International Journal on Artificial Intelligence Tools

Where to Park? Predicting Free Parking Spots in Unmonitored City Areas

Andrei Ionita

Computer Science, RWTH Aachen University
Aachen, Germany
andrei.ionita@rwth-aachen.de

André Pomp

Institute of Information Management in Mechanical Engineering, RWTH Aachen
Aachen, Germany
andre.pomp@ima.rwth-aachen.de

Michael Cochez

Fraunhofer FIT
Aachen, Germany
Computer Science 5, RWTH Aachen
Aachen, Germany
Faculty of Information Technology, University of Jyväskylä
Jyväskylä, Finland
michael.cochez@fit.fraunhofer.de

Tobias Meisen

Institute of Information Management in Mechanical Engineering, RWTH Aachen
Aachen, Germany
tobias.meisen@ima.rwth-aachen.de

Stefan Decker

Computer Science 5, RWTH Aachen
Aachen, Germany
Fraunhofer FIT
Aachen, Germany
stefan.decker@dbis.rwth-aachen.de

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Several smart cities around the world have begun monitoring parking areas in order to estimate free spots and help drivers that are looking for parking. The current results are indeed promising, however, this approach is limited by the high costs of sensors that need to be installed throughout the city in order to achieve an accurate estimation rate. This work investigates the extension of estimating parking information from areas equipped with sensors to areas that are missing them. To this end, similarity values between city

neighborhoods are computed based on background data, i.e., from geographic information systems. Using the derived similarity values, we analyze the adaptation of occupancy rates from monitored- to unmonitored parking areas.

Keywords: smart parking, machine learning, semantic annotation, data mining

1. Introduction

Parking is a known problem in cities. Worldwide we are experiencing an increase in the number of cars [1]. Currently, about 30% of the traffic in cities is caused by cars that are actively searching for parking [2]. Drivers often end up double-parking their cars, which blocks other cars, thus causing unneeded stress. Drivers that are circling for a parking space may cause safety issues, as they are often distracted and are not paying attention to cyclists and pedestrians. When circling for parking spaces, additional fuel is consumed, which has an economical as well as an environmental impact.

Existing parking spaces can be more efficiently occupied if the drivers know about their availability. This information needs to be available in advance, so that drivers can take the decision to drive towards a highly probable free parking space early enough and not get stuck in a traffic bottleneck. A software system predicting available spaces would, ideally, take into account factors such as current parking space availability, traffic, events in the near vicinity, weather, and many more. Existing forecasting systems often start by collecting statistics on available parking spaces. Usually, mounted sensors observe when a car occupies and leaves a parking space. Acquiring the required data, however, be it on the parking spaces themselves or the complementary information, is, most of the time, the bottleneck for these approaches.

Our approach introduces the notion of a *parking demand profile*, which reflects the time of the day where parking occurs and its duration for a given area. Furthermore, we examine how machine learning models for parking occupancy can be transferred between city areas with a similar profile. We start by giving an intuitive example which bridges the understanding path towards the rest of the work. Further, other contributions to the field of smart parking are presented, followed by the concrete approach. The evaluation in this case carries out the instantiation of the proposed approach with an actual use case. We present the results and an analysis. We finish off by pointing out aspects that can be further pursued and by summarizing the outcome. This work is a short version of an earlier master thesis by the first author of this article [3].

2. Motivating Example

In this section, we provide a motivating example illustrating the necessity for introducing the concept of *parking demand profiles* to improve the accuracy of machine learning models for predicting free parking spots in smart cities.

The scenario consists of a large smart city with different parking areas, which are

either dedicated parking spots (e.g., car parks or parking lots) or parking spaces on the side of the street. Some of these parking areas are equipped with sensors to measure the occupancy rate whereas other areas are tracked using cameras or parking meters. Due to the high costs for tracking all parking areas (especially those which offer free parking), most parking areas are not tracked at all. To improve the parking situation in the city, the local government decides to publish the available data sources, such as the sensor measurements, on their Open Data platform.

Data scientists can now obtain the data and start building predictive models for parking areas at which data are available. For that, the data scientist examines the data, performs feature engineering and selection, trains the models and develops an application that predicts the occupancy rates for different locations of the city.

Examining this scenario, we identify several drawbacks. Due to the heterogeneity of available parking data sources and information, it becomes quite hard for the data scientist to build the models. First, the data scientist has to understand each data source in detail based on the information provided on the Open Data platform. If the data were provided homogeneously (e.g., by using RDF and a well-defined ontology), it would become easier for the data scientist to understand the data from one source and to aggregate the data from multiple sources. In addition, the application can be used in other cities, which provide their data based on the same ontology.

Besides the understanding of the data, the data scientist still faces the problem of not covering all parking areas of the city. If we assume that different models for the different parking areas have been built, it is unclear whether any of these models can be applied on untracked areas. Hence, our approach exploits the use of parking profiles which represent parking in a city area. One example would be an area consisting of office buildings. Here, parking demand is usually high during working time, e.g., between 8 and 18 o'clock. Two such areas, perhaps in different parts of a city, or even in different cities, likely have a similar parking situation, i.e., between 8 and 18 on weekdays there is a high parking demand. In case of restaurants, on the other hand, we see a spike in parking demand in the evening, usually from 18 until 22, and even more so on weekends. In residential areas, the cars are parked in the evening and leave again early in the morning. A measure that would capture parking demand will therefore be based on the stay duration of customers or employees of the particular services, the available number of shops and restaurants or other point of interests (e.g., event halls).

For such city areas (e.g., office buildings, restaurants, residential areas, etc.) that have parking occupancy data, the data scientist can build the predictive models. Then, using parking profiles, one can attempt using models created for tracked areas in untracked ones. Hence, we investigate in this article how we can transfer these models to other city areas for which no parking occupancy data is available, but that have similar parking demand profiles.

3. Related Work

The goal of improving the parking situation by estimating free parking spots is well known for many decades. Hence, there are many works related to the problem we are addressing. We shall briefly describe the research in the area of smart parking, while concentrating on the status quo of parking prediction systems.

3.1. *Smart Parking Overview*

For an overview of smart parking projects, including development cost and the number of parking spaces covered, see Figure 1.

Project	City	#Parking	Cost	App	Launched
Stadtinfo Köln	Cologne	16000	-	-	09/1996
MOBINET	Munich	2750	4M€	-	09/1998
SIMERT	Loja	1974	-	SIMERT Loja Ecuador	05/2002
Area Verda (22@)	Barcelona	50000	-	apparkB	04/2005
Cabspotting(ParkNet)	San Francisco	281364	400$/veh	-	04/2006
PARK Smart	New York City	177	-	Parker	10/2008
SFpark	San Francisco	8200	23M$	SFpark	04/2011
SmartSantander	Santander	400	8.67M€	SmartSantanderRA	06/2011
Parking in Melbourne	Melbourne	7114	537,817$A	Street Parking	07/2011
WSN-DPCM	Madrid	15	3.3M€	-	10/2011
Nice Park	Nice	10000	13-15M€	Nice City Pass	01/2012
Optimod'Lyon	Lyon	30	7M€	Optymod'Lyon	02/2012
Smart Parking Pilot	Beijing	20000	-	-	03/2012
Stockholm Parkering	Stockholm	37000	-	P-Stockholm	05/2012
LA ExpressPark	Los Angeles	6300	18.5M$	Parker,ParkMobile,ParkMe	06/2012
ParkRight	London	3400	889,395£	ParkRight	10/2012
Moscow Fastprk	Moscow	50000	-	Parking Moscow	11/2012
ParkChicago	Chicago	36000	153M$[a]	ParkChicago	04/2014
ParkBoston	Boston	8000	-	ParkBoston	01/2015
Smart Canton	Geneva	16	-	-	01/2015
Berlin Pilot	Berlin	50-70	-	-	09/2015
Citypark Bordeaux	Bordeaux	-	87,000€	Citypark	10/2015

Fig. 1. Large-scale smart parking projects

In the scientific literature there have been an increased number of publications on smart parking. We only touch upon some topics, such as parking reservation and resource allocation, our main focusing being occupancy prediction. We refer to a recent smart parking survey completed in 2017, that presents the developments in the field. The survey [4] is a follow-up to the main author's PhD thesis published in 2015 on the same topic [5]. It presents the relevant developments on smart parking solutions since 2000. We shall use this survey to describe the research landscape for this topic. Afterwards, we will focus on parking prediction and we will go in depth on the relevant work done in that department.

As the research and development have been going in all directions, Lin et al. split the available results in *information collection*, *system deployment*, and *service dissemination*.

Under information collection, the authors include all techniques to acquire parking information. Be it static or mobile, in the form of sensors or as parking meters, the infrastructure here is diverse. Most of the time, the deployed sensors are managed through a wireless network infrastructure. The sensors are present inside cars,

such as taxis, which travel through the city and collect roadside information on parking. The piece of information transmitted is an occupancy change bit: either the car has left or arrived at a parking space. A number of types of sensors are usually being used in the process: infrared sensors, ultrasonic sensors, accelerators, optical sensors, inductive loops, piezoelectric sensors, cameras, acoustic sensors. Most of the time, the captured data requires post-processing in the form of image or audio recognition before arriving at the target occupancy information. Some of the captured information also raises privacy issues, at it contains sensitive data about the car and driver. Smartphones provide the means to collect data and can have a great impact through crowdsourcing, if the users are given incentives to enable the respective smartphone functions. On the other hand, there are systems where the driver initiates the data transmission. By using a mobile application, the driver may choose to report that he or she just left a parking space, respectively has arrived at a parking space.

In the system deployment section, the survey refers to the varieties of parking systems, looks into how well they scale, and touches upon their data analysis side. The parking systems software is the interface between the data sources and the users. Software systems are often in the form reservation systems, typically run by municipalities. Other systems also may provide guiding assistance in arriving at the desired parking space. Their vacancy prediction component informs the drivers about the availability of parking spaces at the destination. In this way, they contribute to the decision of the driver, of whether to take the car to the destination or not. Prediction systems recommend parking locations that have at least one parking space free.

Under data dissemination, the authors of the survey address the ways in which parking data is exchanged between drivers. This scenario often occurs in decentralized parking systems, when the drivers find out about free parking spaces in an area where other cars drove by. Dynamic pricing is often used to control parking occupancy: parking prices are raised in areas that are almost fully occupied, whereas areas with low parking rates get assigned a lower price. A more advanced version adjusts the prices when enough demands received by the parking system would point to a future parking overload in the respective area. These approaches take into account parking competition: multiple drivers are looking to park in the same parking location.

In order to view the related work comparatively, we have gathered the benchmark data from the relevant literature into Table 1, which shows an overview of the systems presented in the papers, alongside the data sources used. Further specified are whether the system is dedicated to on- or off-street parking, and the location where it was evaluated.

