragments such as those in Figure 12, while by applying STK on the same tree we obtain different fragments as showed in Figure 13. The difference states in the production rules used by STK, which do not allow the separation of siblings. It turns out that patterns generate by PTK are more general and compact than those provided by STK.

4.2.1 *Hierarchical Point-Of-Interest Kernel (HPK)*

Tree kernels as STK and PTK are designed only to consider the textual label, e.g., word, associated to each node. However, depending on the data encoded in the tree structure, it might be necessary to weight the importance of each node. An example of such case and structure is given by the *Frequency GeoTree*, which is a tree structure that encodes the frequency, and more in general, the weight of POIs in a specific area. In order to learn from such weight in the learning process, from Equation 17 we define the Hierarchical Point-Of-Interest Kernel (HPK), a new structural kernel to compute the similarity between *Frequency GeoTrees*.

1. If n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \mu\sigma(n_1, n_2)$;
2. else:

$$\Delta(n_1, n_2) = \mu\lambda^2\sigma(n_1, n_2) + \mu \sum_{\mathbf{I}_1, \mathbf{I}_2, l(\mathbf{I}_1)=l(\mathbf{I}_2)} \lambda^{d(\mathbf{I}_1)+d(\mathbf{I}_2)} \prod_{j=1}^{l(\mathbf{I}_1)} \Delta(c_{n_1}(\mathbf{I}_{1j}), c_{n_2}(\mathbf{I}_{2j}))), \tag{22}$$

where, $\sigma(n_1, n_2) = w(n_1)w(n_2)$ is the function that take into account the node frequency, and $w(n)$ computes the weight of the node n. The other variables are the same of PTK (see Section 2.4.1.3).

For the urban zone u and the POI to the target node n, we define two main weighting schemes based on the frequency of each word category:

- The standard frequency TF:

$$w(n,u) = \begin{cases} f_{n,u}, & \text{number of times that POI is present in } u \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

- The smooth product $TF \times IDF$:

$$w(n,u) = log(1 + TF(n)) \times log(1 + \frac{N}{DF(n)}) \tag{24}$$

  where N is the total number of urban zones and $DF(n)$ is the number of those one containing the POI n.

With HPK, in addition to take into consideration that two urban zones share the same POI category, in computing their similarity we also consider POIs occurrences, which is an important factor in characterizing an urban zone. It should be noted that our HPK is a valid kernel as we substitute the matching function, which gives a contribution of $\mu\lambda$ or λ^2 with $\mu\lambda w(n_1)w(n_2)$ or $\lambda^2 w(n_1)w(n_2)$, respectively. Since the weighting function is just a product between two scalars it is a valid kernel. Thus, for the property of combination of valid convolution kernels [23], we obtain a valid convolution kernel.

4.2.2 *Combining HPKs and BOCs*

In a typical kernel machine, e.g., SVM, the input test object $\mathbf{x}$ is classified using the following prediction function: $h(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x_i})$, where α_i are the model parameters estimated from the training data, y_i represent the target variables, $\mathbf{x_i}$ are the support vectors, and $K(;)$ it is the kernel function that computes the similarity between two objects, i.e., urban zones. Although HPK includes all the features in BOC, a kernel combination with standard vectors can still be valuable. Thus, we combine HPKs with standard feature vectors, which do use frequencies and can produce a more expressive model. More in detail, given two grid cells, x_1 and x_2, we define the following new kernel combination:

$$K(x_1, x_2) = TK(T_1, T_2) + KV(\mathbf{v}_1, \mathbf{v}_2), \tag{25}$$

where TK is a kernel function applied to the two HRs, T_1 and T_2 extracted from x_1 and x_2 and KV is a kernel applied to the feature vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, also extracted from the cells above using any available data source (e.g. text, social media, mobile phone and census data). Although is out of the scope of our work, it is worth to notice that this feature vector can also include any other numerical feature we may have about the place (e.g. the popularity of a place [92] or the mobile phone activities [51]).

4.3 MINING GEOTREE PATTERNS

One of the interesting aspects of TKs is their ability to represent rich features sets, and thus alleviate the manual process of feature engineering. However, their feature space, being implicit, is extremely difficult to interpret, e.g., finding which subtree is more useful than others.

A viable solution consists in generating the kernel space, and then applying a feature selection to reduce the exponential number of subtrees to a manageable size. This process is based on two main assumptions: (i) it is possible to define subtree weights for efficiently mining the most useful substructures, and (ii) only a small subset of the exponential feature set is essential. Several works have successfully applied such methods for different kernel spaces [45, 44]. We follow the approach by [42] to mine substructures from the syntactic trees associated with semantic role labelling (SRL) [46]. The major challenges of using their approach are: (i) our GeoTrees are not syntactic but rather conceptual trees, thus the weighting approach may not work and need to be tuned for the task, (ii) syntactic trees are typically smaller than GeoTrees and they repeat most their subparts, e.g., NP (noun phrases), PP (propositional phrases), etc., whereas GeoTrees mostly do not contain repetitions.

4.3.1 *Reverse Kernel Engineering*

We apply the feature extraction algorithm for TK space, called *flink*[1], proposed by [42]. This enables the possibility of learning linear models on the explicit feature representation and using any other state-of-the-art machine learning algorithm based on such representation. Given a model learned with a kernel machine, e.g., SVM, the greedy mining algorithm of *flink* takes the support vectors and their weights in input to generate tree fragment weights, which

1 Code is freely available at http://danielepighin.net/cms/software/flink

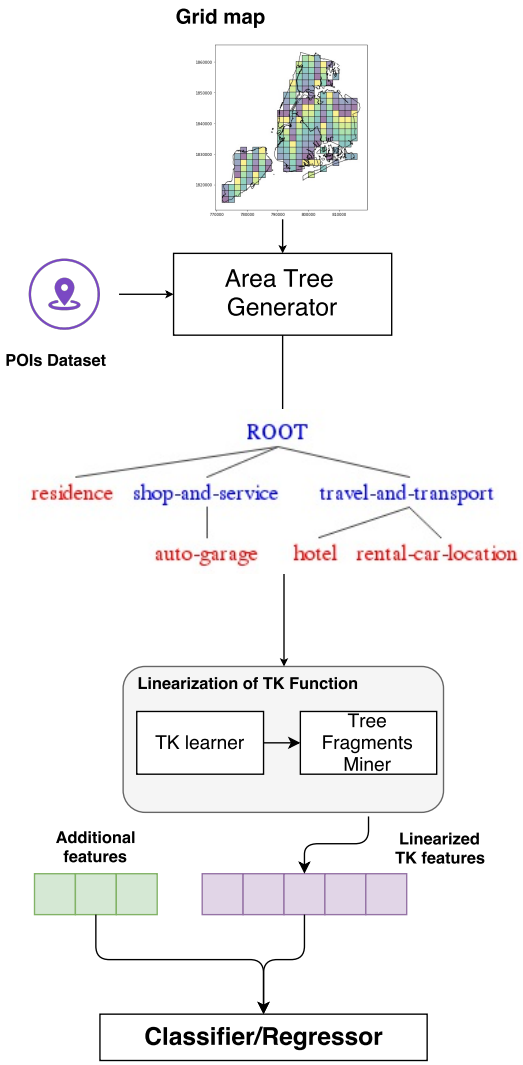


Figure 14: Generation process of structural feature vectors.

in turn can be used to select the most relevant fragments. An important concept to understand the idea of *flink* is the representation of a tree, which is a vector $\mathbf{x_i} = [x_i^{(1)}, ..., x_i^{(N)}]$, where each element corresponds to the number of occurrences of a given fragment, weighted with respect to the decay factor λ.

For a normalized STK, the value of the j-th attribute of the example $\mathbf{x_i}$ is:

$$x_i^{(j)} = \frac{T_{i,j}\lambda^{\frac{s(f_j)}{2}}}{\|\mathbf{x}_i\|} = \frac{T_{i,j}\lambda^{\frac{s(f_j)}{2}}}{\sqrt{\sum_{k=1}^{N} T_{i,k}^2 \lambda^{s(f_k)}}}, \tag{26}$$

where $T_{i,j}$ is the number of occurrences of the fragment f_j, associated with the j-th dimension of the feature space, in the tree T_i and $s(f_j)$ is the number of nodes that have at least a child in f_j, i.e., the *size* of the fragment. It follows that

the components of the model **w** , learned by SVM, can be written as:

$$w^{(j)} = \sum_{i=1}^{\ell} \alpha_i y_i x_i^{(j)} = \sum_{i=1}^{\ell} \frac{\alpha_i y_i T_{i,j} \lambda^{\frac{s(f_j)}{2}}}{\sqrt{\sum_{k=1}^{N} T_{i,k}^2 \lambda^{s(f_k)}}} \tag{27}$$

Once the weights are computed, the relevance is used to filter out those fragments having a relevance score smaller than the mining threshold $\sigma = B/L$, where B is the relevance of the heaviest fragment and L is a free parameter in input to *flink*. Once the most relevant fragments are selected, they are stored in an index together with their relevance score. This allows us to represent all the trees in the dataset as vectors in a linear space, where only a tiny percentage of the fragments of the original space are retained. The vectors contain 0/1 elements to indicate the presence or not of the corresponding fragment.

4.3.2 *Reverse Engineering of GeoTrees*

We propose a novel methodology to extract structural features for Urban Computing tasks, i.e., land use classification Figure 14. Starting from a grid map the *Area Tree Generator* module builds the GeoTrees, one for each grid cell, exploiting the information about POIs in the cell (see, e.g., the GeoTree in the figure). This way, we build a training set, where each example is represented by trees. Then we train our kernel-based machine using such data and the land labels associated with each cell. This gives us the support vectors and their weights, α_i, to fed the mining algorithm.

It should be noted that Equation 26 assigns a weight to the fragments using λ factor exponentiated to $s(f_j)$, which is the fragment size in terms of nodes. Since a GeoTree encodes many concepts of POIs, meaningful structures, i.e., patterns containing several concepts, are heavily down-weighted by the formula. Considering also that the frequency of the subtrees, $T_{i,j}$, (which can increase the fragment weight) is basically always one since subtrees do not repeat in a GeoTree, possibly meaningful structures receive very low weight. This prevents to extract important fragments with the standard *flink* setting as the application domains of the authors was rather different from ours, i.e., syntactic natural language trees for Semantic Role Labeling (SRL).

In particular, the constant B in our domain is typically associated with the most important POI, which, of course, can be extremely relevant, e.g., an Ap-

ple Store for the *Commercial and Office Buildings* category. This suggests that our B can be much larger than the value it gets in other tasks based on syntactic trees. A direct consequence is a very large threshold σ when standard values of the parameter L are used. These considerations led us to explore different weighting and mining parameters by also manually analyzing the mining results.

Finally, we used the mined trees as features to learn and classify with a linear SVM model. This also allows us to combine them with other additional features (e.g., BOC of Foursquare) and use structural features in other machine learning algorithms.

5

APPLICATIONS

5.1 LAND USE CLASSIFICATION AND GEO PATTERN ANALYSIS

In this section, we present the results of our two works on land use classification [2, 3], where the new structural kernels presented in Chapter 4 have been applied to automatically capture the most important patterns from the POI structure of city areas. These patterns are automatically engineered features that we exploit for the *classification* of the land use of urban zones.

5.1.1 *Overview*

The characterisation of urban forms to study key physical attributes and higher-level combinations of cultural and socio-economic layers becomes then of paramount relevance [76]. A traditional way to perform this characterisation, namely associating different areas with specific human activities (e.g. residential, industrial, business, etc.) is through the analysis of land use data. Such information is usually derived from airborne surveys and remotely sensed data, a costly and time-consuming endeavor. Given that these approaches to land use classification suffer from the same problems of census data (i.e. time and budget costs), methods based on novel sources of data have been proposed. An help in this direction is given by sensing technologies and Location-based Social Networks (LBSNs), e.g., Foursquare[1], which have increased the production and availability of geographic data.

Several works have focused on automatizing this task, i.e., *land use classification*, by automatically assigning the activity in a target urban area using mobile phone data (e.g. Call Detail Records) or LBSNs. These approaches represent a valid low cost alternative to the use of traditional data sources [54][59][60]. For example, [92] used the *check-in* information available in Foursquare to classify the land use in New York City (NYC). Foursquare is a well-known LBSN, which provides information about key social, cultural, and infrastructural components of an area by capturing its Point-Of-Interests (POIs), e.g.,

1 https://foursquare.com

Arts & Entertainment, *Nightlife Spot*, etc. Moreover, Foursquare allows users to register their presence in a place by means of a *check-in* action, which collects their latitude and longitude coordinates and additional metadata such as the check-in venue, comments and photos. In particular, [92] applied extensive feature engineering, mainly based on spatio-temporal data of check-in actions, and they built a Gold Standard from the data provided by the NYC Department of City Planning 2013, mapped on a grid of 200×200 meters that covers the whole NYC area. Other works deployed geo-data within machine learning models, creating feature vector representations of geographic entities, whereas vector features encode the information in each grid cell.

[58] used geo-tagged data from social networks (e.g., geo-located tweets) to infer the land use of an area. In particular, they followed a two-step approach: (i) first, they partitioned the geographical area by applying Self-Organizing Maps (SOMs), a neural network able to segment the land in areas considering the amount of tweets; (ii) then, they characterized each area by building a tweet-activity vector, which was used by a clustering algorithm to automatically identify the land use of the area. Recently, [36] proposed a deep learning approach on satellite imagery for land use classification. They applied convolutional neural networks to design land use classifiers for different cities in Europe. Their approach could not effectively learn subjective concepts of urban planning in complex urban environments. This suggests a possible synergy between this work and our approach, as they rely on images without taking in consideration the semantic about the place, e.g., presence of particular POIs.

Some other works have exploited the use of less conventional data (e.g., Call Detail Records (CDRs)) to infer the land use of an area [58, 51, 83]. Telco operators record mobile phone activities for billing purposes, and they usually store the time and the type (e.g., incoming calls, Internet, outgoing SMSs) of the communication, and the location of the radio base station handling the communication. For example, in [51] mobile phone data has been used to catch the time-variant relation between human mobility patterns and land use. In particular, they jointly used CDRs with a Random Forest classifier in order to classify the land use for the city of Boston. Then, for each area, the output of the classifier is compared with the one obtained for the neighboring regions and, if it is in disaccord with a certain number of neighbors, they change it by applying a sort of consensus validation.

We propose to represent urban zones in two different ways: (i) as a bag-of-concepts (BOC) extracted from the Foursquare description of the area, e.g., *Arts and Entertainment*, *College and University*, *Event*, *Food*; and (ii) as the

same concepts above organized in a hierarchical representation (HR), reflecting the hierarchical category structure of the LBSN activities. Then, (iii) we apply standard Tree Kernel (TK) functions [61, 88, 62, 63], to HR to project them in the space of all their possible substructures such that each space dimension corresponds to a given semantic structural feature. However, this approach do not take into account the importance of different nodes, this mainly because it has been designed for natural language processing tasks, e.g., question answering [129]. In contrast, we assume that different POIs can assume different relevance because they repeat with different frequency in different areas. Thus, we apply the Hierarchical POI Kernel (HPK) presented in Section 4.2.1.

We generate four different datasets for the following cities: Barcelona, Lisbon, Amsterdam and Milan, where the label to be analyzed is the land use, provided by the respective city administration. Finally, we use a Support Vector Machine (SVM) with Tree Kernels to project structures in the space of all their possible substructures such that each dimension corresponds to a given semantic structural feature. We make these substructures explicit by means of a mining algorithm, which extracts the most relevant features (tree fragments) from the implicit TK space according to the weights the kernel machine assigned to them. We evaluate this approach on land use classification for New York City. Despite few previous efforts [36], the absence of a standard benchmark still represents an obstacle when comparing different land use classifiers. To this end, we share both our code (pre-processing data and models) and the land use dataset to make comparisons with our model easier[2]. Finally, we analyze the nature of the different cities by applying a reverse kernel engineering algorithm, which extracts the most meaningful substructures from the HPK space. This automatically reveals some of well-known peculiarities of the cities above.

5.1.2 *Problem Definition*

Let $L = \{l_1, ..., l_m\}$ be the set of m labels indicating the land use category (e.g. Residential, Commercial & Office Buildings, and Industrial & Manufacturing), and let s be the polygon defined by q vertices $v_1, ..., v_q$.

2 https://github.com/gbarlacchi/hpk

Definition 5.1.1 *Given a shape s and a label $l \in L$, a land use zone is defined by the tuple $u = (s, l)$, containing the geographical area of the urban zone and the associated land use category.*

Definition 5.1.2 *We define the object city C as the disjoint set of land use zones $\{u_k\}_{k=1,..,n}$, where n is the total number of zones belonging to the same municipality.*

In our approach we rely on machine learning algorithms to perform a classification task, which requires a sampling to select the training samples among the possible zones of the city. Original land use datasets provide zones of different sizes, making not possible the random sampling of urban zones. A possible approach to uniform sample land use zones is to map the original datasets into a tessellation of fixed squared cells defined as in the following.

Definition 5.1.3 *Given a city C, we define the land use grid G as the tessellation covering $C_{shapes} = \bigcup_{k=1}^{n} s_k$ with $m \times n$ squared cells:*

$$G = \{\hat{u}_{ij}\}_{i=1,..,m;j=1,...,n} \tag{28}$$

where $\hat{u}_{ij} = (\hat{s}_{ij}, \hat{l}_{ij})$ is a tuple that models the cell (i, j), $\hat{s}_{ij}$ is a square defined by two vertices representing the top-left and bottom-right coordinates and $\hat{l}_{ij} \in L$ is the category associated to the $s_k \in C_{shapes}$ with the maximum overlapping area with $\hat{s}_{ij}$.

This definition reflects the approach proposed by [92], who assigned the predominant land use class to each grid cell.

Problem 5.1.1 *Given a land use grid G with squared cells of $k \times k$ meters, the problem of land use classification is to assign to each zone $\hat{u} \in G$ the correct land use category $\hat{l} \in L$.*

5.1.3 *Datasets*

Given the heterogeneity of currently available sources, the availability of comparable land use data is a notorious obstacle when reconstructing characterizations of urban environments across geographic domains. Land Use datasets are in fact traditionally provided by different levels of public authorities (e.g., national, regional and municipal), which undertake the costly and laborious tasks of surveying, processing and assembling such collections from a variety of sources. There are two main drawbacks stemming from such approach:

firstly, land cover data is hard to come by in a consistent fashion, often presenting profound discrepancies [39]. Secondly, land use data derived from official sources is often available only at the provided time intervals, which are understandably few and far in-between. For urban study purposes this would translate directly in a difficult comparison of diverse cities in terms of land use classes, let alone an observation of all the urban phenomena and changes over time that may occur. In the following we introduce the dataset with land use information and the POIs dataset built by using data from LBSN, which is a good source of information as it keep tracks of changes during the time.

Land use datasets are provided by the government or the municipality and usually differs from a city to another in terms of land use classes. In the following we introduce the two different dataset used in this work: (i) the MapPluto[3] dataset for NYC; and (ii) the Urban Atlas [4] dataset for four European capital, Barcelona, Milan, Amsterdam and Lisbon. Every city area associated with a land use class is described with the popular shape file format, where each shape is a collection of points geo-localized using their coordinates. The latter are usually provided with the well-known Coordinate Reference System (CRS) WGS84, adopted for the common latitude/longitude geolocation.

5.1.3.1 *MapPluto Dataset*

MapPluto is a freely available dataset provided by the the Department of City Planning (DCP) of NYC. It contains geo-referenced information, such as the precise category and shape of a building, for each city's borough. We use the shape file of New York provided by the NYC government [5]. This file is publicly available and contains the entire shape of New York divided in its five boroughs: Manhattan, Brooklyn, Staten Island, Bronx, and Queens. We focus on the following land use categories: (i) *One and Two Family Buildings*, (ii) *Multi-Family Walk-Up Buildings*, (iii) *Multi-Family Elevator Buildings*, (iv) *Mixed Residential and Commercial Buildings*, (v) *Commercial and Office Buildings*, (vi) *Industrial and Manufacturing Buildings*, (vii) *Transportation and Utility*, (viii) *Public Facilities and Institutions*, (ix) *Open Space and Outdoor Recreation*, (x) *Parking Facilities*, and (xi) *Vacant Land*. To overcome the problems related to the fine-grain information about the land use, we merged the classes (i) One & Two Family Buildings, (ii) Multi-Family Walk-Up Buildings and (iii) Multi-Family Elevator Buildings into a single general *Residential* category. Then, we

3 https://www1.nyc.gov/site/planning/data-maps/open-data.page#pluto
4 https://www.eea.europa.eu/data-and-maps/data/urban-atlas
5 http://www1.nyc.gov/site/planning/data-maps/open-data/districts-download-metadata.page

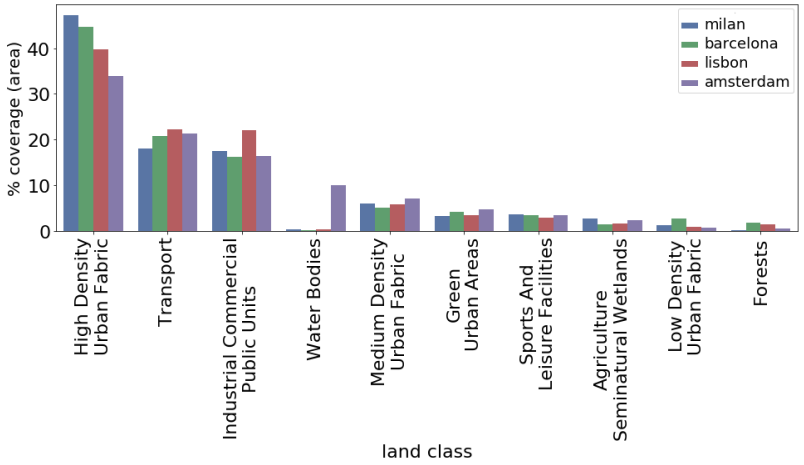


Figure 15: Distribution of coverage by area for the Urban Atlas land use classes in Barcelona, Milan, Amsterdam and Lisbon.

also aggregated (i) Industrial & Manufacturing, (ii) Public Facilities & Institutions, (iii) Parking Facilities and (iv) Vacant Land into a new category called *Other*. Compared to the original categorization, this new taxonomy has a lower granularity, thus facilitating the identification of the predominant class in each cell.

5.1.3.2 *Urban Atlas Dataset*

Urban Atlas is an open-source dataset that is part of the EU Open Data Portal[6]. It contains land use data, standardised in 20 land use classes, covering Large Urban Zones (formally defined as cities of 100, 000 inhabitants or more). It is a long-standing initiative of the European Commission-JRC and the European Environmental Agency, and its stability combined with standard quality makes it a go-to data source for researchers and analysts.

The data for each city is provided in the widely adopted ESRI Shapefile format, the *de facto* standard for GIS analysis and geo-spatial data exchange. Every record in the dataset contains a polygonal shape composed by a sequence of Latitude/Longitude points and the associated land use class. The association of the geometry with an attribute class identifies a patch of land. Following the provided metadata[7] we identified and aggregated a set of relevant classes of interest. By this we intend those classes related to anthropocentric activities in general, and to urban dimension in particular, with the ability to abstract from

6 http://data.europa.eu/

7 https://land.copernicus.eu/user-corner/technical-library/urban-atlas-2012-mapping-guide-new

land use those functions which are distinguishing traits of urban life and can thus be good proxies for the characterisation of urban environments.

Similarly to the NYC case, we selected a sub-group of classes of interested. First, we aggregated some classes and we focused on those related to city centers, discarding those less interesting from a social viewpoint, i.e., forests, agricultural, semi-natural and wetland areas, and mineral extraction and dump sites. Then, we collapsed *Medium* and *Low Density Urban Fabric* into one single category, *ML-Density Urban Fabric* as they only have few samples. The six classes of interest we considered are: (i) *High Density Urban Fabric*, (ii) *Medium Density Urban Fabric*, (iii) *Low Density Urban Fabric*, (iv) *Industrial, commercial, public, military and private units*, (v) *Open Space & Recreation*, (vi) *Transportation*. Figure 15 shows the land use distribution in the four different cities, i.e., Barcelona, Milan, Amsterdam and Lisbon.

5.1.3.3 *Foursquare's Point of Interests*

We chose to use the Foursquare API[8] as a source of Points of Interest. It allows for 100,000 API requests per day, free of charge. A POI is usually characterized by a location (i.e., latitude and longitude), textual information (e.g., a description of the activity in that place) and a hierarchical categorization that provides different levels of detail about the activity of the place (e.g., *Food*, *Asian Restaurant*, *Chinese Restaurant*). We used POIs extracted from Foursquare, a geolocation-based social network supported with web search facilities for places and a recommendation system. We consider all the textual categories associated with each POI. There are in total ten different macro-cotegories, each one specialized in maximum four levels of detail. These levels follow a hierarchical structure[9], where each level of a category has a number of subcategories.

5.1.4 *Experimental Setup*

Our experiments aim at demonstrating the effectiveness of our models on the Urban Atlas dataset composed by four European cities geographically and socially very different: Amsterdam (Netherlands), Barcelona (Spain), Lisbon (Portugal) and Milan (Italy). Additionally, we compare TK models, which implicitly use structural features, with models explicitly using those extracted. In

8 https://developer.foursquare.com/

9 https://developer.foursquare.com/categorytree

particular, we apply a reverse kernel engineering approach to analyze the most important structural patterns automatically selected by our algorithms. In order to show the generality of our approach, we test the Structural Features (SF) on an additional land use dataset of New York City.

We implemented HPK as an additional kernel in Kelp [41], a java machine learning framework focusing on kernel machines for structured data. For building the grids and manage all the geo-spatial components of the datasets we implemented PyGeol, a pre-processing library for geographic data. We released and made freely available both the implementation of our HPK[10] and PyGeol[11] library.

Our experiments, as presented in Sec. 5.1.3, are based on the combination of two urban environments datasets: (i) Foursquare POIs and (ii) Land Use datasets. The latter provides a very fine-grained information, e.g., in most cases, there is a label assigned to just one building. Such level of detail can be hardly captured with POI information. Thus, a reasonable trade-off between classification accuracy and the desired area granularity consists in segmenting the regions in squared cells as previously mentioned. One drawback is that each cell may include more than one land use. We resolve this inconsistency following the same approach by [92, 36], who assigned the predominant land use class to each grid cell. We removed those cells that are not representative of the associated land use classe, i.e., cells where the coverage of the predominant class is less than 25% of the total coverage.

We evaluated our approaches to *land use classification* as a multi-class classification problem, where the objective is to assign the correct land use label (e.g., High Density Urban Areas) to each cell. As every multi-class problem, it can also be decomposed into a series of binary problems. Due to the unbalanced nature of our data set, we use well-known metrics for assessing the accuracy of classification systems: (i) Accuracy (Acc.), (ii) Precision, (iii) Recall and (iv) F1-score. Precision is defined as the ratio $\frac{tp}{tp+fp}$, where tp is the number of true positives and fp is the number of false positives, while Recall is defined as the ratio $\frac{tp}{tp+fn}$, where tp is the number of true positives and fn is the number of false negatives, which are samples erroneously not labelled as belonging to the positive class. As we are dealing with a multi-class problem, we consider the average performance of each individual class, namely Macro-Precision, Macro-Recall and Macro-F1-score. A macro-average treat the classes equally by computing the average of the metric independently for

10 https://github.com/gbarlacchi/hpk
11 https://github.com/PyGeoL

each class. The micro-average aggregates the contributions of all classes to compute the average metric. For this reason, in a multi-class classification setup with an imbalance dataset, the micro-average is a preferable choice In a multi-class classification setup, micro-average is preferable if you suspect there might be class imbalance.

5.1.5 *Land Use Classification Results*

We performed the analysis on four European cities, which are geographically and socially very different: Amsterdam (Netherlands), Barcelona (Spain), Lisbon (Portugal) and Milan (Italy). As shown in Table 3, each city presents a different distribution, both regarding data availability and land area size (number of cells). We labelled the dataset following the classes definition from Urban Atlas, as discussed in Section 5.1.3. In Table 3, we report the number of POIs and cells present in the datasets of each city. We experimented with different grid dimensions across all cities e.g., 50 × 50, 100 × 100, 200 × 200 and 250 × 250 meters. For each grid, we created training, validation and test sets, randomly sampling 60%, 20%, 20% of the cells, respectively. We found that depending on the city, the best performance is obtained with different cell size. This can be intuitively explained considering the remarkable diversity of urban morphologies of the different cities, e.g., the size variability of city blocks, the street network configuration and the overall land use patterns.

In Figure 16, we show the average F1 over the target categories, i.e., Macro-F1 score, obtained with the models trained for each city at different spatial scales and tested on the validation set. Due to space constraints, in the following, we only report our experiments on the grid with 200 × 200 cell size, which represents a reasonable trade-off between classification accuracy and grid granularity. Moreover, this setting is also in line with similar works [92, 51, 2, 3].

City	Cells	POIs
Barcelona	2963	44891
Milan	5258	43526
Amsterdam	5697	35362
Lisbon	2321	19093

Table 3: The number of cells (200 × 200 meters) and POIs availability on the four different cities.

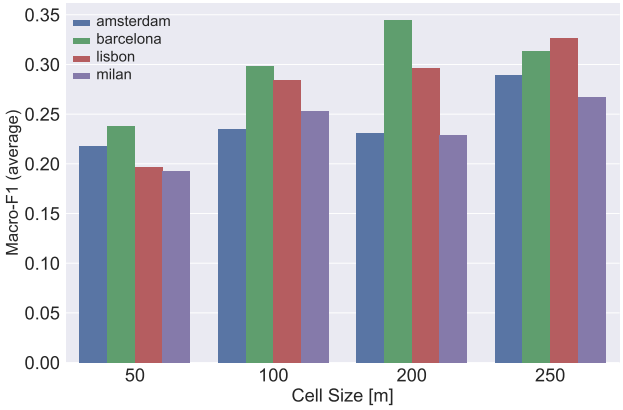


Figure 16: Macro-F1 averaged among the models reported in Table 11 according to a different grid size.

We compared HPK against an extensive set of different land use classification models:

- SVM with linear (Lin), polynomial (Poly) and radial basis function (Rbf) kernels applied to BOC;
- A non-kernel based model such as *XGBoost* [48];
- A dummy classifier (baseline) obtained by always classifying an example with the most frequent label (*Forests*);
- State-of-the-art models based on GeoTree applied to TK [2].

We tuned the models parameters on a validation set by applying a grid search approach to maximize the Macro-F1 score on the development set. HPK involves the following parameters: (i) the decay factors μ and λ for TK, (ii) C value for all the SVM approaches, and the specific parameters for the combined kernel, i.e., degree in *poly* and γ in RBF kernel. In XGBoost, we tuned the most important parameters such as the maximum depth of the tree and the minimum sum of weights of all observations required in a child node. For each city, we obtained a set of different parameters from the development set, which we used to train and test our final models.

5.1.5.1 *Comparing Standard vs. Structural Models*

Table 11 shows the results in terms of Accuracy (Acc.), Macro-Precision (MP), Macro-Recall (MR) and Macro-F1(MF1).

City	Model	Acc.	MP	MR	MF1
Barcelona	**HPK$_{TF\times IDF}$+Rbf**	0.623	0.439	0.357	**0.394**
	HPK$_{TF}$+Poly	0.597	0.445	0.344	0.388
	HPK$_{TF}$	0.607	0.445	0.341	0.386
	TK (PTK)	0.621	0.435	0.330	0.375
	HPK$_{TF}$+Lin	0.607	0.419	0.338	0.374
	Poly	0.482	0.425	0.235	0.303
	Rbf	0.543	0.631	0.186	0.287
	XGBoost	0.624	0.243	0.242	0.236
	Lin	0.382	0.259	0.178	0.211
	Baseline	0.545	0.061	0.111	0.078
Lisbon	**HPK$_{TF}$+Poly**	0.472	0.409	0.253	**0.313**
	HPK$_{TF\times IDF}$+Lin	0.468	0.307	0.267	0.286
	HPK$_{TF\times IDF}$+Rbf	0.470	0.319	0.257	0.284
	TK (PTK+Rbf)	0.555	0.32	0.443	0.25
	HPK$_{TF\times IDF}$	0.459	0.292	0.233	0.259
	Poly	0.369	0.488	0.214	0.298
	Rbf	0.442	0.455	0.175	0.253
	Lin	0.366	0.489	0.236	0.219
	XGBoost	0.460	0.170	0.142	0.129
	Baseline	0.489	0.049	0.100	0.066
Amsterdam	**HPK$_{TF}$+Poly**	0.411	0.264	0.300	**0.281**
	TK (PTK)	0.399	0.266	0.289	0.277
	HPK$_{TF\times IDF}$	0.359	0.278	0.270	0.274
	HPK$_{TF}$+Lin	0.413	0.260	0.275	0.268
	HPK$_{TF}$+Rbf	0.414	0.255	0.279	0.266
	Lin	0.275	0.190	0.241	0.212
	Poly	0.319	0.191	0.184	0.187
	Rbf	0.322	0.166	0.151	0.158
	XGBoost	0.534	0.166	0.138	0.134
	Baseline	0.433	0.043	0.100	0.060
Milan	**HPK$_{TF\times IDF}$+Poly**	0.598	0.271	0.562	**0.179**
	HPK$_{TF\times IDF}$	0.584	0.248	0.542	0.160
	TK (PTK)	0.575	0.246	0.541	0.156
	HPK$_{TF}$+Lin	0.573	0.245	0.525	0.161
	HPK$_{TF}$+Rbf	0.564	0.245	0.502	0.162
	Rbf	0.519	0.210	0.526	0.131
	XGBoost	0.565	0.105	0.167	0.122
	Lin	0.542	0.208	0.736	0.121
	Poly	0.550	0.173	0.411	0.109
	Baseline	0.563	0.080	0.062	0.111

Table 4: Results on land use classification across different cities.

The TK are the models from [2]. In the brackets we report the used tree kernel function (e.g., PTK, STK and HPK) and, eventually, the kernel applied to

feature vector BOC (e.g., PTK+Rbf). For HPK models we indicate the weighting schema in subscript. In particular, TF means that term frequency is used in the HPK weight, while $TF \times IDF$ indicates the homonymous weighting scheme as introduced in Section 4.2.1. For instance, HPK_{TF} means that the TF schema is applied to HPK. Moreover, we combined kernels with a simple summation, e.g., HPK+Rbf indicates an SVM using such kernel combination. We report all the models performance respect to the F1-Score and accuracy in Table 11. The table shows that our approach improves the overall accuracy and F1-score offering the possibility of a deeper semantic interpretation of an area, which goes beyond the simple use of Foursquare classes or check-in and avoid the feature engineering problem. We noted that:

- the MF1 of XGboost and SVM using BOC features is very poor. This might be due to the intrinsic difficult of the problem, that require a rich feature representation to correctly differentiate among the classes. The XGboost accuracy is typically very good but this is at the expenses of the other categories, which results in a trivial classifier mainly outputting the majority class;
- the combination of HPK with BOC significantly improves the performance. On the Barcelona dataset, the improvement is up to 30% points (relative) in F1-score;
- HPK models always outperform the other baselines, with high improvement in MF1. It also improves state-of-the-art kernels, such as TK (PTK) by 1.5% absolute.

These results clearly show that is possible to define a hierarchical POI representation able to capture structural concepts of urban areas since they show that HR together with HPK is an effective hierarchical representation, which highly improves land classification. Additionally, we have shown that HPK improves the state of the art. It should be noted that the overall performance may not look as much good as in other similar works using different datasets. This is mainly due to the different characteristics of the cities we have experimented with, which are in general more difficult to analyse than New York, for example. Once again this suggests potential synergy among different approaches, e.g., satellite imagery and LBSN data.

5.1.6 *Mining Geo Pattern Trees*

In these experiments, we evaluate implicit and explicit structural features on land use classification by using Geographical Tree structures as defined in Section 4.1.2. We choose New York City as: (i) it is a big size city. It helps us to demonstrate the generalization ability of our model even in different settings, i.e., big-size cities vs. medium-size cities; (ii) NYC contains diverse urban functions, (iii) it has a highly active economy.

We first test TK models on Manhattan using several grid sizes focusing on evaluating the best models on all NYC boroughs [2]. Then, we use the best models on the entire NYC, also enabling comparisons with previous work. Finally, among the all land use classes available in the dataset, we compare the best TK models with the linearized feature vectors on two classes: *Residential* and *Commercial*.

Similarly as in Sec. 5.1.5, we trained multi-class classifiers using common learning algorithm such as XGboost [48], and SVM using linear, polynomial and radial basis function kernel, named SVM-{Lin, Poly, Rbf}, respectively, and our structural semantic models, indicated with STK, STK_b and PTK. We also combined kernels with a simple summation, e.g., PTK+Poly indicates an SVM using such kernel combination.

We first tested our models individually just on Manhattan using different grid sizes. Figures 17 and 18 show the accuracy of the multi-classifier for different models according to different granularity of the sampling grid. We note that SVM-Poly, XGboost and LogReg show comparable accuracy. PTK and STK_b perform a little bit less than the feature vector models. Interestingly, the kernel combinations in Fig. 19 provide the best results. This is an interesting finding as XGboost is acknowledged to be the state of the art for land use classification. Additionally, when the size of the grid cell becomes larger, the accuracy of TKs degrades faster than the one of kernels based on feature vectors. This mainly because the conceptual tree becomes too large.

After the preliminary experiments above, we selected the most accurate models on Manhattan and tested them on the other boroughs of NYC. Tab. 5 shows that TKs are more accurate than vectors-based models and the combinations further improve both models.