Table 1. Overview of all the systems introduced in individual articles

Paper	Year	System	Data Sources	On/Off -Street	Location
Caliskan [6]	2007	VANET	vehicle sensors	off-street	Brunswick, Germany
Klappenecker [7]	2010	VANET	vehicle sensors	off-street	-
Wu [8]	2014	parking guidance	municipality	off-street	Taipei City
Tiedemann [9]	2015	parking prediction	sensors	on-street	Berlin
Rajabioun [10]	2013	parking guidance	on-site sensors: infrared, image-processing cameras, inductor loops	both	Los Angeles, San Francisco
Rajabioun [11]	2015	parking guidance	sensors (cf. SF*park* [12])	both	Los Angeles, San Francisco
Vlahogianni [13]	2014	routing system	IoT nodes: car presence	on-street	Santander
Z. Chen [14]	2013	Location-based service for parking finding	manual collection	parking lots	Perth, Australia
Xu [15]	2013	parking prediction	mobile phone	on-street	San Francisco
X. Chen [16]	2014	parking prediction	SF*park* sensors	on-street	San Francisco
Hössinger [17]	2014	parking prediction	parking tickets manual counts traffic flow	on-street	Vienna
Caicedo [18]	2012	parking reservation	in-cars sensors	both	Barcelona
Nandugudi [19]	2014	parking prediction	smarphone	surface lots	Buffalo
Szczurek [20]	2010	VANET	GPS in car	both	-
Ji [21]	2014	parking guidance	-	off-street	Newcastle
Koster [22]	2014	parking recommender	smartphone	both	New York
Kotb [23]	2016	parking reservation	occupancy sensors	-	-
Pullola [24]	2007	parking prediction	GPS in car	on-street	Ottawa
Richter [25]	2014	-	SF*park* sensors [12]	on-street	San Francisco
Shin [26]	2014	-	sensors parking facilities cruising cars	both	Luxembourg
Zheng [27]	2015	-	parking sensors	on-street	San Francisco Melbourne

3.2. *Vacancy Prediction Systems*

On the topic of prediction, there are a number of papers that approach the problem. The circumstances differ in each case, so that it is worth mentioning the papers individually first. Afterwards we will attempt to compare these approaches by extracting the most relevant aspects, e.g. method, type of data, features, error, deployment location, etc., in a table.

As in Section 3.1, all relevant work found on the topic are listed in Tables 2 and 3. The key data shown emphasizes the prediction character of the papers by highlighting their general approach, features, methods, and evaluation.

Xu et al. [15] makes real-time parking availability estimations based on a system that aggregates the data coming from mobile phones. The system uses algorithms based on statistical weighted schemes and Kalman filters. Additionally, the authors create parking availability profiles based on historical data and using statistical

Table 2. Overview of all the prediction methods used in related work

Paper	Approach	Features	Methods	Evaluation
Caliskan [6]	parking prediction	age of parking info, arrival time, total capacity, occupancy, arrival rate, parking rate	Markov chains	probability density, relative deviation
Klappenecker [7]	parking prediction	total capacity, occupancy, arrival rate, parking rate	Markov chains	-
Wu [8]	parking recommendation	occupancy	probabilistic	historical mean, moving average
Tiedemann [9]	parking prediction	occupancy	data threads	-
Rajabioun [10]	parking prediction	pricing, events, location, total capacity, occupancy rate, leaving rate	probabilistic	mean error
Rajabioun [11]	parking prediction	occupancy, total capacity	multivariate-autoregressive model	orthogonality error differential covariance mean absolute -percentage error
Vlahogianni [13]	parking prediction	free space duration, occupancy, traffic volume, type of day, time period	genetically optimzed, multilayer perceptron	Weibull distribution
Z. Chen [14]	parking prediction	parking times	fuzzy logic	occupancy error shortest travel time
Xu [15]	real-time estimation	parking times	Kalman adaptive linear memory filter	RMSE
X. Chen [16]	parking prediction	events, distance, price	ARIMA, linear regression, SVM, feed forward neural network	MAPE
Hössinger [17]	parking prediction	occupancy rate	linear regression, average	RMSE
Caicedo [18]	parking prediction	capacity, price, occupancy	probabilistic models	average error
Nandugudi [19]	parking prediction	parking times	probabilistic model	absolute error
Szczurek [20]	parking prediction	parking times	MALENA algorithm, Naive Bayes	average discovery time improvement
Ji [21]	parking prediction	parking times	wavelet neural network	MSE
Koster [22]	parking recommendation	parking times	Markov chains	Pearson coefficient
Kotb [23]	-	-	Mixed-integer Linear Programming	-
Pullola [24]	parking prediction	parking times	Poisson process	absolute error
Richter [25]	parking prediction	see SF*park* [28]	time series model	absolute error

Table 3. Continuation of Table 2

Paper	Approach	Features	Methods	Evaluation
Shin [26]	parking prediction	driving duration, distance, walking distance, parking cost, traffic	heuristic	average driving duration, average walking distance, parking fail rate, parking utilization rate, average occupancy
Zheng [27]	parking prediction	parking occupancy	regression trees, neural network, SVM	MSE, R^2

algorithms.

Chen et al. [14] develop an Android application that finds a parking location at park-and-ride facilities by calculating the probability of parking availability and taking in consideration the shortest travel time. The authors employ fuzzy logic to model the uncertainty of parking availability. The fuzzy membership function was chosen to be linear. The authors proposed multiple criteria in finding the best parking location, such as train frequency, service quality, parking-and-ride price, which should be considered in further investigations. The use case and parking data were set in Perth, Australia.

Vlahogianni et al. [13] define prediction measures to find out the duration of free parking spaces. Additionally, they calculate 1-hour parking availability forecasts. The authors use neural networks for time-series in the form of a multilayer perceptrons to accurately predict occupancy up to one hour ahead. Their findings show a Weibull distribution for the duration of free parking spaces. The system developed was incorporated into the routing service of the city of Santander, Spain.

Rajabioun and Ioannou [10] introduce an information system for parking guidance that enables communication between vehicles and the infrastructure. It proposes a prediction algorithm that forecasts the availability for parking locations based on real-time parking information. It takes into account parameters such as parking duration, arrival time, destination, pricing, walking distance, parking capacity, rates of vehicles occupying and leaving parking spots, time restrictions, parking rules, events that disrupt parking availability, etc. Their algorithm uses a probabilistic density distribution model. The parking data was collected both from on-street parking meters and off-street garages in Los Angeles and San Francisco, USA. In a following paper, Rajabioun and Ioannou [11] propose a multivariate autoregressive model that considers the temporal and spatial correlations of parking availability when making predictions (cf. Figure 2). The authors hold that the model, which is integrated into a parking guidance and information system, recommends parking locations with high accuracy.

Tiedemann et al. [9] present the development of a prediction system that gives estimated occupancies for parking spaces. The occupancy data is collected online via roadside parking sensors and the prediction is realized using neural gas machine learning combined with data threads. The authors notice that some factors play

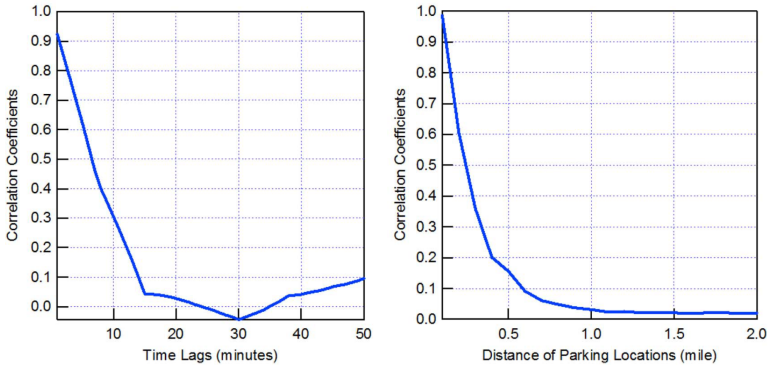

Fig. 2. Left: Temporal correlation for parking occupancy Right: Spatial correlation for parking occupancy. Data collected from on-street locations in San Francisco

a significant part in the predictions, such as holidays, weather and use the neural gas clustering to separate the data, before the data thread method is applied (cf. Figure 3). The system is developed by the DFKI Robotics Innovation Center, while the sensors are manufactured by Siemens AG and are installed in street lights. The pilot region for the project is Berlin, Germany.

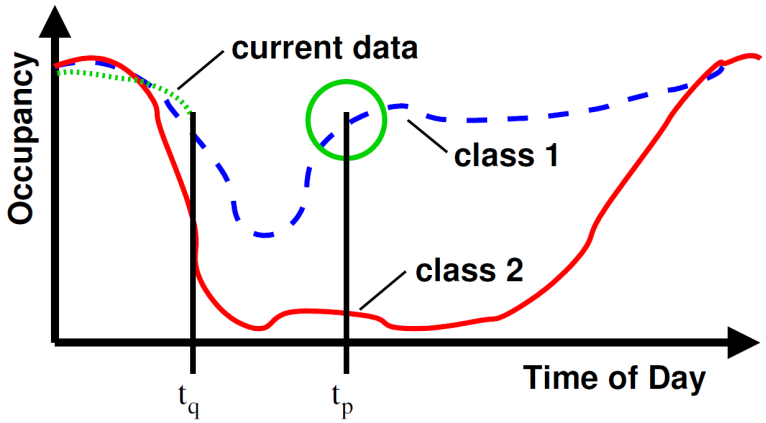

Fig. 3. Visualization of data threads that are used for predicting occupancy. Class 1 (dashed blue curve) and Class 2 (solid red curve) are two learned time series. When the user queries for the occupancy prediction at t_q for time t_p, the curve that is closest to the current data curve (dotted green) up to that point is chosen

Wu et al. [8] present a information system that recommends parking options in the city, in order to reduce the vehicle emissions and decrease traffic. The system constructs an optimal parking sequence based on the forecasted parking availability. For the latter, probabilistic algorithms are used, i.e., shift prediction, thread esti-

mation and probability estimation. The data and experimental results have been carried out in Taipei City in 2011.

Caliskan et al. [6] model the prediction of available parking spaces as a vehicular ad-hoc network (VANET). The network disseminates parking data in order to help with the estimation of future occupancy of parking lots. The pieces of disseminated data are timestamp, total capacity of parking lot, number of parking spaces that are currently occupied, the arrival rate, and the parking rate. The latter two are used in the modeling of continuous-time homogeneous Markov chains. The approach is otherwise based on queuing theory. The evaluation was carried out in Brunswick, Germany, where about 10000 vehicles were connected in the ad-hoc network. Klappenecker et al. [7] builds on the result of Caliskan and uses an improved version of continuous-time Markov chains for predicting availability of parking spaces. Predictions are communicated between cars in an ad-hoc network. The approach simplifies the computations of transitional probabilities inside a Markov chain model. The system applies to parking lots that are connected to the ad-hoc network. These communicate the number of occupied spaces, capacity, arrival and parking rate. Also based on VANETs is Szczurek et al. [20] work, which propose a novel approach that combines machine learning with the information disseminated in ad-hoc vehicular networks. The building block of the system are parking reports, which are issued by vehicles leaving a parking space and comprise a report identifier, a location, and a timestamp. The parking reports are being learned by a model, which then indicates whether a parking is available for a specific vehicle. A conditional relevance is used to determine whether a particular report is useful for a specific vehicle. This is modeled using a Naive Bayes method. A parking availability report R is labeled relevant by vehicle V, if the parking space referenced in R is available when V reaches it. Upon evaluation of the methods, the authors reported an improvement in parking discovery times for vehicles.

PocketParker is a crowdsourcing system, proposed by Nandugudi et al. [19], that uses smartphone data to predict parking availability. The system is used for parking lots. It requires no input from the user, it notices automatically when a user starts to drive or stops, i.e., departure and arrival events (cf. Figure 4). Based on these two events, the system builds a probability distribution model that is used to answer queries about parking availability. PocketParker has proved robust to hidden parkers, i.e., parking vehicles that are not using the application. In the authors' simulation, it has reached 94% rate for parking availability prediction with 105 users over 45 days.

Caicedo et al. [18] develop a methodology for predicting real-time parking space availability. The probabilistic algorithm consists of three subroutines: allocating simulated parking requests, estimating future departures, and forecasting parking availability. The forecast has been reported to improve as the system registers arrivals and departures. Further factors taken into account were duration of stay and capacity of every operating parking facility. The system was tested in Barcelona, Spain, with very satisfactory results.

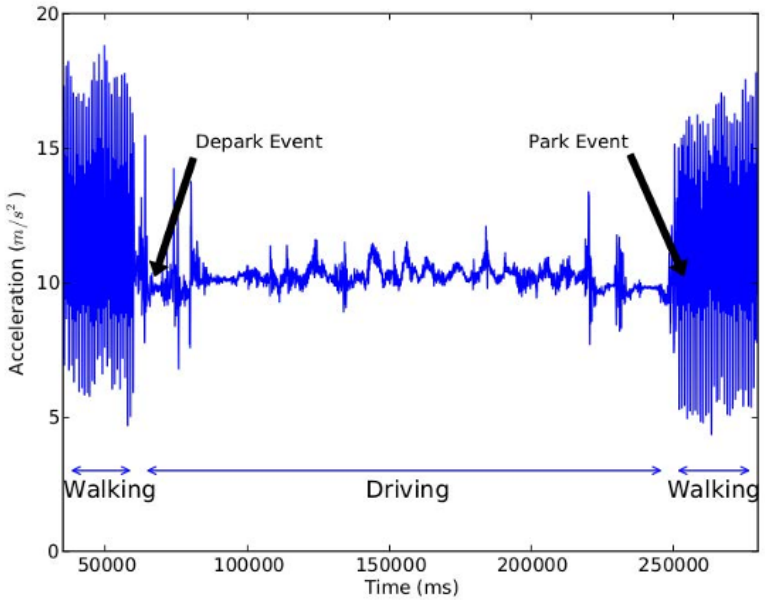

Fig. 4. PocketParker detects a parking event using accelerometer data from the driver's smartphone

Hössinger et al. [17] present a simple real-time occupancy model based on various pieces of data collected in the city of Vienna, Austria. An average day curve model was built using the ticket data from mobile-phone parking, the counts of car parks, and the traffic flow volumes in the city. The data was collected following agreements with the respective mobile phone companies, through surveys and by accessing a dedicated traffic website, respectively. The predictions are valid for short-time spans and applicable to the above mentioned city.

Chen [16] tackles the parking problem by aggregating parking lots. The paper shows that the prediction error of parking occupancy decreases by combining multiple parking lots. The trained models take into account factors such as day, time, event, distance, parking price, etc. The author tries out multiple models, such as ARIMA, linear regression, support vector regression and feed forward neural network. It turns out that the neural networks algorithm scores the best when the model is evaluated on the SF*park* data [29]. Furthermore, the paper investigates the specifics of aggregated parking lots (cf. Figure 5) and finds certain artifacts in the occupancy graph when clustering the respective parking locations.

Richter et al. [25] address the parking prediction problem with the focus on model storage in vehicles. The authors train models of various granularity that would predict parking availability based on the information contained: a one-day model per road segment, a three-day model per road segment, and a seven-day model per road segment. Additionally, models based on regions and time intervals computed by

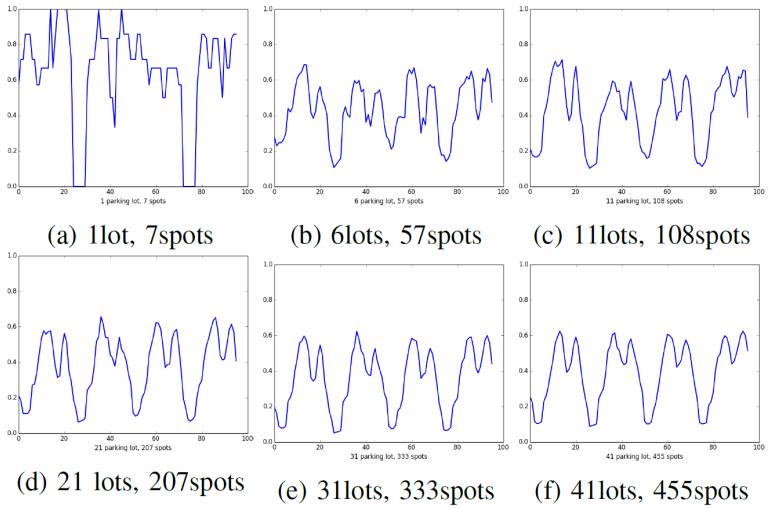

Fig. 5. Increasing aggregation levels of parking lots. The more parking spaces are included, the smoother is the emerging pattern

clustering are tried out. Hierarchical clustering with complete linkage is employed. The models are evaluated on street data from the SF*park* project [29]. The application of clustering before building the models shows a 99% decrease of model storage space. The prediction success rate is at about 70%.

With iParker, Kotb et al. [23] propose a system that handles parking reservations. It achieves resource allocation so that drivers will pay less for parking, while parking managers receive more resource utilization and hence reach higher revenue. The system is based on mixed-linear programming (MILP). The system uses dynamic resource allocation and pricing models to achieve its goal. At its evaluation, it was reported to cut the cost for drivers by 28%, achieving a 21% increase in resource utilization, and it increased the total revenue for parking management by 16%. Although this work only indirectly deals with parking occupancy, it makes use of it in a system that could include prediction models in the future.

ParkNet, developed by Mathur et al. [30] is a system made up of vehicles that captures parking space information while driving. Every ParkNet vehicle is equipped with a GPS receiver and an ultrasonic sensor facing sideways (cf. Figure 6). The latter determines whether it passes by parking spaces and whether they are occupied. The data is sent to a central server that aggregates it, in order to build parking space occupancy maps in real-time. The information is queried by clients that search for a free parking space. The system was evaluated in Highland Park, New Jersey and San Francisco on 500 miles road-side parking data and yield 95% accurate parking maps and 90% parking occupancy accuracy. The authors show that the system can further be improved if the sensors are fitted into taxicabs or city buses.

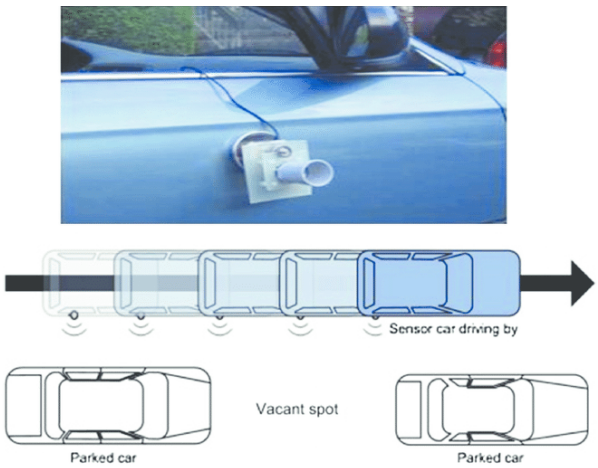

Fig. 6. The ultrasonic sensor mounted sideways on cars in the ParkNet system

Ji et al. [21] presents a forecasting method based on wavelet neural networks. These are feed-forward neural networks that have continuous wavelet function as activation function for the hidden layers. The system was tested in Newcastle upon Tyne, England and evaluation was performed using real-time data of up to 1 day in advance.

Koster et al. [22] propose a smartphone-based solution that recognizes when drivers arrive or leave parking spaces. A "Bayesian approach" and Hidden Markov Models (HMM) are used to model the parking spaces and respond to user queries for the next parking space. The HMM are based on gathered historical data. As answer to the user query is a parking space nearby and the probability of it being free at the respective time. The authors emphasize the non-intrusive nature of the solution, where drivers only have to minimally interact with their phones to get a recommended free parking space.

Pullola et al. [24] propose a solution to determine the availability of parking lots by modeling the occupancy as a non-homogeneous Poisson process. Past occupancy information of the parking lots is stored and leveraged. The data and the computation are made inside the GPS system of the vehicle, in order not to depend on the quality of the transmitted signal.

Zheng et al. [27] investigates the results of a proposed prediction mechanism on the parking data collected in San Francisco and Melbourne. The three algorithms proposed employ regression trees, support vector regression and neural networks. The feature sets included the historical occupancies alongside the time and day of the week.

Shin and Jun [26] propose an algorithm for smart parking that assigns cars to parking facilities in the city. The criteria based on which the assigning is realized includes driving distance to the parking facility, walking distance from the park-

ing facility to the destination, parking cost, and traffic congestion. The real-time data is collected from parking facilities and from sensors that are integrated in cruising cars. The data is transferred from the central server, where it is managed through a wired/wireless telecommunication network. The authors test their approach in Luxembourg City. The results of the simulations showed improved figures for average driving duration, average walking distance, parking failing rate, parking utilization rate, average standard deviation on the number of guided cars to each parking facility, average occupancy ratio of parking facility, and for the parking facility occupancy rate.

While all the presented approaches tackle the problem of improving the parking situation in cities with different ideas and implementations, they fundamentally differ from our work. Some approaches match with our idea of adding additional information about the current circumstances in a city or by building models for smaller areas. However, none of the approaches focuses on building a solution that can be ported to other cities. As opposed to this, our presented approach uses a public pre-defined ontology and data in RDF format as input in order to calculate the prediction models and similarity measures. Further, none of the presented approaches aims at training prediction models for small areas where data are available and transfer those models to other areas where no data is available. Our approach aims exactly at this goal for enabling a more accurate prediction in districts or streets where no parking data are available.

4. Selected Data Source

To implement and evaluate our idea, we use the data from the SF*park* project. This project was realized by the San Francisco Municipal Transportation Agency (SFMTA), the city agency that manages the city's transportation, which includes on-street parking [12] [28]. The SFMTA establishes parking rates for on-street parking meters. Before the project started, parking rates were the same all day, every day, independent of the parking demand. By implementing a demand responsive pricing scheme, parking availability improved dramatically. The facts listed in Table **??** show the improvements on parking after the project completed.

In conducting the project, nine pilot areas were chosen for monitoring. Out of these areas, seven were selected to have new pricing policies, while two were control areas. The number of metered spaces used was 6000, which amounts for 25% of the city's total. The meters allow rates to be deployed remotely and they transmit data to a central server through a wireless connection.

The data was collected using parking sensors. These provided the central server with the information needed to calculate the demand-responsive parking rates and provided real-time parking availability information. A parking sensor is a magnetometer that detects changes in the earth's electromagnetic field. A total of 11700 sensors were deployed, resulting in 8000 spaces that were equipped with one or two sensors. The sensors delivered valid data from April, 2011 to December, 2013. The

sensors can suffer from environmental noise, such as electromagnetic interference, early battery degradation or street construction.

SF*park* made available real-time information on parking rates and parking occupancy through a smart phone application. The SF*park* project and its success played an important role when choosing to base our project on it.

5. Concept

For implementing our approach, we extract parking data from SF*park*. In addition, we collect city data from various sources (cf. Section 5.2). To ensure that our approach will be compatible with data available in other cities, we annotate our data using ontologies from the CityPulse project [31] and store them in RDF format. Afterwards, we merge the data sets and perform a clustering process. Next, we define the parking profile by introducing two similarity functions: *cosine similarity* and *earth mover's distance*. Finally, we explain the machine learning training process and define the estimations for clusters without parking data.