Area	Cell	XGBoost	SVM-poly	PTK	PTK+poly	STK	STK+poly	STK_b	STK_b+poly
Manhattan	50	45.0	39.9	47.6	48.0	45.0	47.6	47.4	**48.6**
	100	54.0	54.4	53.9	55.5	48.1	55.0	53.1	**55.5**
	200	63.0	64.4	61.3	66.1	50.4	65.4	62.1	**65.9**
	250	57.0	63.2	54.6	61.8	39.6	**63.9**	56.1	63.2
Bronx	50	43.0	30.9	44.9	44.9	42.2	**43.4**	42.4	43.2
	100	50.0	43.7	53.2	54.1	51.2	53.2	**54.7**	54.0
	200	59.0	56.4	62.6	60.6	56.4	60.4	61.8	**61.8**
	250	59.0	58.6	63.5	64.9	59.3	59.6	63.0	**65.2**
Brooklyn	50	49.0	44.2	51.3	51.6	48.7	51.3	51.4	**52.2**
	100	61.0	61.0	63.1	**63.5**	62.4	62.9	63.1	63.2
	200	71.0	71.5	72.9	73.6	70.1	73.2	73.3	**73.8**
	250	70.0	68.9	71.3	**72.6**	67.9	70.3	70.6	71.4
Queens	50	48.0	32.4	51.5	**51.5**	50.2	51.0	50.5	50.3
	100	58.0	57.2	61.4	61.3	59.8	60.6	61.6	**61.7**
	200	67.0	66.5	70.5	**71.3**	69.3	69.9	70.4	71.0
	250	68.0	68.3	72.9	73.1	70.1	72.2	72.4	**73.6**
StatenIsland	50	51.0	38.63	54.4	**55.2**	52.8	54.6	53.8	54.9
	100	57.0	56.73	58.1	**58.7**	53.6	57.4	56.0	58.1
	200	60.0	60.0	**61.8**	61.1	60.2	60.0	61.3	60.9
	250	66.0	64.87	67.4	66.3	66.0	67.2	**67.9**	67.4

Table 5: Accuracy of the best models for each New York borough and cell size.

In the final experiments, we tested our best models on the entire NYC with a grid of 200×200. We applied the same tuning procedure and evaluation metrics as in Section 5.1.5.

Table 6 shows the results in terms of Accuracy, Macro F1, Macro-Precision and Macro-Recall. The model *baseline* is obtained by always classifying an example with the label *Residential*, which is the most frequent.

We note that: (i) all the feature vector and TK combinations show high accuracy, demonstrating the superiority of TK over all the other models. (ii) STK_b+poly (polynomial kernel of degree 2) achieved the highest accuracy, improving over XGBoost up to 4.2 and 6.5 absolute percent points in accuracy and F1, respectively: these correspond to an improvement up to 18% of the state-of-the-art model.

Finally, Zhan et al. [92] is the result obtained on the same dataset using check-in data from Foursquare. Although an exact comparison cannot be carried out for possible differences in the experiment setting (e.g., Foursquare data changing over time), we note that our model is 1.8 absolute percentage

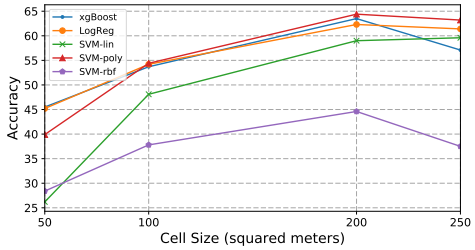


Figure 17: Common machine learning models on different cell sizes in Manhattan.

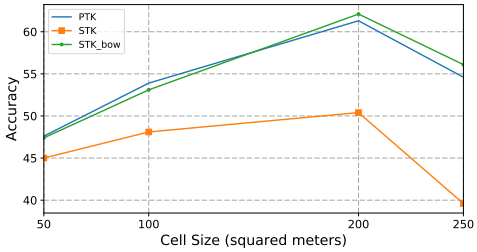


Figure 18: TK models on different cell sizes in Manhattan.

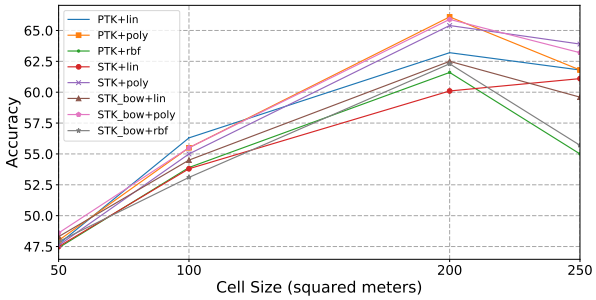


Figure 19: Accuracy of kernel combinations using BOC vectors and TK on GeoTrees according to different cell sizes of Manhattan.

points better.

Using the same configuration with the gird size of 200×200 and the best TK model, in Table 7 we report the binary classification performance for the *Residential* and *Commercial* classes. We follow the previous defined notation SMV-{Lin, Poly, Rbf}, and we experimented with (i) count features (BOC), (ii) structural features (SF) and their combination (SF+BOC). In order to show that linearization is able to yield unaffected accuracy of structural kernels, we compare those models with the best TK model both with and without BOC.

Regarding *Residential* class results, we note that: (i) TK+Lin(BOC) [12] provides the best performance both in Accuracy and F1-score. This confirms our finding from Section 5.1.5.1, where combination of TK and BOC largely outperformed state-of-the-art models. (ii) The combination SF with BOC always improves the performance. In particular, Lin(SF, BOC) improves Lin(BOC) and

12 Although in the previous section, we showed that TK+Poly(BOC) provides the best accuracy, we focus on linear models in these experiments since they are the fastest for the training and test phase.

Model	Acc.	F1	Prec.	Rec.
baseline	58.9	12.4	0.98	16.6
XGBoost	63.2	36.1	57.9	31.9
SVM-poly	62.1	27.4	51.3	25.9
STK_*b*+poly	**67.4**	**42.6**	**63.9**	**37.4**
PTK+poly	66.9	41.4	63.8	36.2
STK_*b*	66.6	38.1	52.8	33.9
PTK	65.9	37.2	58.7	33.0
STK+poly	65.5	37.3	54.5	33.3
STK	62.7	25.9	41.5	24.7
Zhan et al.	65.6	-	-	-

Table 6: Classification results on New York City.

XGB(BOC) by about 3.5 and 1.7 absolute percentage points in F1-score, respectively. (iii) XGB(SF, BOC) improves XGB(BOC) by 1.9% absolute demonstrating the general validity of SF for different learning algorithms. Regarding the *Commercial* class, its highly imbalance in terms of positive and negative examples produces results much worse than for the *Residential* class. Additionally, we did not tune the cost-factor parameter of the models to balance Precision and Recall (this explains null outcome for some models). However, even in these non-optimal conditions SF always improves all the models.

We further show that SF are equivalent to TK as they perform similarly, 0.692 vs 0.702 on the Residential class. Again, Lin(SF, BOC) performs similarly to TK+Lin(BOC), i.e., 0.709 vs. 0.712, suggesting that SF can replace TK also in combinations. Finally, the above results also shows as SF are highly scalable and effectively usable in other learning algorithms. In the next section, we show that they are a relatively small number, thus confirming their high scalability.

5.1.6.1 *Impact of Structural Features*

In these experiments, we study the quality of SF. Figure 20 shows the F1-score of different models according to the increase of fragment numbers. We note that: first the F1-score stabilizes around 4,000 fragments, reaching the value of implicit TK models. For example, the best model in this experiment is TK+Poly(BOC). We can see that Lin(BOC+SF) reaches its same accuracy with only 4, 000 fea-

Area	Model	Acc.	F1	Prec.	Rec.
Residential	Lin(BOC)	0.624	0.746	0.620	0.934
	Lin(SF)	0.692	0.764	0.697	0.845
	Lin(SF, BOC)	0.709	0.781	0.701	0.882
	Poly(BOC)	0.653	0.762	0.639	0.942
	RBF(BOC)	0.589	0.742	0.589	1.000
	TK	0.702	0.774	0.700	0.867
	TK+Lin(BOC)	0.712	0.783	0.705	0.879
	XGB(BOC)	0.671	0.764	0.662	0.903
	XGB(SF)	0.655	0.763	0.641	0.944
	XGB(SF, BOC)	0.696	0.783	0.675	0.933
Commercial	Lin(BOC)	0.956	0.000	0.000	0.000
	Lin(SF)	0.957	0.158	0.593	0.091
	Lin(SF, BOC)	0.963	0.288	0.909	0.171
	Poly(BOC)	0.965	0.352	0.927	0.217
	RBF(BOC)	0.658	0.162	0.091	0.749
	TK	0.956	0.294	0.92	0.175
	TK+Lin(BOC)	0.964	0.343	0.902	0.211
	XGB(BOC)	0.966	0.412	0.887	0.269
	XGB(SF)	0.956	0.044	0.571	0.023
	XGB(SF, BOC)	0.967	0.421	0.906	0.274

Table 7: Binary classification for the most common land use classes in NYC.

tures. This suggests that the mining approach is rather effective as there is almost no loss in accuracy after linearization: only a small subset of structural features plays a very discriminative role in the classification task. Figure 21 confirms the outcome above reporting the F1 plot of TK+Poly(BOC) obtained on the development set for the class residential. The first ten features are taken from BOC and after, the most relevant fragments from SF are added to the features space. We can see that the first 1,000 fragments, which generally correspond to small POI structures or individual POIs do not improve the accuracy as they are already in BOC. However, after many important complex patterns

are introduced, the F1 rapidly increases, reaching a plateau around 2000 features, as noted before.

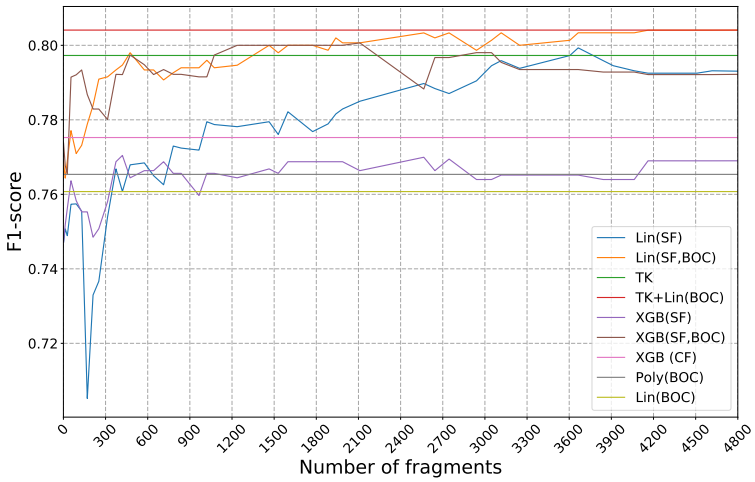


Figure 20: F1-score of several models increasing the structural fragments they use.

5.1.6.2 *Qualitative Analysis of Structural Features*

We sorted the fragments extracted for each class according to the relevance score, and we manually analyzed the top positive and the top negative fragments identified for three interesting classes: (i) *Residential*, (ii) *Commercial* and (iii) *Mixed*. To the best of our knowledge, this is the first work where such approaches are used for geo and location data analysis within machine learning models. As expected, the presence of condos and other residential buildings is very relevant for the residential class, e.g., see Figure 23. Interestingly, a set of concepts as *food*, *outdoors-recreation* and *shop-service* is another relevant features. Note that a subset of such concepts would not characterize well the residential area as it may be associated with *Commercial and Office Buildings*. In particular, the presence of government buildings, offices, and other professional places characterize the *Commercial* category, e.g., the subtree in Figure 23, which is of course a negative example of the *Residential* class. The strength of SF relies on the assumption that there are specific sub-hierarchies of POIs that characterize a given urban area. Our analysis of the more relevant fragments, together with their relevance and presence over the dataset, supports such hypothesis. One test we did was to eliminate one or more nodes from highly relevant fragments: this resulted in a drastic change of their relevance score. For example, we found [*food*[*deli bodega*][*dessert shop*[*ice cream shop*]]] to be one highly relevant positive fragment of the *Residential* class. If

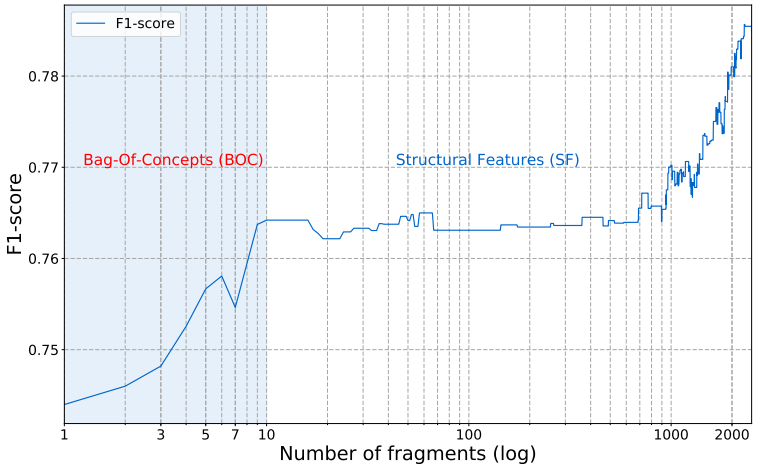


Figure 21: F1-Score on the class Residential adding features step by step.

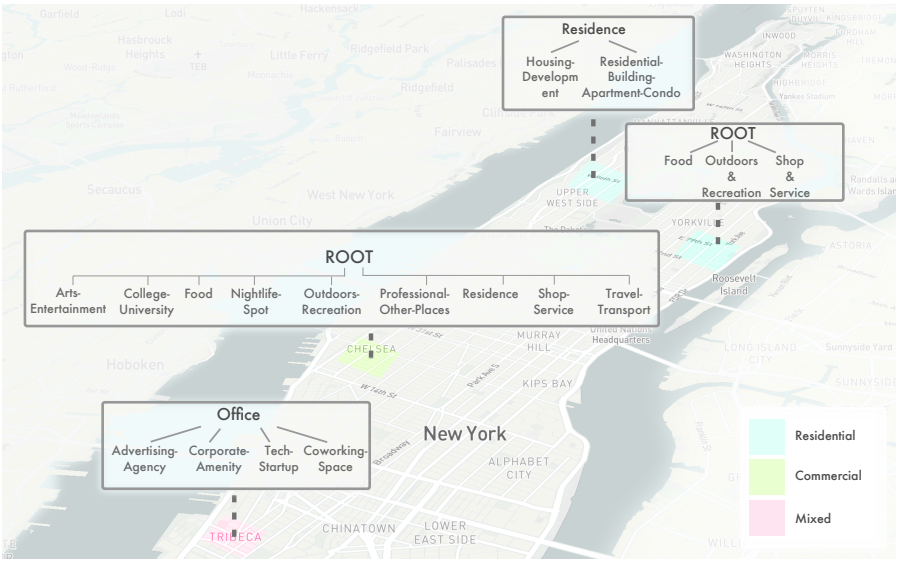


Figure 22: Example of relevant fragments in the class Residential, Commercial and Mixed.

we remove the sub-fragment [dessert shop[ice cream shop]] from it, the resulting feature, [food[deli bodega], completely changes its role. Indeed, it can be matched in different classes losing its ability to characterize the Residential class.

We extract and sort resulting fragments for each class according to their respective relevance score. We report in Figure 23 some of the occurrences found for the classes (i) *High Density Urban Fabric (HD* and (ii) *Sport and Leisure Facilities (SP)*. Analyzing the fragments we can note that similar urban typologies (e.g., *[apartment condos]*) appear together in a strong combination or that the presence of *[residential buildings]* is associated with the class High

Figure 23: Example of relevant fragments in the classes High Density Urban Fabric and Sport and Leisure facilities.

Density Urban Fabric. This is expected as such descriptors pertain to urban functions commonly found in denser areas of a city.

Similarly to NYC, we show a qualitative analysis also for Amsterdam. Figure 23 illustrates the case of Amsterdam, where the presence of the local Indonesian community is highlighted by the *[asian restaurants[indonesian restaurants]]* fragments. This prevalence of hyper-local descriptors is common across classes, as HPK is able to identify Amsterdam *[athletics_sports[skate_rinks]]* for the class *Sport and Leisure Facilites* as well.

Finally, we stress the fact that many concept fragments found in the structural features are very difficult, if not impossible, to manually design, making HPK a suitable tool to alleviate the process of feature engineering and building effective urban zones representations.

5.2 PREDICTING INFLUENZA-LIKE SYMPTOMS USING HUMAN MOBILITY BEHAVIORS

This section is based on our work presented in [6] where we investigate the role of human mobility for predicting the physical health conditions of a person.

5.2.1 *Overview*

Knowing in advance if someone will present certain symptoms may have significant implications in terms of public health strategy and policy. Previous studies have demonstrated the association between the health and behavioral patterns of a person, and the possibility to predict health and well-being conditions using different sources of behavioral information from social media and mobile phones. Detection of emotional states, happiness levels and depressive disorders [120, 121, 134, 126], prediction of physical health conditions [123, 125] and stress levels [84], and modeling of influenza spreading [85, 122, 87, 119, 86] are some common examples of the studies carried out in this area. Interestingly, a recent work has shown that human mobility represents a good proxy for predicting people's mental health conditions such as depressive states [126].

In this paper, we present an initial study to investigate the effectiveness of using individual mobility behaviors for predicting the health conditions of a person. We address the challenging problem of predicting future presence of physical health symptoms such as *fever*, *sore throat*, *cough*, *shortness of breath*, *headache*, *muscle pain*, *malaise*, and *cold* by exploring the past mobility activities of an individual, thus trying to answer the following question: *can mobility behaviors be informative regarding the future health conditions of a person?*

To address this problem, we resort to the data collected during the Mobile Territorial Lab (MTL) study [124], a longitudinal living lab that has been observing the lives of more than 100 parents through multiple data sources (e.g. mobile phone data, questionnaires, experience sampling probes, etc.) for more than two years. Then, we extract a set of daily features capturing the spatio-temporal mobility patterns of a person (e.g. total distance traveled, radius of gyration of visited places, maximum displacement from home, unique number of visited places, etc.). For each individual we analyze how the mobility metrics and the presence of symptoms correlate and change over time. We also design a machine learning framework that, using past mobility behaviors, predicts the presence of flu-like and cold symptoms with a time horizon of two days ahead. To evaluate our machine learning framework, we firstly run experiments using a feature selection step (Recursive Feature Elimination (RFE) [136]). In order to select the more predictive features, we fit one of the regression models and then we rank the features (i.e. total distance) by their weight in the model. Then, once we have a comprehensive analysis of the participant's

mobility features, we use them to predict if s/he will present certain symptoms in the next days (e.g. two days ahead).

Our results show that using the mobility patterns of an individual we can obtain promising performance for our challenging prediction task. Specifically, we obtain an Area Under the Receiving Operating Characteristic Curve (AUCROC) of 0.57, a Precision score of 0.72, a Recall score of 0.84, and F1-score of 0.77 in classifying symptoms two days ahead with a Random Forest (RF) classifier.

5.2.2 *Data*

In this work we use a data set collected during the Mobile Territorial Lab (MTL) study (for a more detailed description of the study see [124]).

During the MTL study, the researchers have observed the lives of more than 100 parents for almost three years (January 2013 - December 2015). The participants live in the province of Trento, an area located in the Northern of Italy, and most of them are of Italian nationality. They have different levels of education (from high school diplomas to Ph.D. degrees) and types of occupation. Participants were provided with (i) an Android-based smartphone running a software able to continuously collect different mobile phone data (e.g. calls, SMSs, locations) and (ii) a survey application which is able to periodically ask the participants some questions designed by the researchers in the context of a specific study [124]. Following the Italian regulations, all participants were asked to sign informed consent forms and the study was conducted in accordance to them. The form and the MTL study were also approved by a joint Ethical Committee of University of Trento and Province of Trento.

In this paper we report a study on health symptoms that we conducted on 70 participants, 20 males and 50 females, with an age ranged from 28 to 46 (the study was run during the first phase of the MTL project when only 70 study participants were enrolled). In particular, we use a combination of two type of data: (i) location data, which we use to characterize the daily mobility of the participant; and (ii) survey data with daily information about the health of the participant, which represents the ground truth of our supervised machine learning models. The data set is completely anonymized in order to ensure individuals' privacy.

We collect symptoms data from February 20, 2013 and March 21, 2013 since in this period we have a high presence of flu-like and cold symptoms. This is

also in line with the epidemic curve of the 2012-2013 influenza season, which presents a peak during our window of time [138]. In particular, we focus only on collecting one month of symptoms data in order to have a high participation rate from our study participants.It is worth specifying that symptoms and mobility data sets do not completely overlap. This is due to the fact that there are some gaps in (i) the mobility data (i.e. participants switched off the mobile phones) and (ii) the survey data (i.e. participants did not fill the health symptoms' survey). Hence, we have mobility data and at least one self-reported symptom for only 60 study participants.

5.2.2.1 *Location Data*

The software installed on the smartphone continuously keeps track of: (i) the communication events (e.g. calls and SMSs), and (ii) the participant's location captured by means of the Global Positioning System (GPS), which recorded 82% of positions with an accuracy within 20 meters [124]. In addition, to increase the number of location points we also use the position retrieved by the network provider source (i.e. the cell towers to which the phone is connected). The raw location data set consists of location point tuples $l = [ID, latitude, longitude, source, accuracy, time]$, where for each tuple l the study participant ID, the latitude, the longitude, the information source (i.e. GPS, Network), the accuracy of the location point in meters, and the timestamp are recorded, respectively.

Then, we employ the well-accepted notion of *mobility trace* of an individual as a set of stops and moves [131, 126]. In this notion a stop is a set of latitude and longitude points where the individual is identified to spend a particular amount of time after performing a clustering procedure,explained in subsection 5.2.3.1 in detail. Formally, a stop in a place is defined as:

$$Place = [ID, t^a, t^d, C]$$

where ID, t^a, t^d and C stand for a place identifier, the arrival time, the departure time and the latitude-longitude coordinates, respectively. This information defines a mobility trace of places $MT(t_1, t_2)$ as the sequence of places visited by an individual in a given period of time:

$$MT(t_1, t_2) = (Pl_1, Pl_2, ..., Pl_{N(t_1,t_2)})$$

, where $N(t_1, t_2)$ is the total number of identified visited places.

5.2.2.2 *Daily Health Symptoms*

Data on physical health symptoms were collected using a daily self-reported survey instrument, designed by an experienced epidemiologist. The survey instrument consisted of eight questions with yes/no responses for each of the following symptoms: *fever*, *sore throat*, *cough*, *shortness of breath*, *headache*, *muscle pain*, *malaise*, and *cold*.

Symptom Type	**# Symptom Cases**	**# Unique Individuals**
fever	37	18
sore throat	196	40
cough	165	27
shortness of breath	86	15
headache	211	50
muscular pain	274	41
malaise	223	41
cold	174	34

Table 8: Description of the different Symptom Types, the number of cases that were present and the unique number of individual reporting each symptom.

Hence, the symptom raw variables have the following form: $symptom = [yes/no]$. In Table 8 we report the frequencies of the eight symptoms during the entire study duration and for each symptom the number of unique individuals reporting at least one case. The daily questions were answered at the evening by using SurveyGizmo and 64 participants, over a total of 70, reported at least one symptom.

5.2.3 *Extracting Mobility Behaviours*

Our main goal is to study the relationship between mobility behavior and self-reported symptoms. To do so, we need a set of characteristics that systematically describe human mobility behavior. Canzian et al. [126] recently introduced metrics able to capture both presence and absence of human mobility. Such features appear to be promising in identifying physical and mental health conditions, since many of them are related with the nature of the movement. For instance, in [126] they focus on depressive symptoms which could go along with decreased movement patterns and increased spending time at home for

a long-term period. In our case, we expect to identify similar signals, but in a short-term period.

5.2.3.1 *Identification of Places*

A very important step is the identification of places where the user is stopping. To this end, we create location clusters using raw data. In order to increase the accuracy of location estimation we consider only location points with accuracy less than 50 meters. Moreover, we ignore any location point that was collected while the user was moving. In order to infer such location points, we compute the speed of the individual by using the distance and the time between the last and the current location points. If the speed is less than a certain threshold (i.e. 5 km per hour) we consider that the location is collected when the participant was not moving.

Then, we use the location clustering approach presented in [118] in order to cluster the filtered location samples. We iterate over all location samples and for each location point we create a new cluster only if the distance of this location from the centroid of each existing cluster is more than 200 meters. Otherwise we add this location to the corresponding cluster and update its centroid.

Finally, we assign a unique place identifier to each centroid for all participants. Moreover, we assign the *home* label to the place where an individual spends the majority of the late evening and night hours (from 7pm to 7am), taking into account the habits in the northern part of Italy [132]. All the remaining places are labeled as *other*.

5.2.3.2 *Mobility Features*

For each individual, we compute all mobility features based on the visited *Places* we identify after performing the clustering procedure described above. The resulting set of mobility features is the following one:

1. **The total distance traveled** ($D_T(t_1, t_2)$): For computing the total distance traveled we consider: (i) the raw collected geo-location points when the individual is moving, and (ii) the detected stops in *Places*. We refer to them as points $p = [id, t^a, t^d, C]$ where $id{=}{-}1$ when the participant is moving and $id{>}0$ when s/he stops in a *Place*. For a time interval $[t_1, t_2]$,

this mobility trace is a set N_p of subsequent p points defined by a latitude-longitude pair C.

$$D_T(t_1, t_2) = \sum_{i=1}^{N_p(t_1,t_2)} d(C_i, C_{i+1}), \tag{29}$$

2. **The standard deviation of the total distance traveled** ($\sigma_{D_T}(t_1, t_2)$): the deviation from the average total distance (Feature 1), which is defined as:

$$Avg_{D_T}(t_1, t_2) = \frac{1}{N_p(t_1, t_2) - 1} \sum_{i=1}^{N_p(t_1,t_2)} d(C_i, C_{i+1}) \tag{30}$$

It is worth noticing that the number of movements is equal to the number of points minus 1.

3. **The total displacement** ($Dis_T(t_1, t_2)$): The total displacement is a measure of the distance covered by an individual. It takes into account the distance between one *Place* where the participant stopped and the subsequent one. Formally is defined as:

$$Dis_T(t_1, t_2) = \sum_{i=1}^{N(t_1,t_2)} d(C_i, C_{i+1}), \tag{31}$$

where $d(C_i, C_{i+1})$ is the geodesic distance between two visited identified places Pl_1 and Pl_2 with latitude-longitude coordinates C_i and C_{i+1}, respectively.

4. **The standard deviation of the displacements** ($\sigma_{Dis}(t_1, t_2)$): the deviation from the average displacement in $[t_1, t_2]$ as defined in [126].

5. **The maximum displacement between two visited Places** ($Dis_M(t_1, t_2)$): this metric quantifies the maximum displacement covered in $[t_1, t_2]$.

6. **The radius of gyration of the visited Places** ($G(t_1, t_2)$): We measure the radius of gyration as in [126], quantifying the span of the area the participant covers. It is the deviation from the centroid of the visited places in a $[t_1, t_2]$ interval weighting the contribution of each *Place* with coordinates C_i within the set N by the time spent there.

$$G(t_1, t_2) = \sqrt{\frac{1}{T} \sum_{i=1}^{N(t_1,t_2)} T_i \cdot d(C_i, C_{cen})^2}, \tag{32}$$

where T_i equals to $t_i^d - t_i^a$ representing the time spent in the place Pl_i and T is the total time spent in all the visited places in $[t_1, t_2]$.

7. **The maximum displacement from Home** ($Dis_{H(t_1,t_2)}$): this metric quantifies the maximum span the participant covered from its home. The ID and coordinates of the home for each participant is computed by considering the place with the maximum frequency of visits in *Places* considering time intervals between 7pm-7am, as explained in Section 5.2.3.1.

8. **The number of different Places visited** ($N_{dif}(t_1, t_2)$): Here we simply count the number of visits in different *Places* (i.e. the number of different places where that the individual visited) within the studied period. For example, if a participant visits within the study period Pl_1 and Pl_2 for one and two times, respectively, then the $N_{dif} = 2$.

9. **The number of different significant Places visited** ($N_{sig}(t_1, t_2)$): Here, we count the number of visits in significant Places within the period under observation. We consider significant a visited place if it belongs to the $top{-}10$ list of the most frequent visited *Places* in the time period of the study. In

10. **The number of moving geo-location points** ($N_{moves}(t_1, t_2)$): This is the count of the p geo-location points where $id{=}{-}1$ indicating that the participant was moving in the time interval $[t_1, t_2]$. It serves to quantify the moving behavior of a person.

11. **The unique number of visited Places** ($N_{unq}(t_1, t_2)$): This feature quantifies the distinct number of stops done or places visited.

12. **The diversity of the visited Places** ($Div_{visits}(t_1, t_2)$): This metric measures how an individual spreads its visits among places in a specific time interval. This metric was introduced in [21] and it is a sort of entropy and it measures mainly the diversity in social communication, but we apply it in a spatial context. Formally it is defined as:

$$Div_{visits}(i) = \frac{-\sum_{j=1}^{k} v_{ij} \log v_{ij}}{\log k}, \tag{33}$$

where v_{ij} is the volume of visits user i pays to the place j normalized by the total number of $i's$ visits, and k is the distinct number of places visited in the time interval, respectively. High values of the diversity measure indicate that participants distribute their visits more evenly among the places.

13. **Aggregated mobility features**: Previously observed mobility patterns in a participant's historical time-line can be useful to describe the trend of the participant's human mobility. In order to capture this information we defined a set of rolling statistics computed for each of the aforementioned mobility features. In particular, given a time window $[t_1, t_2]$ we aggregate the feature with the following statistics: mean, standard deviation, maximum, minimum and the difference of the feature values between the time t_1 and t_2.

5.2.4 *Classification Model*

We model our problem as a binary classification task, where the target variable is called *Symptom Presence* and the possible values of the label are {*Yes/No*}, that is if a user has or not at least one of the symptoms. Given a target date, our ultimate goal is to understand if a user will present or not symptoms in the forthcoming days by looking into its very recent mobility behavior. We expect to capture even slight changes in the mobility behavior (e.g. changes in covered distance) that can testify an upcoming flu and cold symptoms. Formally, given a date t we define:

- t_{hist} as the number of days we go back in individual's historical data from the date t;
- historical time interval as the time interval $[t - t_{hist}, t]$;
- t_{hor} as the number of days ahead we answer our *Symptom Presence: Yes/Not* question.

When $t_{hist} = [0, 2]$ and $t_{hor} = [0, 2]$, we consider a 5-day time window where the current day 0 is included in t_{hist} and t_{hor}. To sum up, we assign the label *Symptom Presence: Yes* to a user who presents flu-like and cold symptoms at time t_{hor}, by using historical data in the interval $[t - t_{hist}, t]$.

5.2.5 *Experimental Setup*

Due to the limited size of the data set, we decide not to built a specific model for each user. Indeed, we design a relatively general machine learning framework that can work for each user u. A sample for the model is built when more than three consecutive days of mobility data are available. Thus, given a date

t, we consider valid a time window of five days if the following conditions are satisfied: (i) mobility data for $t_{hist} \in [0, 2]$, and (ii) symptoms data for the time $t_{hor} \in [0, 2]$. As mentioned in Section 5.2.2, the data set contains gaps (i.e. location points and symptoms are not available for every day). Thus, it is possible that for some samples we do not have symptoms information for all the $t_{hor} \in [0, 2]$. In order to keep the dimension of train/test consistent and independent from the horizon time, we created different training and test sets for each t_{hor}. In this way, we avoid the possibility to have training samples with missing classification labels.

In Figure 24 we present a toy example of the prediction task and the constraints that apply for t_{hist}=2 and t_{hor}=2. Given a starting day t (e.g. March, 6th), we impose two constraints on each participant: (i) her/his mobility data must be available from day t until two days back in the past (e.g. from March, 4th until March, 6th), and (ii) her/his health data must be available from day t up to two days in the future (e.g. from March, 6th until March, 8th).

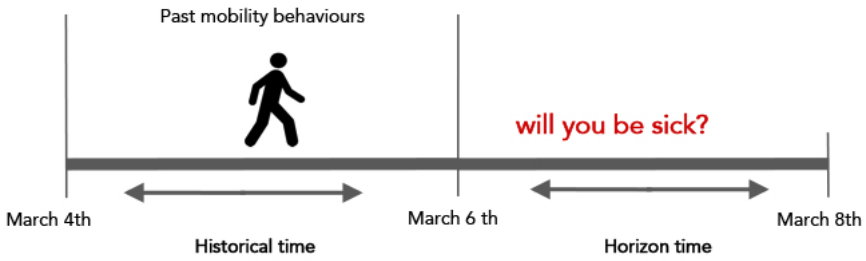


Figure 24: Example of problem setting with $t_{hist} = 2$ and $t_{hor} = 2$

As previously said, among the symptoms described in Section 5.2.2, we classify if a user will present at least one instance of *fever*, *sore throat*, *cough*, *shortness of breath*, *headache*, *muscle pain*, *malaise*, or *cold*. Although we selected a period of the year with many cases of flu-like and cold symptoms, we dealt with a highly unbalanced data set, meaning that the dominant class is the *NO* for the *Symptom Presence* variable. We used the common approach to randomly under-sample the data set by removing samples from the over-represented class.

In order to carry out our experiments, we split the data set in two parts: train and test. Then, we extract the features described in Section 5.2.3.2. For the classification task, we test four state-of-the-art machine learning models: Logistic Regression (LR) [137], Random Forest (RF) [135] and Gradient Boosted Trees (GBT) [128]. We selected these models because of their demonstrated effectiveness and, hence, popularity.

Due to the high number of features and the limited number of samples (i.e. 870 samples), we perform a feature selection step. For each classification model we evaluated several feature selection approaches by using 10-fold-cross-validation. Then, for each model we selected the best one. We found that Recursive Feature Elimination (RFE) is the best-performing feature selection method when using Logistic Regression (LR), Random Forest (RF), and Gradient Boosted Trees (GBT). We evaluate the quality of the feature selection through 10-fold-cross-validation, training the models with the reduced set of features on the training set. At this point, we can proceed with the parameters' optimization for each model by using the selected set of features. In both, feature selection and parameters selection, we choose an optimal set in order to maximize the precision of the algorithm. The last step regards the selection of the best model. Again, through cross-validation, we train each model with its best set of features and the optimal parameters selecting the one that shows the highest precision.

5.2.6 *Experimental Setup*

In our experiments we compare three different models (LR, RF, GBT) to classify if a user will present flu-like and cold symptoms or not (i.e. fever, sore throat, cough, shortness of breath, headache, muscle pain, malaise, or cold) at a time t_{hor}. To train our models, we use the machine learning library scikit-learn [127]. Due to the unbalanced nature of our data set, we use well-known metrics for assessing the accuracy of classification systems: (i) Precision, (ii) Recall, (iii) F1-score, and (iv) AUCROC. Precision is defined as the ratio $\frac{tp}{tp+fp}$, where tp is the number of true positives and fp is the number of false positives, while Recall is is defined as the ratio $\frac{tp}{tp+fn}$, where tp is the number of true positives and fn is the number of false negatives, which are samples erroneously not labeled as belonging to the positive class. F1-score is the harmonic mean of Precision and Recall. Finally, AUCROC refers to the Area Under the Receiver Operating Characteristic curve and provides information about the ability of the models to correctly classify users with or without flu-like and cold symptoms.

		t_{hist} = 0				t_{hist} = 1				t_{hist} = 2			
		Pr.	AUCROC	Re.	F1	Pr.	AUCROC	Re.	F1	Pr.	AUCROC	Re.	F1
t_{hor} = 0	LR	0.67	0.5	0.96	0.79	0.67	0.5	1.0	0.8	0.68	0.51	1.0	0.81
	RF	0.68	0.51	0.72	0.7	0.71	0.56	0.74	0.73	0.73	0.59	0.78	0.75
	GBT	0.69	0.53	0.81	0.74	0.74	0.61	0.84	0.79	0.7	0.56	0.82	0.76
t_{hor} = 1	LR	0.68	0.5	0.93	0.78	0.67	0.49	0.95	0.78	0.68	0.52	0.96	0.8
	RF	0.74	0.6	0.73	0.73	0.71	0.55	0.76	0.73	0.7	0.54	0.72	0.71
	GBT	0.7	0.54	0.77	0.73	0.74	0.62	0.87	0.8	0.71	0.56	0.8	0.75
t_{hor} = 2	LR	0.68	0.51	0.99	0.81	0.68	0.51	0.91	0.78	0.68	0.5	0.95	0.79
	RF	0.71	0.55	0.76	0.73	0.73	0.58	0.72	0.73	0.72	0.57	0.74	0.73
	GBT	0.71	0.55	0.85	0.77	0.72	0.57	0.81	0.76	0.72	0.57	0.84	0.77

Table 9: Precision (Pr.), Recall (Re.), AUCROC and F1-score of the classifiers obtained with 10-fold-cross-validation and variations of t_{hor} and t_{hist}.

5.2.7 *Results on Symptoms classification*

In Table 11 we present the classification results in terms of (i) Precision, (ii) Recall, (iii) F1-score, and (iv) AUCROC. We report the different performances for $t_{hist} \in [-2, 0]$ and $t_{hor} \in [0, 2]$. The results are obtained with 10-fold-cross-validation and using the best setup for each different model.

As expected, we observe that mobility features are relevant for predicting the presence of flu-like and cold symptoms. Interestingly, we obtain one of the best classification performance using Gradient Boosted Trees (GBT) with $t_{hist} = 1$ and $t_{hor} = 1$ (AUCROC of 0.62, a Precision score of 0.74, a Recall score of 0.87, and F1-score of 0.8). This is a consequence to the fact that people may change their mobility habits during the days before the self-reported registration of flu-like and cold symptoms, i.e. they change the mobility once they start to feel not very well. For instance, if a person is getting sick, he/she would likely go home after work instead of doing other activities.

Secondly, we can observe that as more days ahead we consider, more difficult it becomes to classify correctly the presence of symptoms by only looking at the mobility behaviors. This reveals an interesting aspect related to the fact that there is a short time period (e.g. few days) between feeling bad and reporting the symptoms. In summary, the obtained results suggest that mobility behavior can be used for our purpose, but only looking at a short period in the future (e.g. $t_{hor} = 2$) and considering a limited historical period. A long history of mobility data seems to be not relevant, a bigger sample size might be useful to better understand this point.

Moreover, for all the built models the following selected features (see Sec. 5.2.3.2) emerge as the most important ones in predicting correctly the presence of symptoms: (i) the diversity of visited places, (ii) the unique number of visited places, (iii) the number of different significant visited places, (iv) the number of moving geo-location points and (v) the aggregated mobility features. The first three features (i.e. diversity, unique number of visits and number of different significant visits) effectively capture a daily mobility routine of an individual. While the moving geo-location points quantify only the moving patterns of the participant, without considering the stops in places. Finally, the aggregated mobility features summarize the essential short-term history in people's mobility to detect changes (i.e. the aggregated mobility behavior during the crucial days before reporting the symptoms).