5.1. *Parking Data*

We consider the following types of data as parking data: *parking occupancy* contains information on the availability of parking spaces; *traffic data* contains information regarding the city traffic, which is relevant for parking; *weather data* contains weather information for the same area as for the parking problem; *event data* contains event information which may have an impact on parking; *parking revenue data* contains economic information on parking, whose relevance may influence parking prediction. *fuel price data* contains prices of fuel in the region for which we build the models. Each piece of data is geographically referenced by a *location unit*, e.g., street block, district or city. An overview of the different properties available in the data set are shown in Table 4.

5.2. *City Data*

We established that city data reflects parking demand in a city area. We obtain it by collecting public amenity information as provided by OpenSteetMap[a]. The OSM data is available as shapefiles containing geometries such as points, polylines and polygons. We extract the *points of interest* (POIs). A POI contains an *amenity* attribute indicating the public service located at this position as it was annotated by the OSM users (cf. Figure 7).

The amenity information accounts for the times of parking (i.e., morning, day, night, etc.) and, up to some degree, for the duration of parking. An example of a service indicating this is Google Maps. It displays typical *visiting duration* or *time*

[a]`https://www.openstreetmap.org` The maps used in this article are ©OpenStreetMap contributors

Table 4. An overview of the properties available in the data used from the SFpark project

Parking Occupancy		Traffic	
date and time	Recorded usually at full hours or in periodic time intervals	date and time	recorded usually at full hours or in periodic time intervals
parking capacity	The total number of parking spaces at the given location	traffic value	typically expressed as average traffic road occupancy, average vehicle count, median speed, or average speed of the traveling cars
parking price	The price of a ticket at the certain location and the given time in a given currency		
parking occupancy	Expressed either as rate (sub-unitary fraction or percent) or in absolute numbers		
Events		**Weather**	
event name class	the name of the event and its class (road closure, rise of parking demand)	temperature	may be current temperatures or maximum and minimum values per day
		precipitation	expressing the quantity of rain or snow for the corresponding time interval
Fuel Price		**Parking revenue**	
type of fuel	gasoline, diesel, etc.	payment type	the way the driver opted to pay for parking, e.g., cash, credit card, etc.
price per unit	provided as the price per liter or per gallon.	payed amount	the amount in US dollars, Euro or other currency

Note: Each of these data types also has the location unit id, as well as the date and time (interval) when it occurred or was measured. In some cases, the location is a somewhat larger or smaller area as the unit. The time information is provided in different granularities (e.g., per minute, per hour, per day, etc.)

spent values and popularity of the place for specific points in time. The average values are based on the users' smart phone GPS sensors (cf. Figure 8). To obtain the duration values, we manually extract information from Google Maps. The duration information is aggregated by Google using a crowdsourcing approach.

5.3. *RDF Annotation*

We emphasized that our approach should be relevant for any city that wants to solve its parking problem. Thus, all input data used in the present approach is in RDF format, in order to establish a common format that can be used for other cases. Since the actual parking and city data is only available as raw values, we need to annotate it as RDF in the first place. The default process involves the extraction of data, which is afterwards available for further processing. To annotate the city and parking data, we use Apache Jena. As underlying ontologies, we reuse those created

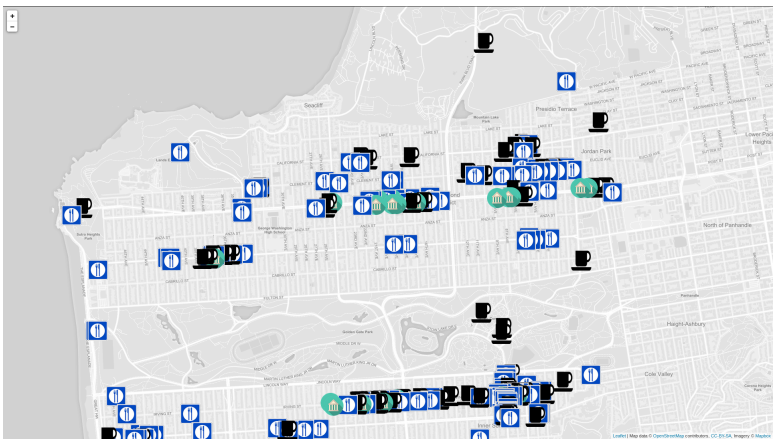

Fig. 7. A map indicating public amenities (cafes, restaurants, banks) found at points of interest in OSM.

Fig. 8. An example of *visiting duration* information found on Google Maps.

as part of the CityPulse project [31]. Afterwards, we can easily extract the data in a well-defined format using SPARQL queries.

5.4. *Merging City and Park Data*

In order to combine the parking and city data, both sets of data require a common location unit. The parking location units are provided together alongside the various types of parking data. The city data, on the other hand, references POI geometries, which are points expressed in a particular reference system, which differs from the one of the parking locations. Therefore, after establishing the coordinate systems of both geometries, we define a *merge distance* that matches a parking space to a public amenity.

The merge distance can be intuitively understood as the radius around a public amenity. It is defined to represent the parking area that is relevant for a particular

public amenity, or, more straightforward, the walking distance from the parked car to, e.g., the restaurant, the office, the bank, etc. Concrete instances of the merging distance can be found in Section **??**.

5.5. *Clustering*

In a realistic scenario, the available parking data does not cover the whole city surface. In fact, it is a fraction of it. Therefore, we first separate the area *with* parking data from the area *without* parking data. Based on this initial split, we perform clustering to further separate these regions into smaller areas. By splitting into city areas, we are making sure that smaller regions lead to more representative parking profiles and therefore parking estimations.

As we want an exclusively location-based separation, we may employ K-Means, DBSCAN or OPTICS to cluster the city areas. The distance is calculated between (latitude, longitude)-pairs of location unit coordinates corresponding to one parking unit. There are two clustering processes executed, one for the city area *with* parking data, another one for the city area *without* parking data. The number of clusters chosen in each area is kept proportional to the number of total location units that each city area contains. Since having control over the number of clusters is the goal here, we choose to use K-Means, where we provide the number of expected clusters as input. More details on the concrete value of k and an overview about the clusters can be found in Section **??**.

5.6. *Similarity Functions*

To calculate the similarity of city areas, we use *cosine similarity* and *earth mover's distance*. For this, we use two representations of city areas: *cluster vectors* and *cluster Gaussians*. As data, we use the city data described in Section 5.2.

5.6.1. *Cosine Similarity*

To form the cluster vectors, we first divide all amenities into categories $Cat_1, Cat_2, ..., Cat_n$. The criteria for division will be their average *visiting duration* duration. For example, a short duration category of up to 30 minutes, a medium duration between 31 and 90 minutes and a large duration of above 90 minutes stay. Each cluster gets represented by an n-dimensional vector, whose components correspond to the amenity categories. The magnitude of component i is equal to the number of amenities of category Cat_i that can be found in that particular cluster. Compare Figure 9 for a general representation.

The cosine similarity between two vectors is defined as the cosine of the angle between the two vectors:

$$cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

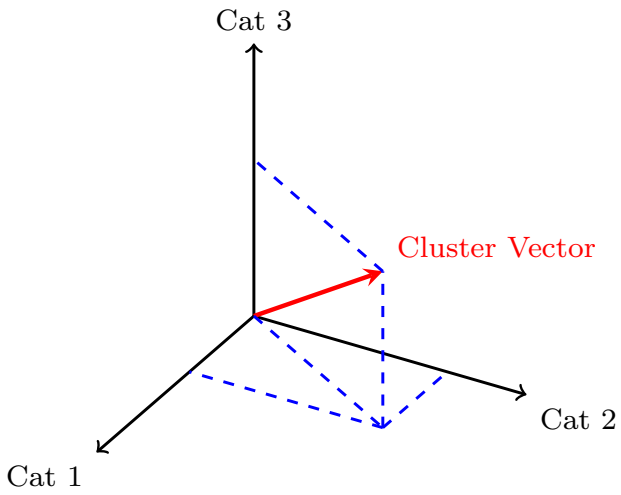

Fig. 9. An example of a cluster vector for three categories.

where A_i and B_i are the components of vector A, respective B.

Unlike the earth mover's distance, the cosine similarity implementation uses the direct mathematical formula by plugging in the magnitudes of the respective vector components.

5.6.2. *Earth Mover's Distance*

A cluster Gaussian is a *kernel density estimation* among amenity probability distributions. In turn, an amenity probability distribution is represented as *Gaussian kernel*. To construct a cluster Gaussian, we first collect the average visiting duration and standard deviation for the individual amenities. A cluster that contains one amenity A is represented as a Gaussian curve, i.e., normal probability distribution. The curve's center is at the average duration of the amenity A and its standard deviation is the one of the amenity. When n amenities A exist in the cluster, the representation will be an A curve multiplied n times. Multiple amenities, each appearing multiple times, will result in a curve that is the linear combination of the individual representations of the amenities as normal distribution curves. Compare Figure 10 for a visualization of the summing process.

$$emd(\mathcal{C}_i) = \sum_{j=1}^{|amenities|} K_{ij} \times A_j \tag{2}$$

$$\forall i \in \{1, ..|clusters|\} \text{ and } \forall j \in \{1, ..|amenities|\}$$

where A_j is an amenity that appears K_{ij} times in the cluster $\mathcal{C}_i$.

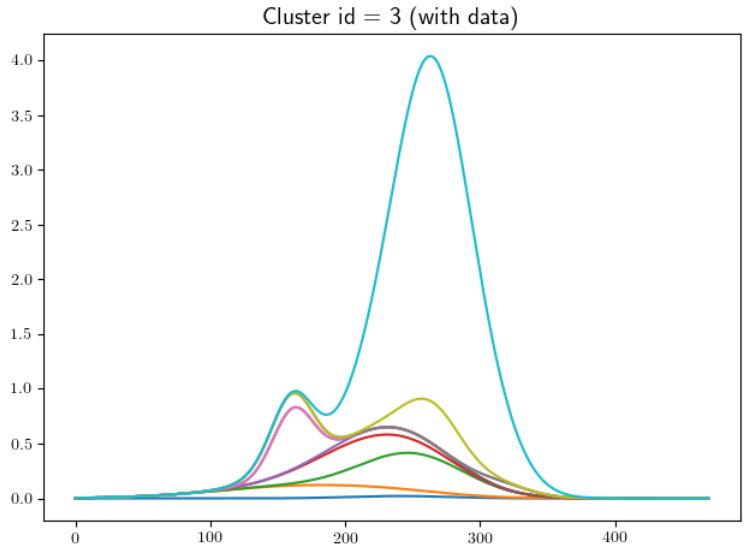

Fig. 10. The summing of Gaussians resulting in a cluster Gaussian.

The earth mover's distance (emd) is a measure used in statistics that roughly expresses the difference between position and magnitude of two curves. It is best explained by regarding the curves as the hull of earth piles. For two separate earth piles, emd computes the minimum effort of rearranging a pile so that the shape of the other pile is obtained. Moving P particles over a distance D is equal to the effort $P \times D$. A prerequisite for this operation is that the two piles need to contain the same quantity of earth.