To summarize, the significant features belong to three different families: (i) visited places' routine, (ii) moving behavior and (iii) overall short-term historical mobility behavior.

For sake of completeness, we also report in Table 10 the confusion matrix for the case $t_{hist} = 1$ and $t_{hor} = 1$ using Gradient Boosted Trees (GBT), which refers to the best results in the setting of predicting future presence of flu-like and cold symptoms, i.e. one day ahead. The confusion matrix describes the performance of our classification model on the test set. We can observe that our model presents a sufficiently high accuracy in classifying the presence of symptoms while, mainly due to the difficult nature of the problem and the wide variety of symptoms we are considering, the performance with respect to the classification of the not presence of symptoms shows room for improvement.

	no symptoms	**symptoms**
no symptoms	0.32	0.68
symptoms	0.18	0.82

Table 10: The confusion matrix for the two-class classification task.

5.2.8 *Discussion*

Previous work has exploited the use of mobility features to predict mental health and well-being dimensions such as positive and negative emotions, stress level, and depression symptoms. To best of our knowledge, this work repre-

sents the first study that utilizes inference algorithms to predict the presence of influenza-like symptoms by only looking at the mobility behaviors of a specific individual. Our results represent a promising starting point for dealing with influenza-like public health issues. Turning to the limitations of our study, we list the small number of study participants used in our analyses (i.e. 29 individuals) and the short temporal duration of our study (i.e. only 4 weeks). However, it is worth noticing that the epidemic curve of 2012-2013 influenza season presents a peak during the four weeks selected for our study. In addition, the symptoms data are self-reported by the study participants. Finally, our sample is composed by parents. Hence, it may be plausible that the predictive performance of our approach is affected also by the changes in parents' mobility behavior related to the health status of the kids. For example, a parent may change her/his mobility behaviors in order to take the children to the doctor or to stay at home with the sick children. Moreover, the parent may get sick from her/his children, thus showing the symptoms few days later. Unfortunately, we do not collect data about the health status of the children due to privacy reasons. Therefore, future studies on different samples of study participants (e.g., students, older adults) should be conducted to better investigate the predictive role of changes in human mobility behaviors for the emergence of flu-like and cold symptoms.

5.3 DATA-DRIVEN MASS MIGRATION PREDICTION

In this section, we present our work on data-driven approach to enable responders to manage mass migration events [133].

5.3.1 *Overview*

Migration is an inherently complex and uncertain process. Direct observations of migration patterns are typically partial and inaccurate. Paths and destinations for migration are influenced by a range of human factors. Information sources for some factors may exist and come in different forms. For example, structured data from organizations such as registration counts at selected sites or refugee camps may be available. For locations without observers, one may need to rely on unstructured data such as news reports. Past data on migration movements and different data sources can be used to learn patterns of movements. Governments policies impacting migration patterns can change over

the course of time. The types of changes and the response of migrants to such changes can be difficult to ascertain from past observations. For such cases, approaches that model the underlying processes are required. The need for hybrid approaches that merge purely data-driven (capture past patterns) and ones that model the physics (capture aggregate interactions and exogenous inputs) are necessary. Models dealing with the spatial movement of people have sparked the interest of researchers for more than a hundred years, since Ravenstein's "Laws of Migration" [139]. The 20^{th} century saw a rise of the more quantitatively oriented models, among which Zipf's early intuition of spatial interactions as shaped by population size and distance (an analogy with the Gravitation Law), based on a previous formulation by Gaspard Monge [31]. Current trends in quantitative mobility models have been influenced in part by (i) the availability of personal digital traces such as mobile phone data, and (ii) new information signals, such as satellite imagery and crowd-sourced initiatives. Additional information signals that have been leveraged to understand migration recently include satellite imagery and crowd-sourcing efforts. Image classification routines on satellite images have been used to quantify size and growth of refugee camps. Examples include techniques using feature detection and extraction methodologies [142, 143, 144, 145] and deep learning and pattern recognition [147, 146]. Such techniques have been advocated for [140, 141] as a promising path to map temporary settlements and refugee camps. To address needs to end-user agencies, this work builds on these previous efforts by taking a multi-scale (global and local) approach that allows for scenario analysis. The multi-scale approach allows inclusion of factors at the global scale, while preserving detailed factors within the crisis region. Modeling scenarios allows for analysts to account for exogenous factors (e.g. repercussions of potential international policies).

In particular, we propose to model the problem of predicting mass migration by:

1. enhancing situational awareness using multiple data sources;
2. providing short and medium-term forecasts on migration patterns to aid operational decision making;
3. enabling 'what-if' scenario analysis for agencies looking to study the impact of exogenous factors.

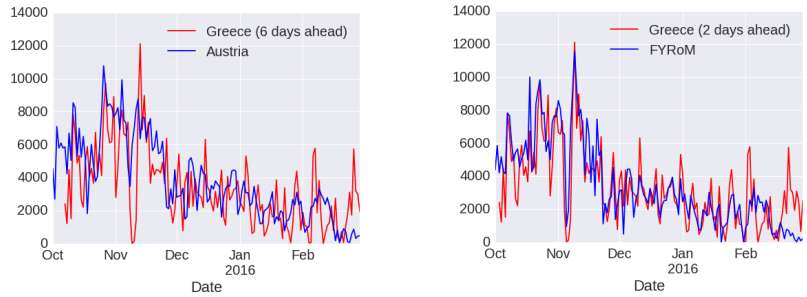
The model development is presented using the European migration crisis as a case study. The choice is motivated by availability of different datasets, for

example the large volume of news reports on the crisis and detailed registration data at various sites. The dynamics of the crisis were notably complex, as migrants were crossing several international boundaries.

5.3.2 *Data*

Complex processes like migration depend on a wide range of factors. Data sources that describe the socio-demographics, overall context, conflict conditions (in the case of forced displacements), and changing conditions over a period of time are all determinants of migrations and paths of flight. There is no single functional relationship between these variables, so its essential to understand key information signals that can add value to data-driven modeling. Using the European crisis as a case, some common information sources are described. For multi-scale approaches, the data sources related to both operational (e.g. daily) and strategic (e.g. annual) characteristics.

MIGRANT REGISTRATION DATA Registration data on migrant arrivals at key sites, such as the Greek islands provides daily, weekly or monthly arrival rates at various points along the crisis region. The UNHCR, UN's Refugee Agency, provides open data that is compiled from several sources[13]. The time series of arrivals at each of the sites can be used for several features in forecasting. Additionally, computing the cross-correlation function between the registration, provides an estimate of transit times through the network. Figure 25 shows examples of time series shift that give estimates of transit times between select countries.



(a) Arrivals in Greece (6 days ahead) and Austria

(b) Arrivals in Greece (2 days ahead) and FYRoM

Figure 25: Estimating transit time using a cross-correlation function

13 http://data.unhcr.org/mediterranean/regional.php

WEATHER DATA Information on weather conditions and seasonal factors influence movements. Adverse conditions are likely to restrict possible migration paths. For sea faring refugees arriving in the Greek islands, factors such as wind speed were found to be negatively correlated with arrivals over the winter months. Figure 26 shows the Pearson correlation coefficient between arrivals in different countries and different weather-related variables. Weather data is sourced from the Weather Underground[14] service.

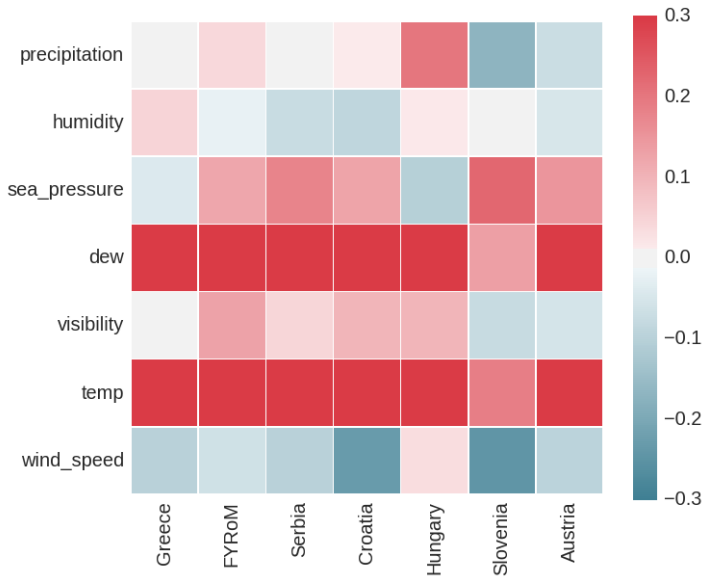


Figure 26: Pearson correlation coefficient between refugee/migrant arrivals and weather conditions for each of the involved countries.

NEWS DATA In order to capture information about policy changes (e.g. close of the Hungary border) and other external events, we used the news data provided by the GDELT Project [15]. It allows users to monitor the web news around the world and extract valuable information from text such as entities (e.g people locations, organizations), counts, emotions and events in over 100 different languages. Documents are annotated applying a combination of state-of-the-art natural language processing techniques. Moreover, it is worth to mention that for each article Global Knowledge Graph (GKG) provides a variable called *locations* in which they report the places mentioned in the article. This is an important information since it allow us to localize the potential country of the news.

14 https://www.wunderground.com
15 http://gdeltproject.org/

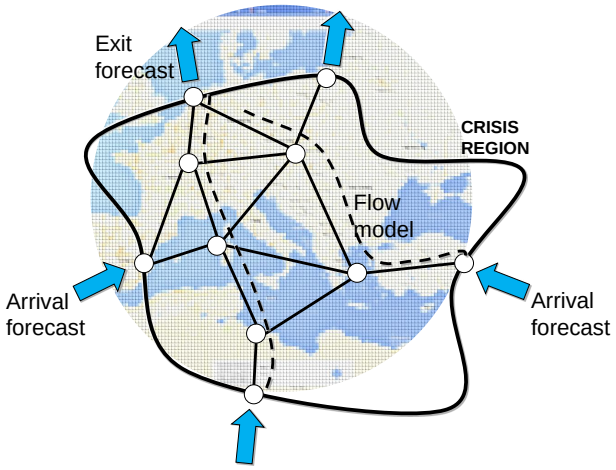


Figure 27: Analytical framework showing delineation of a crisis region of interest. Arrival forecasts at boundary nodes are estimated using arrival prediction models. Exit nodes forecasts are estimated from macro-migration model. A detailed network models the population movement within the crisis region.

COUNTRY-LEVEL DATA Aggregate statistics at the country level were collected from sources such as the World Bank. Socio-economic indicators such as GDP, population, conflict status were considered. Datasets from the gravity model literature[16] where included for dyadic features such as common languages, historical ties, trade volumes, and geographic distances.

5.3.3 *Analytics Framework*

The analysis framework consists of three distinct models that jointly provide forecasts. The main crisis region is modeled as a network flow model. The representation is detailed and the models the rate of movements from one node to another. The boundary conditions are handled by the other two models. An arrivals model is a time series forecasting model that predicts daily arrival rates at nodes on the edge of the crisis region. To determine exit rates and consider intended destinations, a macro push-pull model is used to estimate fraction of migrants for each likely destination. Figure 27 shows this framework.

5.3.3.1 *Arrival Prediction Model*

The aim of this model is to predict the number of arrivals at the entry-points of the crisis region. Future arrivals are dependent on external variables such

16 http://www.cepii.fr/

as weather, as well trends due to (de-) escalation of the crisis in question. To capture these effects we used data from weather reports and news reports and experimented with multiple machine learning models to predict day-ahead arrivals of refugees to the Greek Islands. Around 180 features were extracted from the following datasets:

AUTOREGRESSIVE FEATURES (AF) Previously observed value describe the trend of arrivals until the time of observation. In the model we use as a feature the number of arrivals one day before as well as rolling statistic such as mean, variance, maximum and minimum computed over the previous week.

WEATHER FEATURES (WF) We used six different weather variables: wind speed, temperature, sea level visibility, dew point, sea level pressure, humidity, and precipitation. To generate features, we aggregated the data computing the min, max, sum and mean of each variable over the day.

NEWS FEATURES (NF) To obtain news features, we queried the GKG historical data using the GKG Exporter [17]. We filtered the news selecting those in which the word *refugee* and either *migration* or *border* appear. Then, we selected only articles that match the location field with one of the countries involved in the migration crisis. Starting from these articles, three types of features are computed: (i) number of occurrences of relevant countries name (e.g. Syria and Greece) in the news, (ii) number of occurrences of relevant words related to humanitarian crisis (e.g. refugee camp) and, (iii) number of articles about humanitarian themes (e.g. migration). The former has been computed by applying a keyword-based approach where an article is assigned to a theme considering the presence of predefined words like *border* or *humanitarian*.

We applied four different machine learning models: LASSO, Linear Regression (LR), Ridge Regression (RR) and Gradient Boosting of Regression Trees (GBRT). In order to reduce the trend in the signals, we trained our model on the number of arrivals minus the rolling mean of the arrivals in the previous seven days. We added this value to the prediction before computing error metrics. The hyper-parameters of each of the models were determined through cross-validation; training the models on data between October 2015 and January 2016 and testing it on February 2016. Moreover, we normalized the feature values subtracting the mean and dividing by the variance. In Table 11 the

17 http://analysis.gdeltproject.org/module-gkg-exporter.html

results of the tested models are reported. Results of a persistence model prediction are also shown as a baseline. This model predicts arrivals each day to be equal to arrivals the previous day.

Due to the high number of features, we experimented with a feature selection step. To select features we first fitted one of the regression models. Then we ranked the features by the magnitude of their weight in the model (i.e. the slope). We used this approach on the results of LASSO, Linear Regression, and Ridge Regression. We also tried recursive feature elimination (RFE), and simple correlation.

We evaluated the quality of the feature selection by again through cross validation, training with the reduced set of features on the training set. RFE produced the worst set of features. The other methods produced similar sets of features with similar performance.

The smaller set of features lead to better model performance. After feature selection, each of the models shows improved performance, but the difference between the models is small. The best performance are obtained using the LASSO model on a subset of 10 features that included weather, autoregressive and news data. In particular, from the weather features we used the forecasting mean and max for wind speed, the forecasting gust speed and the forecasting mean for humidity, temperature and sea level pressure. Regarding autoregressive features, we used the difference in number of arrivals between one and two days before, the mean and the minimum in number of arrivals for last three days, the number of arrivals one day before and the sign of the derivative in the last two days. Finally, the model also includes the number of news (for the arrival country) about international organization for migration. Table 11 shows the results.

	No Feature Selection		Feature Selection	
Models	RMSE	Error Reduction(%)	RMSE	Error Reduction(%)
Baseline	1429.075	-	-	-
Linear Regression	$> 10^9$	>> 100	1111.214	22.24
Lasso	1197.096	16.23	1110.503	22.30
GBRT	1266.349	11.38	1257.430	12.01
Ridge Regression	1258.782	11.91	1116.311	21.88

Table 11: Arrivals model results for February in the Greek islands.

For all of the models, the weather features are the most present, followed by the historical data. This could be due to the fact that we tested our model on arrivals in the Greek Islands where weather conditions influence arrivals. Figure 28 shows a sample of forecasts for the Greek islands. There is not a significant difference in terms of results between the linear models after the feature selection step.

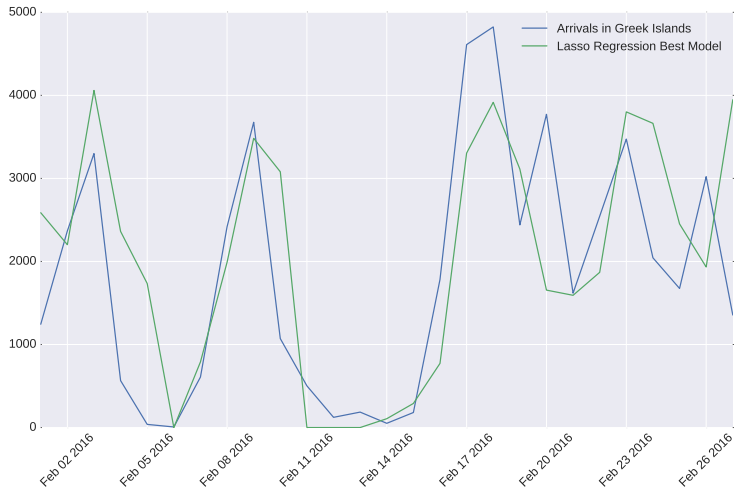


Figure 28: Models predictions in February in the Greek islands.

5.3.3.2 *Network Flow Model*

The challenge in modeling the paths of refugees through the crisis region is two-fold. First, measured arrivals at different location along a path through the crisis region are known to be inaccurate. Second, the movement patterns can change due to changing conditions on the ground, such as border closures. In this section we present a network flow model that can address both issues.

To mitigate the effect of inaccurate measurements, we developed a model that can impose some common sense boundary conditions on the prediction. Specifically, arrivals in each country must correspond to departures in a different country. Departures from any country cannot exceed the total number of refugees present in that country. Our model represents locations along the path of movement as nodes on a graph. The edges that connect the nodes i and j represent the likelihood that refugees will travel from node i to node j. As an example we will model travel of refugees from Greece to Austria through FYROM, Slovenia, Hungary, Croatia, and Serbia.

The number of people traveling from node i to node j at time t, denoted by $F_{ij}(t)$, is given by

$$F_{ij}(t) = P_i(t) f_{ij} \tag{34}$$

where $P_i(t)$ is the number of people present at node i, and f_{ij} is a constant that we will determine from historical arrivals data. The f_{ij} can be interpreted as split-fractions for the departures of each node. The net flow from node i to node j is then given by

$$N_{ij}(t) = P_i(t) f_{ij} - P_j(t) f_{ji} \tag{35}$$

To simplify the problem, we consider only these net flows. Arrivals ($A_i(t)$), and departures ($D_i(t)$) at node i are given by

$$A_i(t) = \sum_j \max(0, N_{ji}(t)) \tag{36}$$

$$D_i(t) = \sum_j \max(0, N_{ij}(t)) \tag{37}$$

The populations of the next time step can then be calculated as

$$P_i(t+1) = P_i(t) + A_i(t) - D_i(t) + E_i(t) \tag{38}$$

where E_i are exogenous arrivals, to be specified independently. Equations 34-38 can then be used iteratively to compute future flows.

In the present case we use E_i to specify arrivals to Greece from Turkey and departures from Austria to the rest of Europe. For arrivals to Greece, we use the outputs of the arrival prediction model discussed in the previous section. For departures from Austria, we assume all people present in the country, depart each day.

To determine the values f_{ij}, we optimized the following loss function.

$$L = \sum_{it} (A_i(t) - M_i(t))^2 + \alpha \sum_{ij} |f_{ij}| \tag{39}$$

where M_i are the measured arrivals at node i, and α is a regularization parameter. The second term in this equation is an L1 regularization. This regularization imposes sparsity in the possible paths.

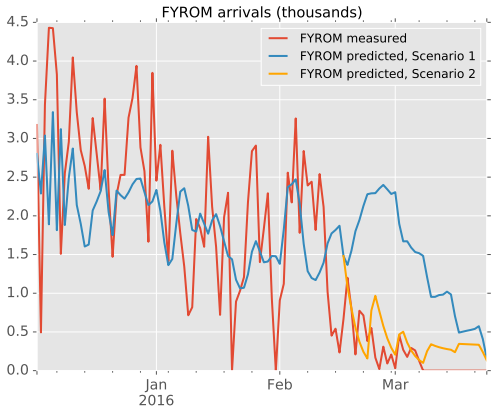


Figure 29: Output from network flow model - Predicted arrivals in the Former Yugoslav Republic of Macedonia (FYROM). The figure shows the measured arrivals as well as predicted arrivals in two different scenarios.

The model is trained at the beginning of each week on the preceding 30 days. Forecasts are then generated for the subsequent 2 weeks. This means that for each day, there are two forecast values; the mean of those two values is shown in Figure 29.

Changing conditions on the ground can be defined through adjustment of the exogenous arrivals. For example, starting mid February there were several policy changes that reduced the number of arrivals in Greece. This impacted arrivals in the other nodes of the network as well. We modeled this change by manually setting (exogenous) arrivals in Greece to 0 after February 16. Figure 29 shows both scenarios compared to measured arrivals. Scenario 1 assumes no change to the arrivals in Greece, Scenario 2 assumes no more arrivals after February 16. This method can be used to make more accurate predictions if policy changes are known. In addition, this methods could be used to do counter-factual scenario analysis.

5.3.3.3 *A push-pull macro migration model*

To derive destination preferences, a global model of migration that seeks to determine bilateral flows between countries is considered. The model seeks to estimate the fraction of refugees from each country that are likely to migrate to any other country. Such an approach is necessary to estimate macro-level movement patterns and serves to project intended destinations of migrants within the detailed network flow model.

The model assumes there are *push* (repulsion') factors at a home country along with *pull* (enticing) factors at the destination. Push factors at origin and

pull factors at destination cause migration. Movement is also a function of distance/affinity between two countries. Countries with higher affinity are likely have more migration. Affinity metrics can be defined in several ways. Classical approaches are using spatial properties such as geographic distance. Here, we consider an affinity function trained on past migration data and considering distance metrics, exogenous variables such as Gross Domestic Product (GDP), colonial relationships, commonality of language, contiguity, and conflict status of origin countries. One endogenous variable in bilateral migration in previous years (if available) was included to model 'social pull' factors. News articles were also considered to include more current reports of migration.

Since the true affinity measure is not known, a log-transformed linear model with these features is fitted on past bilateral migration and the set of features described above. Let N be a set of countries. Given a vector of inflows $I_j \; \forall \, j \in N$, a vector of outflows $O_i \; \forall \, i \in N$, and a matrix of distances/affinity metrics d_{ij} for each pair of sites, the following quadratic program seeks to estimate bilateral flows between a set of sites N (where internal migration is ignored).

$$\min \sum_{i \in N} \sum_{j \in N} M_{ij}^2 d_{ij} \tag{40}$$

subject to:

$$\sum_{j \in N} M_{ij} = O_i \qquad \forall \, i \in N \sum_{i \in N} M_{ij} = I_j \qquad \forall \, j \in N \tag{41}$$

The model computes M_{ij}, the migration flows between sites i and j. The numerical push-pull estimates can be ascertained by solving a linear system of equations, once all outflows and inflows are known.

$$\sum_{j \in S, j \neq i} M_{ij} = R_i \sum_{j \in S, j \neq i} \frac{1}{d_{ij}} + \sum_{j \in S, j \neq i} \frac{E_j}{d_{ij}} = O_i \; \forall \, i, \tag{42}$$

$$\sum_{i \in S, i \neq j} M_{ij} = \sum_{i \in S, i \neq j} \frac{R_i}{d_{ij}} + E_j \sum_{i \in S, i \neq j} \frac{1}{d_{ij}} = I_j \; \forall \, j, \tag{43}$$

where R_i is the 'repulsion' (push) factor and E_j are estimates of the 'enticing' (pull) factor. For distance-based affinity functions, the units associated with the push-pull quantities can be interpreted as 'person-kilometers'. However, for more complex affinity functions, the values are relative and cannot be interpreted directly.

In Table 12 we report results for three different affinity function. d indicates the affinity function, while in parentheses we report the different features that

are taken into consideration by the measure. The first case, Distance, considers geographic distances only. The second case, Distance + Exogenous Variables, additionally considers exogenous factors such as GDPs, common languages, contiguity, conflict status. The third case, Distance + Exogenous Variables + Past Migration, considers additionally social pull proxies, such as historical migration.

Affinity Function	**RMSE**
d (Distance)	11882.58
d (Distance + Exogenous Variables)	9494.30
d (Distance + Exo. Vars + Past Migration)	814.21

Table 12: RMSE for different cases in persons per year (2013)

5.3.4 *Discussion*

Data-aware tools, processes and methods have the potential to improve operations in the humanitarian sector. Forecasting and scenario modeling tools, such as those presented here, aid agencies to move to more proactive operations. There are several challenges to be addressed for wider uptake. The classical philanthropic engagement model with the private sector needs to shift to a more collaborative approach, where varied expertise across organizations can be tapped and models and methods refined. The work presented has the following limitations. Since each migration crisis is unique, impact of some features used to model the process will be different across different contexts. Past observed factors may have little or no role in future crises. For example, wind speeds used to forecast migration arrivals in Greece will not yield useful information for land-based migration. Partly relying on physical models and enabling scenario analysis helps mitigate some of these limitations. However, such models are based on assumptions that may also be specific to a particular context. Ensuring that the right assumptions are considered and appropriated incorporated within the models is key. Precise measurements, and in turn forecasts, remain a challenge on the ground and for models. While relative measures provide indications on how resources could be potentially deployed, the absolute numbers may be critical for some applications.

5.4 NEXT TAXI DESTINATION PREDICTION

5.4.1 *Overview*

Despite being far from adequately addressing the mobility management optimization problem, nowadays the variability of massive data describing human movements allows planners and policy-makers to face relevant urban challenges through computational methods [20, 133, 6, 102]. Current advances in technology made available a variety of low-cost electronic devices, which can be of great support to rapidly move in the city traffic. Inside taxis, electronic GPS terminals are installed in order to grab the vehicle position or the taximeter status, and send this information to the dispatcher unit. In addition to GPS data, there are novel sources of information, e.g., data from mobile phones and Location Based Social Networks (LBSNs), that can be exploited to model the human mobility behaviour and to optimize the city traffic. Common LBSNs, like Foursquare[18], for instance, can provide the number and type of activities present in a target area (e.g., Arts & Entertainment, Nightlife Spot, etc.), giving an insight on how many people can converge to that zone. All the collected information can then be used to infer the taxi destinations and the average travel time.

In human mobility, the most straightforward way to predict the next location is to build a grid over an area of interest, then treating the problem as a multiclass classification, where the aim is to predict the next visited cell. For instance, in the winning approach [99] of the ECML/PKDD 2015 challenge[19], the next taxi destination is obtained by employing a Multi-Layer Perceptron (MLP) network trained on the taxi trajectory, represented as a variable-length sequence of GPS points, and on diverse associated meta-information, such as the departure time, the driver identity, and the client information. Some of the limitations of previous approaches [99, 98] reside in the need of using the whole trajectory data and in the possibility of performing an instantaneous prediction of the destination only when most of the trajectory is available. To model the problem of next location prediction, some other works proposed neural network approaches. For instance [96, 97, 98, 95] present interesting solutions to capture the regularities characterizing human mobility based on taxi rides, while in [108] trajectories are modeled as two-dimensional images

18 www.foursquare.com

19 https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

that constitute the input for a Multi-Layer Convolutional neural network used to predict the next destination point.

This section is based on our work on next taxi destination prediction [4]. We propose a Long-Short-Term-Memory network [100], equipped with a self-attention module, to improve the prediction performance of the coordinates of the next drop-off point for a taxi. In particular, our model grounds on the individual driver's history, intended as the sequence of the last visited points, i.e., pick-up or drop-off points, instead of on the whole GPS trajectory. In order to represent such locations, we build a semantic representation based on LBSN data, coming from Foursquare. We tested our model on data referred to three different cities, namely Porto, as in the ECML/PKDD 2015 challenge, New York City (more specifically, Manhattan), and San Francisco. We have also compared our method with the winning model of the ECML/PKDD 2015 challenge under various settings, obtaining an improvement of 10.5% (equal to approximately 0.355 km) in the average error distance.

5.4.2 *Background and Problem Definition*

In this sub-section, we provide a formal definition of the taxi destination prediction problem and the background theory.

5.4.2.1 *Notation and problem definition*

Definition 5.4.1 *A spatio-temporal point p is defined as the couple $p = (t, l)$, where t indicates the time at which the location $l = (x, y)$ is visited, being x and y spatial coordinates in a given coordinate reference system (CRS), e.g. latitude and longitude in the CRS WGS84.*

In our framework, we aim at modeling the taxi drivers' behaviour. To this extent, it is useful to introduce the definition of what we call a *taxi trajectory*.

Definition 5.4.2 *Let u be a taxi driver. A taxi trajectory $T_u = p_1, p_2, ..., p_k$ is a time-ordered sequence composed by alternating pick-up and drop-off spatio-temporal points, describing the last $k/2$ taxi rides of driver u.*

As in the ECML/PKDD 2015 challenge, we focus on predicting where a taxi will drop-off a client.

Problem 5.4.1 *Let u be a taxi driver and T_u his/her taxi trajectory. Given $T_u \cup p_{k+1}$, being p_{k+1} the current pick-up point, we define the next taxi destination*

task as the problem of predicting the drop-off location $l_{k+2} = (x_{k+2}, y_{k+2})$*, which will be the actual destination of the driver* u*.*

5.4.2.2 *Recurrent Neural Networks*

Recurrent Neural Networks are equipped with feedback connections that produce internal loops. Such loops induce a recursive dynamics within the networks and thus introduce delayed activation dependencies across the processing elements. In doing so, RNNs develop a kind of memory, that makes them particularly tailored to process temporal (or even sequential) data, e.g., coming from written text, speech, or genome and protein sequences. Unfortunately, properly training RNNs is hard, due to both the vanishing and the exploding gradient pathologies, which introduce the long-term dependency problem, making difficult the processing of long sequences.

Long-Short-Term-Memory networks, introduced in [34] and usually just called LSTMs, are a special kind of RNNs, capable of learning long-term dependencies. All these networks have the form of a chain of repeated modules where each module is composed by four interacting layers with different functions.

In particular, the forget layer f_t selects the part of the cell state h_{t-1} which is responsible for removing the information no longer required for the LSTM to carry on its task. This allows the optimization of the LSTM performances. The forget gate processes also x_t, i.e. the input at the current time step. Instead, the input gate i_t, whose elaboration depends on h_{t-1} and x_t, is responsible for the addition of information to the cell state at time t; next, a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, possibly added to the cell state. Then, these two sources of information are combined to create an updated state. Finally, the output gate o_t selects those parts of the cell state that must be produced in output. The complete computing algorithm for LSTMs can be summarized as follows:

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= tanh(W_C[h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t tanh(C_t)
\end{aligned}
\tag{44}
$$

5.4.2.3 *Attention Mechanisms*

One of the most interesting human cognitive processes is the attention mechanism, a well studied phenomenon also in neuroscience. Indeed, just looking at an image, humans are able to focus on certain regions with *high resolution*, while perceiving the surrounding parts as *background*. In this way it is then possible to adjust the focal point over time. A very similar approach can be attached to deep learning tasks, even if attention in neural networks is, indeed, loosely related to the visual attention mechanism found in humans.

The main idea behind *attention* is to produce a score for each element of the current input (in the sequence case a score for each timestamp of the sequence itself). The implementation of this mechanism is quite straightforward and suited to this case study. Formally speaking, an attention model is a method that takes n arguments $y_1, \ldots, y_n$ and a context C, and returns a vector a which is supposed to be the summary of the inputs, focused on particular information dictated by the context C. This can be obtained as a weighted arithmetic mean of y_i, $i = 1, \ldots, n$, with the weights chosen according to the relevance of each y_i, given the context C.

Attention mechanisms have been applied to a very wide range of applications, such as speech recognition [104], machine translation [103], text summarization [105], and image description [106]. They are widely used also in few-shot learning [112, 113], in order compare an unknown sample and a set of labeled samples called the support set, indeed focusing the attention on the most similar support samples to the queried one. Finally, attention has been successfully employed for human mobility [95], in order to capture the important periodicities that govern human movements [20, 101].

5.4.3 *Data*

Taxi trajectory data are of great interest for the observation, evaluation, and optimization of transportation infrastructures and policies. For example, major problems of modern cities, such as traffic jams, are caused by an improper road planning, maintenance, and control. Taxi trajectory datasets are available nowadays for many large cities [114, 94] and they provide a valuable resource for modeling and understanding urban transportation patterns and human mobility behaviours. In our study, we focus on Porto, Manhattan (New York City)[20], and San Francisco [117]. In particular, the Porto dataset was released

20 https://chriswhong.com/open-data/foil_nyc_taxi/

in the context of the ECML/PKDD 2015 challenge and hosted as a Kaggle competition[21], which allows us to easily compare our model with the winner's approach. In order to provide semantic information about people's activities in each visited location, we enriched the taxi dataset with geo-located texts. In particular, we provide a feature representation of each location in terms of Foursquare Point-Of-Interests (POIs). We used freely-available data sources in order to make our research and experiments easily reproducible [22].

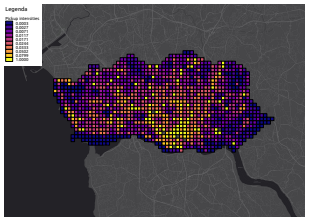


Figure 30: Pick-up intensities in the city of Porto.

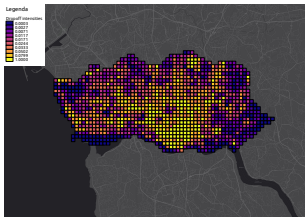


Figure 31: Drop-off (right) intensities in the city of Porto.

TAXI MOBILITY DATA The Porto taxi dataset is composed by 1.7 million records, coming from 442 taxis running in Porto, for a period ranging from 2013-07-01 to 2014-06-30. Fig. 31 reports the pick-up and drop-off distributions in Porto with data linearly scaled between 0 and 1. For each ride, a GPS trace is provided, making the description of a taxi trajectory different from the one introduced in Definition 5.4.2. In this case, the first point represents the pick-up location, the last one is the drop-off place, while in between there are spatio-temporal points sampled every 15 seconds. In addition, other metadata, like the *taxi id*, the *type* and the *origin of the call*, the *day type* (i.e., holiday, working day, weekend), and the *starting ride timestamp*, are attached to each trip. Such information is useful in order to detect some recurrent patterns, such as the flow of people who go to work or return home in specific time slots on working days. Similarly, for San Francisco, we can rely on trips of 536 taxis. The dataset contains GPS trajectories for 464,019 trips in 2008. The trajectories, as in the Porto dataset, describe the whole trip with points sampled every 10 seconds. The last, and biggest, dataset contains data for Manhattan and provide taxi trajectories defined as an Origin-Destination matrix. It is composed by 13,426 taxis, driven by 32,224 different drivers, during the entire 2013. Each month accounts for approximately 15 million trips. We selected the first three months of 2013, considering the most 5,994 active drivers, which turns out in

21 https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i
22 https://bitbucket.org/alberto91c/taxidestinationprediction

a total of 9,362,829 trips. All the above mentioned datasets do not contain vacant trips, which means that all the trips are made with a customer on board. In addition to the trajectory data, both these datasets also provide the driver's id and the starting time of the ride.

Following Definition 5.4.2, for both San Francisco and Porto, we map each GPS trace of the same taxi driver into a sequence of pick-up and drop-off points. More in detail, we group the trips by the driver's id, and then we sort them in ascending order based on their timestamp. Then, for each trace we select the pick-up and the drop-off point and create a taxi trajectory. In particular, given a set of taxi trajectories as in Def. 5.4.2 and the problem setting as in Problem 5.4.1, we impose $k = 8$ in order to keep up to four trips from the past, i.e., we use up to four pairs of pick-up and drop-off points. Such a choice represents a good trade-off between keeping a relevant past history while maintaining a sufficient number of trajectories for each driver and thus enabling the model to learn the cabdriver habits. Finally, in order to select trips relative to the same driver work shift, we use trips that are not separated by more than three hours each other.

Following this procedure, we obtain 260,600 trajectories for Porto, 87,548 for San Francisco, and more than a million for Manhattan. In the last case, we select 600 drivers out of the initial 5,994, for computational reasons, obtaining 184,000 sequences.

POINTS OF INTEREST In Section 3.2.1 we introduced Location-Base Social Networks data that provides, among the others, point-of-interest location and information. A Point Of Interest (POI) is usually characterized by a location (i.e., latitude and longitude), some textual information (e.g., a description of the activity in that place), and a hierarchical categorization, that provides different levels of detail about the activity of a particular place (e.g., *Food*, *Asian Restaurant*, *Chinese Restaurant*).

We extracted 8,928 POIs for Porto, 72,567 for Manhattan, and 30,059 for San Francisco. Despite the availability of many other interesting geographical datasets, e.g., land use and census data, we decided to limit our model by only using POIs data. This is due to the fact that those additional geographical datasets are not consistently defined among different cities, making their availability not uniform in terms of definition and coverage.

5.4.4 *Learning to Predict the Next Taxi Destination*

A mobility trace is a temporally-ordered collection of GPS locations. As the majority of sequential data, mobility traces are suited to be treated with models such as Recurrent Neural Networks (RNNs) instead of static architectures like Multi Layer Perceptrons (MLPs), which are not tailored to work with temporal and sequential data.

Previous approaches for predicting the next taxi destination were mainly focused on fine-grained single trajectories, meaning that they were based on the whole GPS trace of each ride [99], showing some important limitations such as a huge amount of data to be stored (i.e., all the GPS points of a trip), and the need of knowing almost all the trajectory in order to predict its final point. Instead, to avoid the aforementioned problems, we modeled the trajectory as in Definition 5.4.2. It turns out a sequence composed by pairs of GPS points (pick-up and drop-off locations) of several taxi rides. In this way, there is no need to wait the whole trajectory and the prediction can be performed instantaneously as soon as the trip starts and the pick-up location is available. Finally, in most of the mobility models, the next location prediction is set as a classification problem, where the goal is to classify to which locations, belonging to a known set, our driver is moving [98, 95]. The main drawback of this approach is that many locations will never be produced by the model, namely unseen locations in the training set. To overcome this limitation, we propose to predict the coordinates of the destination, by approximating directly two different functions, for the latitude and the longitude, respectively.

The adopted RNN architecture is relatively simple: it uses a Recurrent Neural Network in order to model the taxi behaviour, looking at the trip history and at the geographical information, to include the semantic meaning of each visited location. Our framework consists of three main components: (i) Feature Extraction and Embedding, (ii) Recurrent Module with Attention, and (iii) Prediction. The proposed neural network architecture is illustrated in Figure 32.

$$d_{haversine}(p_1, p_2) = 2R\left(\sqrt{\frac{a(p_1, p_2)}{a(p_1, p_2) - 1}}\right)$$

$$a(p_1, p_2) = sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos(\phi_1)cos(\phi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$$

where R is the earth radius and (λ_i, ϕ_i), $i = 1, 2$ are the longitude and the latitude of p_i, respectively.