More rigorously, the earth mover's distance is better known in mathematics under the name Wasserstein Metric. Given two normal distributions $\mu_1 = \mathcal{N}(m_1, C_1)$ and $\mu_2 = \mathcal{N}(m_2, C_2)$, where m_1 and $m_2 \in \mathbb{R}^n$ are their respective expected values and C_1 and $C_2 \in \mathbb{R}^{n \times n}$. Then, the 2-Wasserstein distance between μ_1 and μ_2 is:

$$W_2(\mu_1, \mu_2)^2 = \|m_1 - m_2\|_2^2 + trace(C_1 + C_2 - 2(C_2^{1/2} C_1 C_2^{1/2})^{1/2}) \tag{3}$$

In practice, we will not apply the Wasserstein metric directly, but rather resort to some levels of discretization. First off, a number of so-called bins is determined. Each bin represents a unit on the X axis, the same on which the visiting duration is expressed. We will take a number of buckets equal to the maximum amenity mean and add 3× the largest standard deviation to it, as it is known that within 3× standard deviation on both sides of the mean over 99% of the Gaussian sum is covered. Moreover, an offset on the X-axis equal to 3× the maximum standard deviation is used. This way, we are sure the landscape of summed Gaussians will easily fit into the number of bins.

Notice that emd is applicable only when the sum under both Gaussian curves is equal. Therefore, all cluster Gaussians will get normalized before emd is computed.

5.7. *Machine Learning Models*

The prediction of parking occupancy is realized using machine learning. We choose to explore this methodology, following the solid results machine learning models have delivered for the various smart parking settings investigated in Section 3. A machine learning model $\mathcal{M}$ will be trained for every cluster *with* parking data.

The training data *features* are composed of the parking data previously enumerated. We aggregate feature values around the location unit id, as on the cluster level this is irrelevant. The occupancy rate is set as the target variable. The model training and evaluation is performed in Python via the *scikit-learn* library.

During the training phase, we evaluate different machine learning approaches ranging from simple decision trees over random forests to support vector machines and gradient boosting. In addition, we make use of grid search to determine the best model parameters for the current approach. As error metric, we use root mean square error and performed a ten-fold cross-validation. Furthermore, a model is evaluated on the other clusters with parking occupancy data.

5.8. *Parking Occupancy Estimations*

Once all models $\mathcal{M}$ have been built for the clusters *with* parking data, making estimations on parking occupancy in these areas is straightforward. However, we want to apply these models, on the clusters that are missing parking data. We derive the *estimation interval* for cluster $\mathcal{C}^j_{wout}$ based on the model of cluster $\mathcal{C}^i_{with}$ as follows.

For cosine similarity:

$$E(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}) = [\mathcal{M}(\mathcal{C}^i_{with}) - (1 - sim_{ij}),\ \mathcal{M}(\mathcal{C}^i_{with}) + (1 - sim_{ij})] \tag{4}$$

$$\text{where } sim_{ij} = sim(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}) \in [0, 1]$$

For earth mover's distance:

$$E(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}) = [\mathcal{M}(\mathcal{C}^i_{with}) - emd_{ij},\ \mathcal{M}(\mathcal{C}^i_{with}) + emd_{ij}] \tag{5}$$

$$\text{where } emd_{ij} = emd(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}) \in [0, 1]$$

$$\forall i \in \{0, ..., |\mathcal{C}_{with}| - 1\} \text{ and } \forall j \in \{0, ..., |\mathcal{C}_{wout}| - 1\}$$

.

X is a parking data record containing feature values. The result is an *estimation interval* that "stretches" the punctual estimation into an interval depending on

the similarity value. The lower the similarity value is, the larger the length of the resulting estimation interval will be.

Notice that X should be valid for both $\mathcal{C}^i_{with}$ and $\mathcal{C}^j_{wout}$. The averaged values used for features related to the location unit id will be transferred as they are similar to those of the target cluster $\mathcal{C}^j_{wout}$.

Furthermore, we define an *estimation intersection interval*, whose purpose is to narrow down the computed estimation interval. An estimation intersection interval for the clusters $\mathcal{C}^i_{with}$ and $\mathcal{C}^j_{wout}$ is computed by intersecting the *estimation intervals* that have a better similarity among the clusters with data $\mathcal{C}^0_{with}, ..., \mathcal{C}^{i-1}_{with}$ and the same cluster without data $\mathcal{C}^j_{wout}$.

$$EII(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}) = \bigcap_{k=0}^{i-1} EI(\mathcal{C}^k_{with}, \mathcal{C}^j_{wout}) \tag{6}$$

where

$$sim(\mathcal{C}^k_{with}, \mathcal{C}^j_{wout}) < sim(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}), k \in \{0, .., i-1\} \text{ for emd} \tag{7}$$

$$sim(\mathcal{C}^k_{with}, \mathcal{C}^j_{wout}) > sim(\mathcal{C}^i_{with}, \mathcal{C}^j_{wout}), k \in \{0, .., i-1\} \text{ for cosine} \tag{8}$$

$$\forall i \in \{0, ..., |\mathcal{C}_{with}| - 1\} \text{ and } \forall j \in \{0, ..., |\mathcal{C}_{wout}| - 1\}$$

.

6. Experimental Setup

In the following, we describe the pre-evaluation setup, consisting of practical considerations on the following: parking- and city data, clustering, similarity functions, model training and testing, as well as data aggregation. The first elements needed for evaluation are parking and city data. As parking data, we use those from the SF*park* project and determine the data sets that are relevant for our purposes. As city data, we take OpenStreetMap data from San Francisco and obtain the available public amenity information. Google Places provides the visiting duration values for the amenities. Based on these data sets, we determine relevant values for the *merge distance*. Since we cannot evaluate the performance of our approach using clusters *without* data, we evaluated the performance on clusters for which occupancy data were available. Therefore, we first calculated the similarity score between every pair of clusters $(\mathcal{C}_{source}, \mathcal{C}_{target})$, after which the estimation errors were computed by applying the model of cluster $\mathcal{C}_{source}$ to $\mathcal{C}_{target}$, for all such pairs of clusters.

6.1. *SFpark parking data*

The SF*park* data are visualized in Figure 11 using a built Leaflet application. The actual SF*park* data has some particularities. While the *occupancy data* is provided

with reference to *blocks* as location units, all the other data sets use different location units. For the traffic and events data sets, the location units are street names. For parking revenue, they are districts. In case of weather and fuel price, the location reference is valid for the whole city of San Francisco.

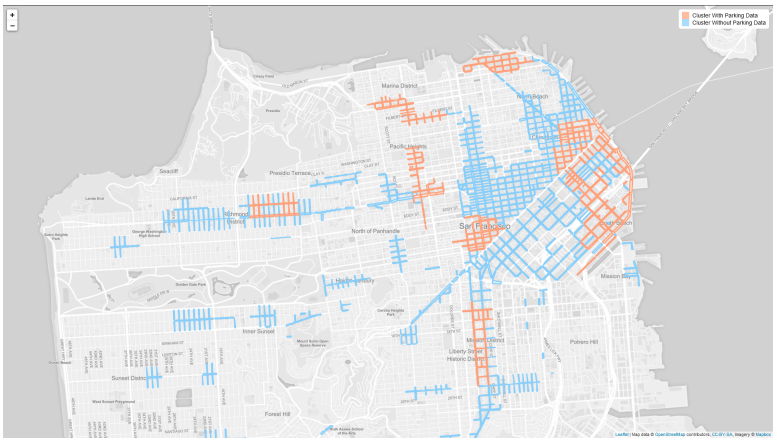

Fig. 11. The blocks accounted in SF*park*. The light blue ones are blocks *without* parking data, the light red ones are *with* parking data.

6.2. *OpenStreetMap for San Francisco*

Following the selection of SF*park* data as parking data, the city data is found in the corresponding OpenStreetMap layer for San Francisco. The actual public amenity information collected from POIs is listed in Table 5.

Table 5. List of all OSM amenities found in the SF*park* blocks.

arts_centre	dojo	marketplace	shelter	conference_centre
bank	embassy	music_rehearsal_place	shop	fire_station
bar	fast_food	music_school	spa	fuel
biergarten	grocery	nightclub	stripclub	parking
bureau_de_change	gym	pet_grooming_shop	studio	place_of_worship
cafe	hookah_lounge	pharmacy	training	social_centre
clinic	ice_cream	police	veterinary	swimming_pool
clothes_store	karaoke	post_office	vintage_and_modern_resale	theatre
community_centre	lan_gaming_centre	pub	bus_station	training
dentist	laundry	restaurant	car_rental	bicycle_parking
doctors	library	salon	childcare	car_wash
brokarage	community_centre	courthouse	fountain	nursing_home
recycling	social_facility	toilets		

6.3. *Merging parking and city data*

In case of SF*park*, the blocks are given in latitude and longitude. In OpenStreetMap, the geometry is set to EPSG 4326. With both systems using the same reference, we can therefore fix a *merge distance*. The distance d should express the impact that a POI P has on the block B, when $dist(P, B) = d$. For instance, it expresses the impact of the parking demand that a restaurant induces on a parking block situated at d meters away. Further in the evaluation we assign it distances of 100m, 200m, and 400m.

6.4. *Clustering*

As established in Section 5.5, we apply K-Means to cluster the city areas. In the evaluation, we will refer to the number of clusters *with* parking data as *the number of clusters*. The area without parking data is going to be split into a proportional number of clusters, as the sizes of clusters should be kept roughly equal for both sides. It turns out that for SF*park* the proportion is approximately 2.6, following the division between the total number of blocks from each group. We have chosen two numbers of clusters to run the evaluation, namely 8 clusters and 16 clusters. The area without parking data will therefore have 20 and 41 clusters, respectively.

After running the K-Means clustering process, the Leaflet application map reveals the individual clusters by highlighting them on mouse-over. The clusters *with* parking data will turn dark red, while the clusters *without* parking data will appear in dark blue (cf. Figure 12).

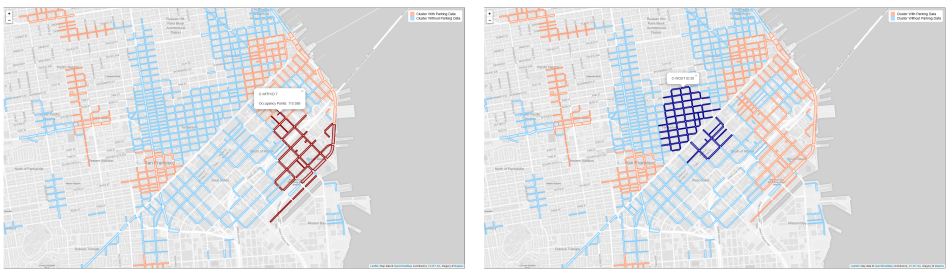

Fig. 12. Highlighted cluster *with* parking data on the left side and a cluster *without* parking data on the right side.

The resulting aggregated blocks are worth taking into account when training the machine learning models, as these will average over pieces of information contained in individual blocks.

6.5. *Similarity functions*

Before computing *cluster vectors* and *cluster Gaussians*, we will establish the visiting duration in every amenity. For this, we use information gathered from Google

Places available via Google Maps[b].

We manually collected information from 470 places in San Francisco, for which a maximum duration of stay was provided (the minimum duration is not always given, as indicated earlier). The data was obtained by manually navigating to every business place and writing the duration visit information in a spreadsheet. This piece of information is not accessible yet via the Google Places API[c]. The results are shown in Table 6 and the numbers are given in minutes and have been rounded to the nearest integer. We have included only amenities for which at least two stay duration sources were found.