We model each location with an embedding layer in order to capture all the factors that influence the human mobility, such as time, day and the semantic characterization of each place. In addition, we represent each location as a single word and we apply Word2Vec [115] on the resulting sequences, which allows us to obtain a dense representation based on the co-occurrence of both the origin and the destination. Then, based on a recurrent module, we collect all the sequential information we can derive from the past visited locations. For this task, we employ an LSTM architecture as the basic recurrent unit, because of its effectiveness in modeling human mobility [96]. Following the idea proposed in [95], we also apply an attention mechanism on the input sequence, in order to capture mobility regularities from previous visited locations. Finally, a prediction module composed by a softmax layer, followed by a linear layer, is designed to estimate the latitude and the longitude values of the next taxi destination.

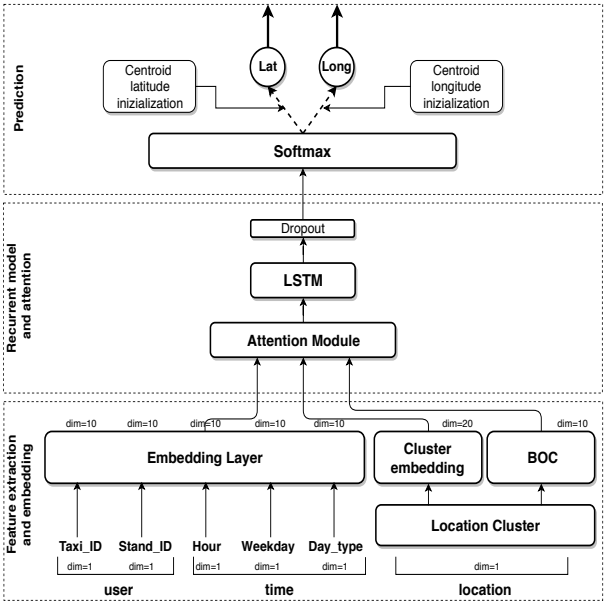


Figure 32: The architecture of our model with the three submodules: (i) Feature Extraction and Embedding, (ii) Recurrent Model and Attention, and (iii) Prediction.

5.4.4.1 *Feature extraction and embedding*

Mobility transitions are governed by multiple factors, such as the day-time and the geographical characteristics of visited places. In particular, we can assign to each location a *particular meaning* related to the activities present in that zone. Thus, an effective representation of a particular area turns out to be very important for mobility data analysis since it allows to enrich the information related to each location. We propose a multimodal embedding module to jointly embed the spatio-temporal and the driver features into a vector representation, which is then used as the input for the RNN-based prediction model.

DRIVER BEHAVIORAL FEATURES We represent the behaviour of the driver with a set of categorical features: (i) the time, expressed with the hour of the day $h \in [0, 23]$; (ii) the week-day $wd \in [0, 6]$; and (iii) the day-type, $dt \in [0, 2]$ (i.e., workday, pre-holiday, and holiday). Instead of modeling these categorical features with the simple one-hot representation, we follow the approach of producing a dense representation vector [115] based on an embedding layer, whose weights are updated during the training phase.

SPATIAL SEMANTIC FEATURES Given the set of location clusters C and a set of POIs, we assign each POI to the closest cluster. Thus, for each point of a given mapped trajectory CT_u, we build a feature representation making use of the associated POIs. Every venue is hierarchically categorized (e.g., *Professional and Other Places* → *Medical Center* → *Doctor's office*) and the categories are used to produce an aggregated representation of the area. We model spatial semantic features by using the Bag-Of-Concepts (BOCs) representation proposed in [2] and presented in Section 4.1.1, based on aggregating all the POIs associated to the trajectory and counting their macro-categories (e.g., *Food*). In other words, Bag-Of-Concept (BOC) features are generated by counting the number of activities for each category in a cluster.

SPATIAL ZONE EMBEDDING Spatial Semantic Features provide a time-independent representation, which does not capture the *dynamics* of a particular area. In this perspective, we can use human mobility data to learn a dense representation based on the mobility flows in the city. In the resulting space, embeddings of zones with similar urban mobility will result geometrically close each other. Human mobility can be seen as a language, where sequences of locations are sequences of words. It turns out that sequence models

designed for textual data, and used in natural language processing (NLP), can be properly applied to a sequence of locations. For instance, Word2Vec [115] is a well-known textual embedding technique to learn word embeddings. Given a word, we learn the embedding from the co-occurrence words belonging to a near by word-window. In this way, in the generated space of Word2Vec, two semantically similar words will results to have a similar vector representation. Given the set of location clusters C, we assign a textual label to each cluster. Then, we map each trajectory in a sequence of words, enabling Word2Vec to learn the zone embedding by relying on the mobility relation from different zones.

5.4.4.2 *Recurrent module with attention*

Recurrent Neural Networks are particularly suited to process sequential data, such as GPS traces [96, 98, 95]. For this reason, we model our problem using Recurrent Neural Networks with an LSTM as a basic unit, having ReLu as the activation function. First, we apply the attention mechanism to the spatio-temporal vector sequence in input. The implemented structure is presented in Figure 33. The aim of the attention module is to learn on which part of the trajectory is more important to focus on. This is done by transposing the input and feeding it to a softmax, which estimates the weight distributions, that are then combined with the input sequence. Thus, the recurrent layer is focused on specific parts of the sequence, to capture mobility regularities from the current trajectory. Finally, a dropout layer is applied before the softmax layer, in order to prevent overfitting.

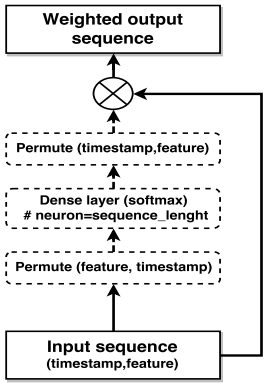


Figure 33: The attention mechanism applied on the input trajectory.

5.4.4.3 *Prediction*

The prediction module represents the last component of our architecture, that combines the output of the previous modules and completes the prediction task. In particular, it consists of a softmax layer and a linear layer. First, the softmax layer, having $m = |C|$ as the number of neurons, takes the representation generated by the recurrent module as its input. Then, since the network must evaluate the coordinates of the destination point, an additional output layer with two neurons is added, representing the latitude and the longitude coordinates, respectively. It is worth noting that this operation is equivalent to add a simple linear output layer, whose adjustable weight matrix is initialized with the cluster centers. Thus, the output of this layer is defined as:

$$\tilde{y} = \sum_{i=1}^{C} P_i c_i$$

$$P_i = \frac{exp(e_i)}{\sum_{j=i}^{C} exp(e_j)}$$

where P_i is the softmax probability associated to each cluster point.

5.4.5 *Experimental setup*

We performed experiments in Porto, Manhattan and San Francisco by using 260.600, 184.000 and 87.500 taxi rides, respectively. For the three cities we used the whole datasets, randomly splitting the data into 65%-15%-20% for the training, validation, and test sets. For the city of Porto and San Francisco the available dataset is composed by the complete trajectories, while for Manhattan the trajectories are only composed by pick-up and drop-off points, as in Definiton 5.4.2). Due to the unbalancing of the datasets and the spatial nature of the problem, standard classification measures such as accuracy and F1-score are not appropriate, not giving an adequate quantification of the error. Hence, we used the Error Distance Score (EDS), which is defined as the Haversine distance between the predicted point and the actual destination of the trip:

$$EDS = d_{haversine}(\tilde{y}, y)$$

where $\tilde{y}$ is the predicted point and y is the correct destination.

The experimental study compares our work with several baseline approaches and state-of-the-art models, listed in the following:

- **Nearest Neighbors (NN)**: Given a pick-up point, it outputs the coordinates of the closest cluster centroid.
- **MMLP**: The Multi-Layer Perceptron (MLP) implementation that won the ECML/PKDD 2015 challenge [99]. In this model, the input layer receives a representation of the taxi trajectory, composed by the first five and the last five GPS points, and its associated metadata. The network is composed by a standard hidden layer, containing 500 Rectifier Linear Units (ReLUs) [111], followed by a softmax layer with $m = |C|$ neurons. As in our approach, each coordinate is assigned to a given set of cluster centroids. Thus, the output layer predicts $\tilde{y}$, which is a weighted average between softmax output and the destination cluster centroids. The optimization problem is modeled as a multi-class classification, having the cross-entropy as the loss function. The network is trained with Stochastic Gradient Descent with a batch size of 200, a learning rate of 0.001 and a momentum term of 0.9. To perform a fair comparison, we trained the MMLP model on the last trip of our trajectory (Definition 5.4.2), thus obtaining the same number of training, validation and test patterns.
- **MMLP-SEQ**: It is trained following the same approach of MMLP but, instead of using the first five and the last five GPS points as input, we have linearized the driver's trajectory (Definition 5.4.2).

5.4.5.1 *Training and hyperparameter setting*

We tuned the parameters of our models for each city by performing a grid search on the number of neurons, on the number of layers and on the learning rate. The performance was evaluated on the validation set and the selected values are reported in Table 13. The K parameter used in the K-means algorithm is set to 3,392 for Porto, as in the original work [99], and to 2000 for Manhattan and San Francisco, being both areas smaller than the extension of Porto. Further investigations on the number of clusters could be of interest but are out of the scope of this paper. For the embedding of both the driver and the time features, we adopted a layer of size 10, for each single feature, as in [99]. The size of the location embedding is 20, while the BOC representation has 10 components, equal to the number of the macro-categories present in Foursquare. The input dimension of the LSTM (with ReLu activation functions [111]) is equal to the size of the vector obtained after concatenating all the previous representations. To train the network, we used the Adam optimizer [93], approximating the latitude and the longitude values separately, but sharing the underlying

structure — embeddings, attention module and LSTM layer. The network is trained using the Mean Squared Error (MSE) as the loss function and implementing the early stopping. This means that we stop the training procedure if no changes in the MSE occur on the validation set for at least 10 epochs. We compute the MSE score on the validation set after each epoch and save the network parameters if a new best MSE score is obtained. During the test phase, we use the parameters of the network that produced the best MSE score on the validation set. The dropout rate is set to $p = 0.5$. The word embedding for the textual label of each cluster is done by using the Gensim[23] implementation [110]. We opted for a Continuous Bag-of-Words (CBOW) model with a window size of 5 and without filtering words on frequency. The dimensionality of this embedding is set to 20.

City	LSTM Neurons	Learning Rate	Batch Size
Porto	128	10^{-3}	64
San Francisco	128	10^{-3}	64
Manhattan	256	10^{-3}	64

Table 13: Training parameters for each city.

5.4.6 *Results*

The performances of the compared approaches are listed (per column) in Table 14. We indicate with LSTM the simplest approach that uses the input representation only constituted by the driver and the time representation. The zone, i.e., the coordinates of the cluster location, is fed into an embedding layer with randomly initialized weights, allowing their update during the training. LSTM (BOC), indicates the model with the BOC features in input (see Section 5.4.4.1). Finally, LSTM (BOC+W2V) represents the Recurrent Neural Network that includes both the BOC features, but with the representation of the zone embedding obtained with (W2V).

All the models reported in Table 14 have been trained on the same set of trajectories. We note that:

- LSTM (BOC+W2V) reduces the EDS for Porto and Manhattan, outperforming MMLP of 10.5%, and 18%, respectively;

23 https://radimrehurek.com/gensim/

Model	Porto	San Francisco	Manhattan
NN	3.215	3.023	2.375
MMLP	3.211	1.994	$2.543^{(*)}$
MMLP-SEQ	3.003	2.762	2.554
LSTM	2.923	2.547	2.111
LSTM (BOC)	2.923	2.397	**2.085**
LSTM (BOC+W2V)	**2.88**	**2.270**	2.088

Table 14: Error Distance Score (km) for Porto, San Francisco and Manhattan. $^{(*)}$ The trajectory is composed by pick-up and drop-off points only.

- in San Francisco, where the trajectories are really dense of coordinates, MMLP obtains better results. However, such a result is due to the over-simplification of the problem, since the model works with the first five and last five trajectory points, meaning that the last input point is really close to the final destination of the taxi. Indeed, MMLP-SEQ, which is the MMLP used with a linearized driver trajectory, is outperformed by 17% by the LSTM (BOC+W2V);
- the usage of spatial semantic features and zone embeddings (BOC+W2V) improves over LSTM;
- the results of MMLP in Porto are lower than those obtained in the challenge, due to the small number of trips (only 300) contained in the Kaggle private test set. As explained in the original work [99], such model performs worst than other techniques, e.g, Bidirectional Long-Short-Term-Memory Networks (BLSTM) [116], when evaluated on a bigger test set obtained by slicing the training set.

In Figure 34 we compare LSTM (BOC+W2V) both in the regression and in the multiclass classification settings. We trained the classification models using the categorical cross-entropy as the loss function and removing the two output neurons. In this way, the output locations are limited to the list of cluster centroids. Finally, the output coordinates are obtained, as in [99], by weighting each cluster centroid through the corresponding probability coming from the softmax layer. The experimental results show that approximating the exact location, i.e., by applying the regression procedure on the latitude/longitude

values, allows to reduce EDS both in Manhattan, showing the ability of the proposed approach to better perform in cities not equally elongated in latitude and longitude.

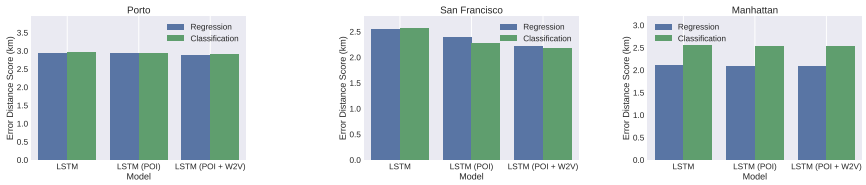


Figure 34: Classification vs. Regression in Porto, San Francisco and Manhattan.

5.4.7 *Discussion*

In this work, we have introduced a new way of looking at the problem of the next destination prediction in a taxi journey. To the best of our knowledge, we are also the first ones to propose a model that treats this task as a regression problem, instead of a multiclass classification one [98, 95]. Moreover, instead of using the complete taxi trajectory, we rely on the individual driver's history, namely the sequence of the last visited points by the driver, i.e., pick-up and drop-off points. Thus, we are able to obtain an instantaneous prediction, just after the starting of the trip, which results in information of paramount relevance for the taxi company dispatcher. This information may improve and optimize fleet management, save costs and reduce waiting times for the customers. In addition, our proposed model captures the taxicab driver's habits, looking at his/her recent history, and this information can be useful for policy makers and urban transportation planners to monitor the impact of taxicab drivers' habits on the whole traffic of a city.

6

CONCLUSION AND FUTURE WORK

Learning from heterogenous urban data sources is a key aspect when approaching Urban Computing tasks. In particular, it emphasizes the focus on the need for an input object representation that can effectively aggregate the knowledge from the different urban data sources. In this thesis, we address this challenge by proposing novel models to represent urban spaces by means of kernel methods. Current structural kernels were designed for very different tasks, for example, for syntactic parse tree reranking in the context of natural language processing applications. In contrast, we need to model a structural representation of an entire city area, which introduces two different aspects: (i) different venues, e.g., restaurants, schools and residential buildings, may assume different importance for an area and (ii) simple feature-based representations might be not expressive enough in capturing information from a such complex structured input. To overcome these problems, we introduced a novel structural representation that leveraging on the hierarchical structure of Location-Based Social Networks can encode data into a tree structure, the so-called GeoTree. Then, we used such representations in kernel machines, which can thus perform accurate classification exploiting hierarchical substructure of concepts as features. Finally, to explore these research lines and ideas, we experimented with such representations to solve novel tasks such as predicting the next taxi destination.

In Chapter 2 we provide the background theory of the supervised machine learning algorithms that are at the core of the models presented in this thesis. In particular, we introduce the convolution tree kernels that have been successfully applied in many natural language processing tasks.

In Chapter 3 we introduce the most common urban data sources and we describe our work about the richest open multi-source dataset ever released. This dataset is a multi-source aggregation of telecommunication, weather, news, social network and electricity data which we believe will stimulate researchers to design algorithms able to exploit an enormous number of behavioral and social indicators.

Chapter 4 contains the two main contributions of the thesis. In particular we have introduced a framework for automatically analyzing city areas using com-

plex semantic structures, i.e., concept hierarchies of point-of-interest. First, we present novel structural semantic representations for urban zones. Then, we design a new general kernel, which can take node weights into account, while computing the structure matching in the space of the exponential number of substructures. We demonstrate that this approach is largely applicable as (i) it can use any hierarchical category structure for POI categories (e.g. OpenStreet Map POI data); and (ii) many cities offer open access to their land use data. Finally, we propose to characterize and analyze a city in terms of its tree fragments. We propose to apply a mining algorithm to extract the most relevant features (tree fragments) from the TK space according to their weights assigned by the kernel machine. We apply our approach to the *land use classification* of an urban area, a task of paramount relevance in Urban Computing. Our experiments show the effectiveness of both our hierarchical representation, also tested with more traditional kernels, and our new Hierarchical Point-Of-Interest Kernel, which significantly improves the state-of-the-art. In this thesis, we introduce the idea of representing urban zones with semantic information. In this perspective, following the recent trend in NLP and many other research fields, a future research line could go in the direction of uncovering urban dynamics by learning spatial embeddings from large-scale urban data.

In Chapter 5 we present our previous works, based on multi-source urban datasets, on three challenging novel tasks: (i) predicting influenza-like symptoms using mobility data; (ii) data-driven mass migration prediction and (iii) predicting next taxi destination. In particular, the results of our work on predicting taxi destination [4] demonstrate how using (i) geo-located semantic information, such as Foursquare POIs [109], and (ii) embedding urban zones by considering mobility flows strongly improves the prediction accuracy.

biblio/thesis-articles.bib biblio/thesis-others.bib

[16] George S Kimeldorf and Grace Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[17] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[18] Michael Batty. The size, scale, and shape of cities. *Science*, 319(5864):769–771, 2008.

[19] Manlio De Domenico, Albert Solé-Ribalta, Sergio Gomez, and Alex Arenas. Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences*, 111(23):8351–8356, 2014.

[20] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *nature*, 453(7196):779, 2008.

[21] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[22] Matthew James Kelley. The semantic production of space: Pervasive computing and the urban landscape. *Environment and Planning A*, 46(4):837–851, 2014.

[23] D. Haussler. Convolution kernels on discrete structures. Technical report, University of California, Santa Cruz, 2008. Technical Report.

[24] Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 458–467, 2013.

[25] Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 75–83, 2013.

[26] Aliaksei Severyn, Alessandro Moschitti, Olga Uryupina, Barbara Plank, and Katja Filippova. Multi-lingual opinion mining on youtube. *Information Processing & Management*, 52(1):46–60, 2016.

[27] Danilo Croce, Alessandro Moschitti, and Roberto Basili. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics, 2011.

[28] Andrey Bogomolov, Bruno Lepri, Roberto Larcher, Fabrizio Antonelli, Fabio Pianesi, and Alex Pentland. Energy consumption prediction using people dynamics derived from cellular network data. *EPJ Data Science*, 5(1):13, 2016.

[29] Aamena Alshamsi, Edmond Awad, Maryam Almehrezi, Vahan Babushkin, Pai-Ju Chang, Zakariyah Shoroye, Attila-Péter Tóth, and Iyad Rahwan. Misery loves company: happiness and communication in the city. *EPJ Data Science*, 4(1):7, 2015.

[30] Rex W Douglass, David A Meyer, Megha Ram, David Rideout, and Dongjin Song. High resolution population estimates from telecommunications data. *EPJ Data Science*, 4(1):4, 2015.

[31] Filippo Simini, Marta C González, Amos Maritan, and Albert-László Barabási. A universal model for mobility and migration patterns. *Nature*, 484(7392):96, 2012.

[32] Vladimir M Krasnopolsky and Michael S Fox-Rabinovitz. Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction. *Neural Networks*, 19(2):122–134, 2006.

[33] Rodger A Brown, Corey Potvin, and Amy McGovern. Using large-scale machine learning to improve our understanding of the formation of tornadoes. In *Large-Scale Machine Learning in the Earth Sciences*, pages 95–112. Chapman and Hall/CRC, 2017.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[35] Chaoyun Zhang and Paul Patras. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 231–240. ACM, 2018.

[36] Adrian Albert, Jasleen Kaur, and Marta C Gonzalez. Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1357–1366. ACM, 2017.

[37] Department of Economic and Social Affairs. World urbanization Prospects: the 2014 Revisions, Highlights. Technical report, Population Division United Nations, 2014.

[38] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[39] S. Fritz, L. See, C. Perger, I. McCallum, C. Schill, D. Schepaschenko, M. Duerauer, M. Karner, C. Dresel, J.-C. Laso-Bayas, M. Lesiv, I. Moorthy, C.F. Salk, O. Danylo, T. Sturn, F. Albrecht, L. You, F. Kraxner, and M. Obersteiner. A global dataset of crowdsourced land cover and land use reference data. *Scientific Data*, 4:1–8, 2017.

[40] Chao Zhang, Keyang Zhang, Quan Yuan, Fangbo Tao, Luming Zhang, Tim Hanratty, and Jiawei Han. React: Online multimodal embedding for recency-aware spatiotemporal activity modeling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 245–254. ACM, 2017.

[41] Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. KeLP: a Kernel-based Learning Platform in java. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*, Lille, France, July 2015. International Conference of Machine Learning.

[42] Daniele Pighin and Alessandro Moschitti. On reverse feature engineering of syntactic tree kernels. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 223–233. Association for Computational Linguistics, 2010.

[43] Edward Glaeser. *Triumph of the city: How our greatest invention makes us richer, smarter, greener, healthier, and happier*. Penguin, 2011.

[44] Daniele Pighin and Alessandro Moschitti. Efficient linearization of tree kernel functions. In *Proceedings of the Thirteenth Conference on Compu-*

tational Natural Language Learning, pages 30–38. Association for Computational Linguistics, 2009.

[45] Taku Kudo and Yuji Matsumoto. Fast methods for kernel-based text analysis. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 24–31. Association for Computational Linguistics, 2003.

[46] Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224, 2008.

[47] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. Exploiting semantic annotations for clustering geographic areas and users in location-based social networks. *The Social Mobile Web*, 11(2), 2011.

[48] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, New York, NY, USA, 2016. ACM.

[49] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *KDD*, pages 186–194. ACM, 2012.

[50] Haytham Assem, Lei Xu, Teodora Sandra Buda, and Declan O'Sullivan. Spatio-temporal clustering approach for detecting functional regions in cities. In *ICTAI*, pages 370–377. IEEE, 2016.

[51] Jameson L Toole, Michael Ulm, Marta C González, and Dietmar Bauer. Inferring land use from mobile phone activity. In *SIGKDD International Workshop on Urban Computing*, pages 1–8. ACM, 2012.

[52] Nuria Oliver, Aleksandar Matic, and Enrique Frias-Martinez. Mobile network data for public health: opportunities and challenges. *Frontiers in public health*, 3, 2015.

[53] Marco De Nadai, Jacopo Staiano, Roberto Larcher, Nicu Sebe, Daniele Quercia, and Bruno Lepri. The death and life of great italian cities: a mobile phone data perspective. In *Proceedings of the 25th International Conference on World Wide Web*, pages 413–423. International World Wide Web Conferences Steering Committee, 2016.

[54] Vıctor Soto and Enrique Frias-Martinez. Robust land use characterization of urban landscapes using cell phone data. In *The first workshop on pervasive Urban Applications (PURBA)*, 2011.

[55] Pierre Deville, Catherine Linard, Samuel Martin, Marius Gilbert, Forrest R Stevens, Andrea E Gaughan, Vincent D Blondel, and Andrew J Tatem. Dynamic population mapping using mobile phone data. *Proceedings of the National Academy of Sciences*, 111(45):15888–15893, 2014.

[56] Fabien Girardin, Francesco Calabrese, Filippo Dal Fiore, Carlo Ratti, and Josep Blat. Digital footprinting: Uncovering tourists with user-generated content. *IEEE Pervasive computing*, 7(4), 2008.

[57] Francesco Calabrese, Giusy Di Lorenzo, and Carlo Ratti. Human mobility prediction based on individual and collective geographical preferences. In *ITSC*, pages 312–317. IEEE, 2010.

[58] Vanessa Frias-Martinez, Victor Soto, Heath Hohwald, and Enrique Frias-Martinez. Characterizing urban landscapes using geolocated tweets. In *SocialCom*, pages 239–248. IEEE, 2012.

[59] Yihong Yuan and Martin Raubal. Extracting dynamic urban mobility patterns from mobile phone data. In *International Conference on Geographic Information Science*, pages 354–367. Springer, 2012.

[60] Jonathan Reades, Francesco Calabrese, Andres Sevtsuk, and Carlo Ratti. Cellular census: Explorations in urban data collection. *IEEE Pervasive Computing*, 6(3), 2007.

[61] S. V. N. Vishwanathan and Alexander J. Smola. Fast kernels for string and tree matching. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 569–576. MIT Press, 2002.

[62] T Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[63] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, pages 318–329. Springer, 2006.

[64] Balázs Cs Csáji, Arnaud Browet, Vincent A Traag, Jean-Charles Delvenne, Etienne Huens, Paul Van Dooren, Zbigniew Smoreda, and Vincent D Blondel. Exploring the mobility of mobile phone users. *Physica A: Statistical Mechanics and its Applications*, 392(6):1459–1473, 2013.

[65] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[66] Markus Schläpfer, Luís MA Bettencourt, Sébastian Grauwin, Mathias Raschke, Rob Claxton, Zbigniew Smoreda, Geoffrey B West, and Carlo Ratti. The scaling of human interactions with city size. *Journal of the Royal Society Interface*, 11(98):20130789, 2014.

[67] Giovanna Miritello, Rubén Lara, Manuel Cebrian, and Esteban Moro. Limited communication capacity unveils strategies for human interaction. *Scientific reports*, 3:1950, 2013.

[68] Thomas Louail, Maxime Lenormand, Oliva G Cantu Ros, Miguel Picornell, Ricardo Herranz, Enrique Frias-Martinez, José J Ramasco, and Marc Barthelemy. From mobile phone data to the spatial structure of cities. *Scientific reports*, 4, 2014.

[69] Nathan Eagle, Michael Macy, and Rob Claxton. Network diversity and economic development. *Science*, 328(5981):1029–1031, 2010.

[70] Christopher Smith-Clarke, Afra Mashhadi, and Licia Capra. Poverty on the cheap: Estimating poverty maps using aggregated mobile communication networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 511–520. ACM, 2014.

[71] Andrey Bogomolov, Bruno Lepri, Jacopo Staiano, Nuria Oliver, Fabio Pianesi, and Alex Pentland. Once upon a crime: Towards crime prediction from demographics and mobile data. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 427–434. ACM, 2014.

[72] Amy Wesolowski, Nathan Eagle, Andrew J Tatem, David L Smith, Abdisalan M Noor, Robert W Snow, and Caroline O Buckee. Quantifying the impact of human mobility on malaria. *Science*, 338(6104):267–270, 2012.

[73] Daniele Quercia, Jonathan Ellis, Licia Capra, and Jon Crowcroft. Tracking gross community happiness from tweets. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 965–968. ACM, 2012.

[74] Bartosz Hawelka, Izabela Sitko, Euro Beinat, Stanislav Sobolevsky, Pavlos Kazakopoulos, and Carlo Ratti. Geo-located twitter as proxy for global mobility patterns. *Cartography and Geographic Information Science*, 41(3):260–271, 2014.

[75] Maxime Lenormand, Miguel Picornell, Oliva G Cantú-Ros, Antònia Tugores, Thomas Louail, Ricardo Herranz, Marc Barthelemy, Enrique Frías-Martinez, and José J Ramasco. Cross-checking different sources of mobility information. 2014.

[76] Chen Zhong, Stefan Müller Arisona, Xianfeng Huang, Michael Batty, and Gerhard Schmitt. Detecting the dynamics of urban structure through spatial network analysis. *International Journal of Geographical Information Science*, 28(11):2178–2199, 2014.

[77] Federico Botta, Helen Susannah Moat, and Tobias Preis. Quantifying crowd size with mobile phone and twitter data. *Royal Society open science*, 2(5):150162, 2015.

[78] Paolo Bajardi, Matteo Delfino, André Panisson, Giovanni Petri, and Michele Tizzoni. Unveiling patterns of international communities in a global city using mobile phone data. *EPJ Data Science*, 4(1):3, 2015.

[79] Blerim Cici, Minas Gjoka, Athina Markopoulou, and Carter T Butts. On the decomposition of cell phone activity patterns and their connection with urban ecology. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 317–326. ACM, 2015.

[80] Manlio De Domenico, Antonio Lima, Marta C González, and Alex Arenas. Personalized routing for multitudes in smart cities. *EPJ Data Science*, 4(1):1, 2015.

[81] Yves-Alexandre de Montjoye, Zbigniew Smoreda, Romain Trinquart, Cezary Ziemlicki, and Vincent D Blondel. D4d-senegal: the second mobile phone data for development challenge. *arXiv preprint arXiv:1407.4885*, 2014.

[82] Vincent D Blondel, Markus Esch, Connie Chan, Fabrice Clérot, Pierre Deville, Etienne Huens, Frédéric Morlot, Zbigniew Smoreda, and Cezary Ziemlicki. Data for development: the d4d challenge on mobile phone data. *arXiv preprint arXiv:1210.0137*, 2012.

[83] Anastasios Noulas, Cecilia Mascolo, and Enrique Frias-Martinez. Exploiting foursquare and cellular data to infer user activity in urban environments. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 167–176. IEEE, 2013.

[84] Andrey Bogomolov, Bruno Lepri, Michela Ferron, Fabio Pianesi, and Alex Sandy Pentland. Daily stress recognition from mobile phone data, weather conditions and individual traits. In *Proceedings of the ACM International Conference on Multimedia*, pages 477–486. ACM, 2014.

[85] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[86] Qian Zhang, Corrado Gioannini, Daniela Paolotti, Nicola Perra, Daniela Perrotta, Marco Quaggiotto, Michele Tizzoni, and Alessandro Vespignani. Social data mining and seasonal influenza forecasts: The fluoutlook platform. In *Machine Learning and Knowledge Discovery in Databases*, pages 237–240. Springer, 2015.

[87] Marcel Salathe, Linus Bengtsson, Todd J Bodnar, Devon D Brewer, John S Brownstein, Caroline Buckee, Ellsworth M Campbell, Ciro Cattuto, Shashank Khandelwal, Patricia L Mabry, et al. Digital epidemiology. *PLoS Comput Biol*, 8(7):e1002616, 2012.

[88] Michael Collins and Nigel Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *ACL*, 2002.

[89] Michele Tizzoni, Paolo Bajardi, Adeline Decuyper, Guillaume Kon Kam King, Christian M Schneider, Vincent Blondel, Zbigniew Smoreda, Marta C González, and Vittoria Colizza. On the use of human mobility proxies for modeling epidemics. *PLoS computational biology*, 10(7):e1003716, 2014.

[90] Vincent D Blondel, Adeline Decuyper, and Gautier Krings. A survey of results on mobile phone datasets analysis. *EPJ data science*, 4(1):10, 2015.

[91] Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. Identifying important places in peopleś lives from cellular network data. In *In-*

ternational Conference on Pervasive Computing, pages 133–151. Springer, 2011.

[92] Xianyuan Zhan, Satish V Ukkusuri, and Feng Zhu. Inferring urban land use using large-scale social media check-in data. *Networks and Spatial Economics*, 14(3-4):647–667, 2014.

[93] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[94] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

[95] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1459–1468. International World Wide Web Conferences Steering Committee, 2018.

[96] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, pages 194–200, 2016.

[97] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, 2016.

[98] Di Yao, Chao Zhang, Jianhui Huang, and Jingping Bi. Serm: A recurrent model for next location prediction in semantic trajectories. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2411–2414. ACM, 2017.

[99] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021*, 2015.

[100] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[101] Christian M Schneider, Vitaly Belik, Thomas Couronné, Zbigniew Smoreda, and Marta C González. Unravelling daily human mobility motifs. *Journal of The Royal Society Interface*, 10(84):20130246, 2013.

[102] Luca Pappalardo, Filippo Simini, Salvatore Rinzivillo, Dino Pedreschi, Fosca Giannotti, and Albert-László Barabási. Returners and explorers dichotomy in human mobility. *Nature communications*, 6:8166, 2015.

[103] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[104] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.

[105] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[106] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[107] Austin W Smith, Andrew L Kun, and John Krumm. Predicting taxi pickups in cities: which data sources should we use? In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pages 380–387. ACM, 2017.

[108] Jianming Lv, Qing Li, Qinghui Sun, and Xintong Wang. T-conv: A convolutional neural network for multi-scale taxi trajectory prediction. In *Big Data and Smart Computing (BigComp), 2018 IEEE International Conference on*, pages 82–89. IEEE, 2018.

[109] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang, Jieping Ye, and Weifeng Lv. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD International Confer-*

ence on Knowledge Discovery and Data Mining, pages 1653–1662. ACM, 2017.

[110] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.

[111] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[112] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. 2018.

[113] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[114] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98. ACM, 2011.

[115] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[116] T. Thireou and M. Reczko. Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(3):441–446, 2007.

[117] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). https://crawdad.org/epfl/mobility/20090224.

[118] Fani Tsapeli and Mirco Musolesi. Investigating causality in human behavior from smartphone sensor data: a quasi-experimental approach. *EPJ Data Science*, 4(1):1–15, 2015.

[119] Michele Tizzoni, Paolo Bajardi, Adeline Decuyper, Guillaume Kon Kam King, Christian M. Schneider, Vincent Blondel, Zbigniew Smoreda, Marta C. Gonzalez, and V. Colizza. On the use of human mobility

proxies for modeling epidemics. *PLOS Computational Biology*, 10((7): e1003716), 2014.

[120] Andrey Bogomolov, Bruno Lepri, and Fabio Pianesi. Happiness recognition from mobile phone data. In *Proceedings of the International Conference on Social Computing (SocialCom)*, pages 790–795. IEEE, 2013.

[121] Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. Predicting depression via social media. In *Proceedings of the AAAI International Conference on Weblogs and Social Media (ICWSM)*. AAAi, 2013.

[122] Enrique Frias-Martinez, Graham Williamson, and Vanessa Frias-Martinez. An agent-based model of epidemic spread using human mobility and social network information. In *Proceedings of the International Conference on Social Computing (SocialCom)*, pages 57–64. IEEE, 2011.

[123] Anmol Madan, Manuel Cebrian, David Lazer, and Alex Pentland. Social sensing for epidemiological behavior change. In *Proceedings of the 12th ACM international Conference on Ubiquitous Computing (UbiComp)*, pages 291–300. ACM, 2010.

[124] Simone Centellegher, Marco De Nadai, Michele Caraviello, Chiara Leonardi, Michele Vescovi, Yusi Ramadian, Nuria Oliver, Fabio Pianesi, Alex Pentland, Fabrizio Antonelli, et al. The mobile territorial lab: a multilayered and dynamic view on parents' daily lives. *EPJ Data Science*, 5(1):1, 2016.

[125] Anmol Madan, Manuel Cebrian, Sai Moturu, Katayoun Farrahi, and Alex Pentland. Sensing the "health state" of a community. *IEEE Pervasive Computing*, 11(4):36–45, 2012.

[126] Luca Canzian and Mirco Musolesi. Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1293–1304. ACM, 2015.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duch-

esnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[128] Robert E. Shapire and Yoav Freund. *Boosting: Foundations and algorithms*. MIT Press, Cambridge, MA, 2012.

[129] Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. Sacry: Syntax-based automatic crossword puzzle resolution system. *ACL-IJCNLP 2015*, page 79, 2015.

[130] Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. Learning to rank answer candidates for automatic resolution of crossword puzzles. *CoNLL-2014*, page 39, 2014.

[131] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto, and Christelle Vangenot. A conceptual view on trajectories. *Data & knowledge engineering*, 65(1):126–146, 2008.

[132] Francesco Calabrese, Laura Ferrari, and Vincent D Blondel. Urban sensing using mobile phone network data: a survey of research. *ACM Computing Surveys (CSUR)*, 47(2):25, 2015.

[133] Mohammed N Ahmed, Gianni Barlacchi, Stefano Braghin, Francesco Calabrese, Michele Ferretti, Vincent Lonij, Rahul Nair, Rana Novack, Jurij Paraszczak, and Andeep S Toor. A multi-scale approach to data-driven mass migration analysis. In *SoGood@ ECML-PKDD*, 2016.

[134] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. Moodscope: building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 389–402. ACM, 2013.

[135] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[136] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, 2002.

[137] David R. Cox. The regression analysis of binary sequences (with discussion). *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

[138] Influnet: Sorveglianza epidemiologica.

[139] E Ravenstein. On the Laws of Migration. *The effects of brief mindfulness intervention on acute pain experience: An examination of individual difference*, 48(2):167–235, 1885.

[140] Einar Bjorgo. Using very high spatial resolution multispectral satellite sensor imagery to monitor refugee camps. *International Journal of Remote Sensing*, 21(3):611–616, 2000.

[141] UNHCR. Refugees, Asylum-Seekers, Returnees, Internally Displaced and Stateless Persons. Technical report, UNHCR, 2009.

[142] Shifeng Wang, Emily So, and Pete Smith. Detecting tents to estimate the displaced populations for post-disaster relief using high resolution satellite imagery. *International Journal of Applied Earth Observation and Geoinformation*, 36:87–93, 2015.

[143] D Tiede, S Lang, D. Hölbling, and P. Füreder. Transferability of obia rulesets for idp camp analysis in darfur. *Geobia*, 2006, 2010.

[144] G Laneve, G Santilli, and I Lingenfelder. Development of automatic techniques for refugee camps monitoring using very high spatial resolution (VHSR) satellite imagery. In *2006 IEEE International Symposium on Geoscience and Remote Sensing*, pages 841 – 845, Denver, CO, 2006. IEEE.

[145] S. Giada, T. De Groeve, D. Ehrlich, and P. Soille. Information extraction from very high resolution satellite imagery over Lukole refugee camp, Tanzania. *International Journal of Remote Sensing*, 24(22):4251–4266, 2003.

[146] Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. Transfer Learning from Deep Features for Remote Sensing and Poverty Mapping. *arXiv.org preprint*, page 16, 2015.

[147] Lars Roemheld. Humanitarian Mapping with Deep Learning. 2010.