Table 6. All amenities listed with their corresponding mean visiting information as collected from Google Places

amenity name	**mean**	**stdev**	**cat**	**amenity name**	**mean**	**stdev**	**cat**
arts_centre	110	37	3	laundry	78	16	2
bank	42	65	2	library	83	13	2
bar	121	38	3	music_school	120	30	3
cafe	76	39	2	nightclub	189	20	3
clinic	100	29	3	pharmacy	25	20	1
clothes_store	41	37	2	post_office	16	2	1
community_centre	119	40	3	pub	135	21	3
dentist	104	35	3	restaurant	135	32	3
doctors	60	42	2	salon	141	53	3
embassy	75	24	2	shelter	90	0	2
fast_food	31	15	2	shop	43	21	2
grocery	20	10	1	spa	161	54	3
gym	100	22	3	stripclub	140	46	3
hookah_lounge	130	17	3	studio	60	0	2
ice_cream	23	7	1	veterinary	67	29	2
karaoke	188	15	3	vintage_modern_resale	38	32	2

Note: Duration and standard deviation are expressed in minutes. The assigned category for cluster vectors is included.

Alongside this information, we need the amenity categories in order to derive the *cluster vectors*. As defined in Section 5.6.1, the categories are based on the visiting duration mean. We split them in three categories:

(i) under half an hour,
(ii) 31 to 90 minutes and
(iii) more than 1.5 hours.

The assigned partitions for every amenity are shown in a separator column in Table 6.

The computation of *cluster Gaussians* relies on both the mean and standard deviation of the amenity visiting duration, as defined in Section 5.6.2.

[b]http://maps.google.com
[c]Google Feature Request: `https://issuetracker.google.com/issues/35827350`

6.6. *Amenity area as similarity basis*

Together with the similarity functions in Section 6.5 we considered *visiting duration* as the basis for creating the urban profile. However, there are other measures that reflect the parking demand towards an amenity, one of them being the area of the particular amenity. OpenStreeMap provides a polygon layer for a certain geographic bounding box, which contains information across all the surfaces in that region (recall Section 5.2). See Figure 13 for a visualization of the polygons and their areas in OSM. In extracting this information we had two options.

Fig. 13. OSM screenshot emphasizing polygons as buildings and the amenities that are housed by them

Matching the amenities' POIs with the containing OSM polygon and then computing the polygon areas per amenity was the option tried first. This has several drawbacks. The relation POI:polygon is in practice by no means 1:1, i.e. many cases arose where multiple POIs were contained by the same polygon, in which situation the area was split between them; a POI might also be on the edge of several polygons, in which case we have to either (arbitrarily) assign it to the first polygon or to all. The deciding factor against this option was, however, the high *coefficient of variation* for the area values.

The coefficient of variation (CV) is a measure of dispersion of a probability distribution and is defined as the ratio between the standard deviation and the mean of a sample (cf. Equation 9). When calculating CV based on the matching of POI with the polygons, it came out 2.1 (value is referenced below).

$$c_v = \frac{\sigma}{\mu} \tag{9}$$

The other option was to use the *amenity* attribute in the polygon layer for the region. We could avoid the cumbersome matching by leveraging solely the polygon

layer and calculating the amenity area mean and its standard deviation. The results are listed in Table 7. On top of that, the coefficient of variation is 0.9 in this case, significantly lower than before.

Note that we have reduced the values in the table by a factor of 20, as it turned out that the actual mean and standard deviation were large enough to make the emd Gaussian computation extremely slow. As the standard deviation is linear with regard to the mean, both mean and standard deviation values were reduced conveniently.

Table 7. Amenity area values gathered and averaged from OMS polygon layer for the SF*park* region

amenity name	**mean**	**stdev**	**cat**	**amenity name**	**mean**	**stdev**	**cat**
arts_centre	68	60	2	bank	39	20	2
bar	19	8	1	bicycle_parking	8	7	1
biergarten	11	12	1	brokerage	39	9	2
bus_station	588	737	3	cafe	17	10	1
car_rental	70	43	2	car_wash	43	48	2
childcare	101	130	3	cinema	75	43	2
clinic	61	32	2	community_centre	52	74	2
conference_centre	401	519	3	courthouse	459	201	3
dentist	17	12	1	doctors	324	568	3
embassy	68	38	2	fast_food	25	24	1
fire_station	52	27	2	fountain	24	22	1
fuel	25	27	1	library	102	124	3
marketplace	325	228	3	music_rehearsal_place	33	15	1
nightclub	32	9	1	nursing_home	97	47	2
parking	182	309	3	pharmacy	65	38	2
place_of_worship	60	62	2	police	137	124	3
post_office	39	11	2	pub	25	25	1
public_building	280	236	3	recycling	28	20	1
restaurant	22	16	1	school	740	1280	3
social_centre	30	21	1	social_facility	356	801	3
stripclub	50	10	2	studio	268	307	3
swimming_pool	16	9	1	swingerclub	27	4	1
theatre	174	191	3	toilets	7	5	1
training	72	94	2	veterinary	21	7	1

Note: The mean and standard deviation values were reduced by a 20x factor. The categories for cosine vectors are 0 - 35, 36 - 100, 100+.

6.7. *Model training and evaluation for clusters* with *parking data*

As training data, the SF*park occupancy data* is used with *street blocks* as location unit. It turns out that training on the additional SF*park* data, i.e., traffic and events, encounters some problems.

The *traffic data* do not share the same location unit with the parking occupancy's street block. Even when aggregating traffic data on the district level, which is available for the occupancy data as well, it does not prove an additional value to the training.

For the SF*park events data* we encounter the same problem as for the traffic data: the location unit does not match the block. In fact, the events are marked for streets, whose association to blocks is not determinable.

Parking revenue data is provided for districts, which again are too general to make a difference in training.

Finally, *weather data* and *fuel data* are given per city, hence making even less an impact to improve the model.

6.8. *Aggregating training data*

As indicated in Section 5.7, the training data is aggregated across all blocks so that it becomes comparable to other clusters and can be used in training and testing models. The averaging is performed per timestamp, i.e., if multiple blocks have an occupancy record for the same time and block, the occupancy rate will be averaged for both of these. Features such as *price* and *parking capacity per block* are averaged as well. See Table 8 for an example of this process. This means that the original collection of data records shrinks, which should decrease the training time. In Table 9, the expansion/shrinking rate is shown for various number of clusters.

Table 8. Example of aggregating datapoints

block id	**timestamp**	**price rate**	**total spots**	**occupied**
902	2011-04-02 7:00:00	0	46	58
32800	2011-04-02 7:00:00	0	32	2
33005	2011-04-02 7:00:00	3	36	12
902	2011-04-02 8:00:00	2	46	54
32800	2011-04-02 8:00:00	4	32	5
33005	2011-04-02 8:00:00	3	36	22

timestamp	**price rate**	**total spots**	**occupied**
2011-04-02 7:00:00	**1**	**34**	**26**
2011-04-02 8:00:00	**3**	**34**	**27**

Note: In the subtable above, three distinct blocks belonging to a cluster are transformed into two entries by averaging *price rate*, *total spots* and *occupied* attributes for the two distinct timestamps (subtable below)

Table 9. Number of datapoints aggregated per timestamp vs. all datapoints alongside the shrinking rate for 8, 16 and 32 clusters

cluster size	aggregated datapoints	all datapoints	shrinking/expansion rate
8	9741	128525	12.3
16	8409	73332	8.3
32	6257	29355	4.6

Note: Values have been averaged across clusters

In Figure 14 a screenshot of the web application showing a sample of the results is illustrated. Figure 15 displays the presented table in more detail.

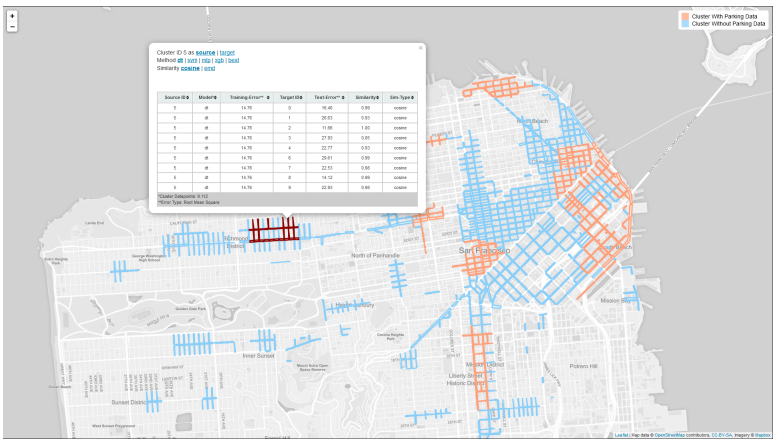

Fig. 14. Selected cluster *with* parking data and the pop-up table in the Leaflet application.

Source ID	Model*	Training-Error**	Target ID	Test-Error**	Similarity	Sim-Type
5	dt	14.76	2	11.66	1.00	cosine
5	dt	14.76	8	14.12	0.99	cosine
5	dt	14.76	0	16.40	0.99	cosine
5	dt	14.76	7	22.53	0.98	cosine
5	dt	14.76	4	22.77	0.93	cosine
5	dt	14.76	9	22.93	0.98	cosine
5	dt	14.76	1	26.63	0.93	cosine
5	dt	14.76	3	27.93	0.85	cosine
5	dt	14.76	6	29.61	0.99	cosine

*Cluster Datapoints: 9.112
**Error Type: Root Mean Square

Fig. 15. The pop-up table for the Leaflet application view of Figure 14.

7. Evaluation

We shall evaluate various pieces of the system that has been presented. Firstly, in 7.1, we shall establish the machine learning method that achieves best results on average across clusters.

Section 7.2 is the heart of the evaluation. It consists in comparing the cluster models' test error with the independently-computed similarity values between clusters. More specifically, a *source* cluster's model will tested on a *target* cluster and the error is correlated to the similarity between the *source* and the *target* clusters. Both *cosine* and *emd* functions will be used. The correlations will be expressed as Pearson- and Spearman's rank coefficients.

In the following Section 7.3, we take a look at the results of applying the models to clusters *without parking data* and show screenshots from the concrete web application. Section 7.4 looks at the model test errors and correlation results by skipping the aggregating step in machine learning training, i.e. instead of averaging

the datapoints over timestamp per cluster, we build the cluster models using the entire occupancy data.

Section 7.5 follows up on Section 6.6 and uses the *amenity area* as the basis for the similarity functions in calculating the correlations between model test errors and similarity values. Finally, in Section 7.6 we question whether the similarity function approach is the most efficient and transfer its purpose to the machine learning phrase. The model will receive absolute *cosine* and *emd Gaussian* values as additional features and its model test error and correlation values will be compared to the ones from the original approach.

7.1. *Best Model Method*

Models were trained using four methods: decision trees, support vector machines, multilayer perceptrons, gradient boosted trees. Section 10 shows the distribution of best machine learning methods in case of 8 and 16 clusters. The values were obtained by summing up the number of times a method produced the least estimation error, i.e., RMSE, among the four methods for all combinations of clusters with parking data $(\mathcal{C}_{source}, \mathcal{C}_{target})$. Extreme gradient boosting claims the first spot in both cases.

For further experiments in the evaluation we shall train the models exclusively with extreme gradient boosting.

Table 10. The proportion of best models among decision trees, support vector machine, multilayer perceptron and gradient boosting measured as RMSE when applied on all pairs of clusters

	dt	**svm**	**mlp**	**xgb**
8 clusters	24.6%	17.5%	12.3%	**45.6%**
16 clusters	14.6%	13.8%	13.8%	**57.9%**

7.2. *Similarity Values vs. Estimation Errors*

The central goal of this work is estimate the occupancy for clusters where no parking data is available using model predictions and pair-wise cluster similarity values. If the model prediction accuracy can be measured on clusters with parking data by means of their test errors, we still need to show that the cluster similarity values express indeed a similarity of occupancy. We shall use clusters with parking data to check this and use the model predictions as occupancy references: the higher the absolute correlation between the cluster similarity values and the model test errors, the better the accuracy of the cluster similarity. Concretely, for every pair of clusters $(\mathcal{C}_{source}, \mathcal{C}_{target})$ a model $\mathcal{M}_{\mathcal{C}_{source}}$ is trained on $\mathcal{C}_{source}$ and its test prediction error on $\mathcal{M}_{\mathcal{C}_{source}}(\mathcal{C}_{target})$ shall be correlated with the cosine and emd similarity values between $\mathcal{C}_{source}$ and $\mathcal{C}_{target}$. Here we use two correlation coefficients: the Pearson correlation coefficient and Spearman's rank correlation coefficient, resulting in four correlation measures:

(i) cosine (Pearson) correlation
(ii) cosine (Spearman's) rank correlation
(iii) emd (Pearson) correlation
(iv) emd (Spearman's) rank correlation.

The evaluation was performed for configurations of 8 and respectively 16 clusters. Additionally, we varied the *merge distance* to see how the correlation behaves. The similarity values are hence calculated for 100m, 200m, and 400m merge distance respectively. In Table 11 the final results are shown. For each correlation measure (*cosine*, *rank cosine*, *emd*, *rank emd*) the value is averaged across all clusters. Due to their mathematical meaning explained in 5.6, for cosine measure a negative correlation is optimal, whereas for emd a positive correlation value is beneficial. The models taken were trained with gradient boosted trees.

We notice that the cosine similarity achieves better results than emd for the same testing configuration, peaking for 8 clusters and 100m merge distance. Its average Pearson coefficient is -0.55, while the mean Spearman rank coefficient is -0.49. emd positively correlates the most for the same testing configuration (8 clusters, 100m), when the average Pearson coefficient is at 0.28 and Spearman's rank coefficient equals 0.23. There is a clear descending trend in correlations, as the merge distance increases. Also, the results for 8 clusters are superior to the ones when the city is split in 16 clusters.

Further in the evaluation runs we shall fix the merge distance to 100m.

Table 11. Correlations between similarity values and model estimations errors for pairs of clusters *with* parking data $(\mathcal{C}_{source}, \mathcal{C}_{target})$.

	8 clusters			
merge distance	cosine	rank_cosine	emd	rank_emd
100m	**-0.55**	**-0.49**	**0.28**	0.23
200m	-0.34	-0.30	0.26	0.23
400m	-0.23	-0.08	0.25	**0.27**

	16 clusters			
merge distance	cosine	rank_cosine	emd	rank_emd
100m	**-0.20**	**-0.17**	**0.10**	**0.11**
200m	-0.13	-0.11	0.02	0.02
400m	-0.17	-0.17	0.08	0.11

Note: For cosine similarity the values show a negative correlation tendency, while for the correlation based on emd similarity expresses a positive correlation tendency. The correlations are measured using Pearson coefficient and Spearman's rank coefficient.

7.3. *Estimations for Clusters without Parking Data*

We apply the models trained on SF*park* data on clusters *without* parking data. The testing data records are composed of values equal to the averages of the respective data types in all clusters *with* parking data. This is the case for *parking price* and *parking capacity*. One piece of data that still needs to be provided so that the estimation is computed is the date and time. For convenience, we choose the next day at the point when the user starts the model training. An example of the estimation is visualized Figure 16.

C-WOUT ID 19 Timepoint: 2017-11-04 09:00:00

Similarity **cosine** | emd

C-WTIH ID	Similarity	Sim-Type	Estimation	Interval	Intersection
6	1.00	cosine	45.27	[45.19 - 45.35]	[45.19 - 45.35]
7	1.00	cosine	45.15	[45.01 - 45.29]	[45.19 - 45.29]
8	0.99	cosine	57.73	[56.99 - 58.46]	empty
9	0.99	cosine	50.84	[49.84 - 51.83]	empty
5	0.99	cosine	66.71	[65.49 - 67.93]	empty
0	0.98	cosine	52.52	[50.58 - 54.46]	empty
1	0.98	cosine	36.32	[34.10 - 38.54]	empty
2	0.98	cosine	55.45	[53.03 - 57.87]	empty
4	0.97	cosine	58.11	[55.53 - 60.68]	empty
3	0.92	cosine	45.26	[37.51 - 53.01]	empty

Fig. 16. The pop-up table of a cluster without data. Notice the drop-down list from which the time can be selected.

7.4. *Models built on all occupancy datapoints*

Up to now, the evaluation involved models trained and tested on aggregated datapoints, as explained in 6.8. We ask ourselves however, whether a model trained on all datapoints performs better when tested on an aggregated cluster. Or whether an aggregate model deliver better results on an all-datapoints cluster than a model trained on all datapoints? Here we experiment these combinations by training models on both all- and aggregate datapoints and apply them on both types of aggregation forms.

See Table 12 for an overview of test errors for 8 and 16 clusters. One observes that the errors from models applied on aggregated datapoints are about 5 unit points (=5%) smaller than the errors from models applied on all datapoints. This is naturally accounted for by the smaller spread of occupancy values that the aggregation brings with itself. As far as source models are concerned, there is no significant difference between the aggregate and all datapoint models, i.e. the margin is under 1%.

In Table 13 the resulting correlations of testing errors with cosine- and emd similarity values are listed for models of 8 and 16 cluster configurations. Regarding the number of clusters, the values for 8-cluster models are clearly superior to the 16-cluster models. When comparing by target datapoints type, the models applied on aggregated datapoints have a slight edge, with the cosine- and rank cosine correlation values being on average closer to -1 and emd- and rank emd correlation values being closer to 1. However, for 16 clusters the correlations tendency is barely noticeable.

Table 12. Test error for ML models build alternatively with all- and aggregated datapoints

cluster size	datapoints source	datapoints target	test error
8	aggregate	aggregate	**20.19**
8	all	aggregate	21.37
8	aggregate	all	26.25
8	all	all	26.68
16	aggregate	aggregate	**20.47**
16	all	aggregate	21.32
16	aggregate	all	25.97
16	all	all	26.52

Note: Test errors for the same number of clusters can be accurately compared when the *datapoints target* is the same. All models above were build using extreme gradient boosting.

Table 13. Resulting correlation values for ML models built using all- and aggregated datapoints

cluster size	datapoints source	datapoints target	cosine	rank_cosine	emd	rank_emd
8	aggregate	aggregate	**-0.53**	**-0.52**	0.30	0.17
8	all	aggregate	-0.53	-0.43	**0.37**	**0.27**
8	aggregate	all	-0.35	-0.36	0.20	0.14
8	all	all	-0.41	-0.43	0.34	0.25
16	aggregate	aggregate	-0.16	-0.11	0.10	0.05
16	all	aggregate	**-0.18**	**-0.17**	**0.22**	**0.17**
16	aggregate	all	-0.09	-0.06	0.08	0.00
16	all	all	-0.10	-0.11	0.17	0.08

Note: All models above were build using extreme gradient boosting.

7.5. *Amenity area results*

Following up on 6.6 and taking *amenity area* as the basis for the cosine- and emd similarity values leads in practice to correlations listed in Table 14 for 8- and 16 cluster configurations. The superiority of the correlation values using *visiting duration* is observed both for cosine- and rank cosine correlations, which are closer to -1, and for the emd- and rank emd correlations, which are nearer to 1. All correlation values for 16 clusters are, however, relatively weak in absolute measures.

Table 14. The correlation results computed using similarity values based on *amenity area*.

cluster size	amenity type	datapoints (train/test)	cosine	rank_cosine	emd	rank_emd
8	visiting duration	agg/agg	**-0.53**	**-0.52**	**0.30**	**0.17**
8	area	agg/agg	0.05	0.11	0.14	0.22
16	visiting duration	agg/agg	-0.16	-0.11	0.10	0.05
16	area	agg/agg	-0.09	-0.03	0.09	0.07

Note: All models above were build using extreme gradient boosting, trained and tested on aggregated datapoints.

7.6. *Extended prediction models*

An alternative to building urban measures and similarity functions is to let machine learning figure out the similarities by itself. One can simply add the city data as further training information for clusters. The purpose is to produce better predictions by leveraging unknown patterns in the city data. Hence, features representing the *cosine* and *emd* functions are added to the model, as follows:

(i) k features corresponding to the k categories the amenities are split in, i.e. the magnitudes of the vectors for each category;
(ii) a feature corresponding to the emd Gaussian value for that cluster, loosely interpreted as the "total accumulated visiting duration" for that cluster, in case of *visiting duration*, or the "total accumulated area" among all amenities in that cluster, for the amenity area; mathematically it is expressed in Equation 10

$$feature(emd) = x \cdot f(x) \tag{10}$$

where $x =$

$$\begin{cases} \text{visiting duration in minutes,} & \text{if similarity is } \textit{amenity visiting duration} \\ \text{area in square meters,} & \text{if similarity is } \textit{amenity area}. \end{cases}$$

and f is the constructed emd Gaussian function that equals the number of amenities in the cluster for any value x.

We subsequently build *extended machine learning models* that additionally contain the features above. Since these features are identical for all datapoints in a cluster, the model needs to be trained on multiple clusters. Therefore, models on $n-1$ out of n clusters will be build and tested on the remaining cluster. These *all-but-one* or *total models* will be constructed for all n combinations of $n-1$ clusters and their averaged test errors will compared to total models that contain the normal features. The resulting test errors are shown in Table 15 for various number of clusters.

With one single exception the simple total models achieve a better prediction performance, meaning that the addition of the two features did not help with finding better parking occupancy values for the test clusters.

Table 15. Total models extended with cosine and emd features compared to total models with the previous feature set

cluster size	model	test error average
4	xgb	**14.92**
4	xgb extended	16.18
8	xgb	**18.12**
8	xgb extended	19.58
16	xgb	18.19
16	xgb extended	**18.09**
32	xgb	**18.65**
32	xgb extended	20.38

8. Further possible variations

To further investigate parking occupancy prediction given the assumptions in this work, there are several improvements or alternative approaches that can be realized.

(i) **Use other parking settings**. In the present work, several pieces of data could not be integrated because of merging issues, i.e., the location unit did not coincide with the occupancy data's block. Traffic, events, weather, etc. could improve estimation results and hence the final estimations for clusters without parking data. Other sources for parking occupancy data can be found for the cities of Cologne [32], Zurich [33], Santa Monica [34]. In Germany, Deutsche Bahn provides an API to obtain data from parking around train stations [35]. Data pertaining to street occupancy is, however, hard to find. At the time of writing, open data portals mostly provide the location of parking lots, parking meters, parking price and opening times, if applicable.

(ii) **Gather better visiting duration information** Accessing data on the time people spend in various amenities is restricted by Google by not providing an API for it. Other social media services, such as Foursquare offers, however, information on how busy an amenity is by means of the number of check-ins that people send from their phones[de]. The API is however subject to costs and we have not tried it. By gathering more information automatically than we managed to collect manually, the evaluating similarity values based on *visiting duration* will likely increase in accuracy and so will the resulting correlation values.

(iii) **Use more city data**. The parking profiles in the present work are relying on the public amenities from OpenStreetMap. OSM has great potential as a collaborative map service but it lacks many pieces of information that could be useful. Data such as opening hours would be interesting to include in the parking demand profile, which would then take into account the number of

[d]https://developer.foursquare.com/docs/api/endpoints
[e]https://developer.foursquare.com/docs/api/venues/details

public amenities that are available at a certain point in time. Furthermore, the stay duration data collected for the present approach is currently limited. Adding more stay duration data may fine tune the emerging similarity values. Overall, more and finer city data, together with an appropriate representation and similarity function could eventually improve the occupancy estimations for clusters without parking data.

(iv) **Use another clustering method** The effectiveness of the approach is based on the fact that the computed clusters are of approximately same size. Such clusters are likely to contain a similar number of amenities and similarity values can be accurately computed based on them, e.g. cosine similarity is dependent on the absolute number of amenities in a particular category. In contrast emd is robust to the number of amenities as the resulting emd Gaussian is normalized. Clustering alternatives to K-Means such as DBSCAN may improve the similarity computation. Even though it focuses on local density of neighborhoods, DBSCAN can be made to keep the cluster sizes more or less even, hence perhaps produce better cosine similarity results.

(v) **Apply semi-supervised machine learning**. Another relevant machine learning approach in this case is based on organizing the city areas as an undirected graph. The vertices represent the clusters with their respective occupancy data, while the edges between them are assigned similarity values. Initially, only a part of the vertices have the occupancy value known, i.e., the clusters with parking data, while the rest has undetermined occupancy, i.e., the clusters without parking data. At each step, the value for a vertex whose value is undetermined is being computed by considering the occupancies of the linked vertices and their corresponding similarity values.

9. Conclusion

In this work, we have presented an attempt at approximating street parking occupancy in cities. Under the assumption that parking data is lacking, in order to build scalable occupancy prediction systems, we proposed an alternative solution to the ones previously developed for this problem. We built parking demand profiles by using complementary city data, which localize various types of public amenities and indicate the average visiting duration there. All data has been made available in an established RDF format, so that it can be easily reused. We merged the *parking data* with the *city data* by matching parking *location units* to *points of interest*, split the city into clustered areas, and built machine learning models for them. K-Means was used to cluster the location units, while four methods were employed to train models for the clusters: decision trees, support vector machines, multilayer perceptrons and extreme gradient boosting. Based on the city data, urban measures were built in the form of *cluster vectors* and *cluster Gaussians*, both of which took advantage of the mean visiting duration and its standard deviation. The vectors were part of the *cosine similarity* computation, while Gaussians contributed to the *earth mover's*

distance calculation. The occupancy estimations for clusters *without* parking data were defined in terms of model estimations from clusters *with* parking data and the corresponding cluster similarity values. The estimations are expressed as intervals which extend the model prediction values by the magnitude of the similarity values.

As use case, we chose the SF*park* project from San Francisco, which gathered parking data for more than 2 years starting in 2011 and now offers it for free usage. The city data was collected from OpenStreetMap as amenity information, and from Google Places as stay duration values. Both sources are open and free of charge. Over 30 types of public amenities were found in the San Francisco blocks, which corroborated with over 470 Google Places sources, lead to building the urban measures and similarity values.

The results confirmed that clustering the city into smaller areas yields better occupancy estimations than those of entire city area models. Following our tests, the best machine learning model turned out to be *extreme gradient boosting*. We used the clusters *with* parking data for the evaluation of the similarity values and calculated correlation coefficients between the similarity values and the estimation errors, using both absolute values and ranks. The best correlation were reached for the 100m merge distance for 8 clusters, averaging at -0.55 as Pearson Coefficient and -0.49 as Spearman's rank coefficient. In the same configuration, both *cosine similarity* and *emd distance* reached their best results from all the test configurations. Overall, *cosine similarity* achieved better correlations than *emd*. Finally, the models for 8 clusters produced superior results over the models for 16 clusters.

10. Note

You may observe some inconsistencies in the evaluation results for e.g. testing errors built with the same models for the same data aggregation and other attributes. Please note that the clustering associated with a experiment run is different every time (due to its random seed) and therefore this may result in e.g. testing errors that may slightly differ for the same configuration in various stages of the evaluation. We cross-checked the results and intended to keep everything consistent but since the exact values are not 100% reproducible we decided to leave some parts as they are while pointing out their tendency and meaning in each case.

References

1. M. N. Smith, The number of cars worldwide is set to double by 2040 (2016), `https://www.weforum.org/agenda/2016/04/the-number-of-cars-worldwide-is-set-to-double-by-2040`.
2. D. Shoup, Free parking or free markets (2001), `https://www.accessmagazine.org/spring-2011/free-parking-free-markets/`.
3. A. Ionita, Extending estimation of parking occupancy to untracked city areas using city background information, master's thesis (December 2017).
4. T. Lin, H. Rivano and F. Le Mouël, A survey of smart parking solutions, *IEEE Transactions on Intelligent Transportation Systems* (2017).

5. T. S. Lin, *Smart parking: Network, infrastructure and urban service*, PhD thesis, Lyon, INSA2015.
6. M. Caliskan, A. Barthels, B. Scheuermann and M. Mauve, Predicting parking lot occupancy in vehicular ad hoc networks, in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th* IEEE2007, pp. 277–281.
7. A. Klappenecker, H. Lee and J. L. Welch, Finding available parking spaces made easy, *Ad Hoc Networks* **12** (2014) 243–249.
8. E. H.-K. Wu, J. Sahoo, C.-Y. Liu, M.-H. Jin and S.-H. Lin, Agile urban parking recommendation service for intelligent vehicular guiding system, *IEEE Intelligent Transportation Systems Magazine* **6**(1) (2014) 35–49.
9. T. Tiedemann, T. Vögele, M. M. Krell, J. H. Metzen and F. Kirchner, Concept of a data thread based parking space occupancy prediction in a berlin pilot region., in *AAAI Workshop: AI for Transportation*2015.
10. T. Rajabioun, B. Foster and P. Ioannou, Intelligent parking assist, in *Control & Automation (MED), 2013 21st Mediterranean Conference on* IEEE2013, pp. 1156–1161.
11. T. Rajabioun and P. A. Ioannou, On-street and off-street parking availability prediction using multivariate spatiotemporal models, *IEEE Transactions on Intelligent Transportation Systems* **16**(5) (2015) 2913–2924.
12. S. F. M. T. Agency, Sfpark (2011–2013), `http://sfpark.org`.
13. E. I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos and M. G. Karlaftis, Exploiting new sensor technologies for real-time parking prediction in urban areas, in *Transportation Research Board 93rd Annual Meeting Compendium of Papers*2014, pp. 14–1673.
14. Z. Chen, J. C. Xia and B. Irawan, Development of fuzzy logic forecast models for location-based parking finding services, *Mathematical Problems in Engineering* **2013** (2013).
15. B. Xu, O. Wolfson, J. Yang, L. Stenneth, S. Y. Philip and P. C. Nelson, Real-time street parking availability estimation, in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on* **1**, IEEE2013, pp. 16–25.
16. X. Chen, Parking occupancy prediction and pattern analysis, *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CS229-2014* (2014).
17. R. Hossinger, K. Heimbuchner and T. Uhlmann, Development of a real-time model of the utilisation of short-term parking zones, in *19th ITS World Congress*2012.
18. F. Caicedo, C. Blazquez and P. Miranda, Prediction of parking space availability in real time, *Expert Systems with Applications* **39**(8) (2012) 7281–7290.
19. A. Nandugudi, T. Ki, C. Nuessle and G. Challen, Pocketparker: Pocketsourcing parking lot availability, in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* ACM2014, pp. 963–973.
20. P. Szczurek, B. Xu, O. Wolfson, J. Lin and N. Rishe, Learning the relevance of parking information in vanets, in *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking* ACM2010, pp. 81–82.
21. Y. Ji, D. Tang, P. Blythe, W. Guo and W. Wang, Short-term forecasting of available parking space using wavelet neural network model, *IET Intelligent Transport Systems* **9**(2) (2014) 202–209.
22. A. Koster, A. Oliveira, O. Volpato, V. Delvequio and F. Koch, Recognition and recommendation of parking places, in *Ibero-American Conference on Artificial Intelligence* Springer2014, pp. 675–685.
23. A. O. Kotb, Y.-C. Shen, X. Zhu and Y. Huang, iparker—a new smart car-parking system based on dynamic resource allocation and pricing, *IEEE Transactions on Intelligent Transportation Systems* **17**(9) (2016) 2637–2647.
24. S. Pullola, P. K. Atrey and A. El Saddik, Towards an intelligent gps-based vehicle

navigation system for finding street parking lots, in *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on* IEEE2007, pp. 1251–1254.

25. F. Richter, S. Di Martino and D. C. Mattfeld, Temporal and spatial clustering for a parking prediction service, in *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on* IEEE2014, pp. 278–282.
26. J.-H. Shin and H.-B. Jun, A study on smart parking guidance algorithm, *Transportation Research Part C: Emerging Technologies* **44** (2014) 299–317.
27. Y. Zheng, S. Rajasegarar and C. Leckie, Parking availability prediction for sensor-enabled car parks in smart cities, in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on* IEEE2015, pp. 1–6.
28. S. F. M. T. Agency, Sfpark - pilot project evaluation summary (2011–2013), `http://sfpark.org/wp-content/uploads/2014/06/SFpark_Eval_Summary_2014.pdf`.
29. S. F. M. T. Agency, Sfpark - open data (2011–2013), `http://sfpark.org/how-it-works/open-data-page/`.
30. S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser and W. Trappe, Parknet: drive-by sensing of road-side parking statistics, in *Proceedings of the 8th international conference on Mobile systems, applications, and services* ACM2010, pp. 123–136.
31. U. of Surrey (UK), A. I. (Denmark), E. (Sweden), N. U. of Ireland/DERI (Ireland), U. of Applied Sciences Osnabrück (Germany), S. (Austria, Romania), C. of Aarhus (Demark), C. of Brasov (Romania) and W. S. U. (USA), City pulse (2013–2016), `http://www.ict-citypulse.eu/page/`.
32. S. Köln and DKAN, öffene daten köln (2015), `https://www.offenedaten-koeln.de/dataset/taxonomy/term/52/field_tags/Transport%20und%20Verkehr-52?query=park&sorting=changed%7CDESC`.
33. S. Zürich, Stadt zürich - open data (2015), `https://data.stadt-zuerich.ch/dataset/parkleitsystem`.
34. C. of Santa Monica, Santa monica - open data (2014), `https://data.smgov.net/Transportation/Parking-Lot-Counts/ng8m-khuz`.
35. D. B. GmbH, Db - parkplätze api (2016), `http://data.deutschebahn.com/dataset/api-parkplatz`.