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.

Xu et al. [4] make real-time parking availability estimations based on data collected from mobile phones. The system uses algorithms incorporating statistical weighted schemes and Kalman filters. Additionally, the authors create parking availability profiles based on historical data and statistical algorithms.

Other works making use of smart phones are the ones by Nandugudi et al. [5], Koster et al. [6], and Chen et al. [7]. After data collection, often using the sensors in the smartphone, they use different machine learning models, such as probabilistic distribution, Bayesian, and Hidden Markov models, as well as fuzzy logic to predict the available parking spaces.

Instead of using mobile devices that are carried by drivers, Park-Net, developed by Mathur et al. [8] is a system made up of dedicated vehicles that captures parking space information while driving. Every ParkNet vehicle is equipped with a GPS receiver and an ultrasonic sensor facing sideways. 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.

Compared to approaches that collect data via mobile phones or dedicated cars, other approaches use data collected by sensors, parking meters or cameras that are installed on streets or parking lots. For instance, Tiedemann et al. [9] developed a prediction system that gives estimated occupancies for parking spaces in Berlin, Germany. 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 a significant part in the predictions, such as holidays, weather and thus, they use the neural gas clustering to separate the data, before the data thread method is applied. This perfectly matches our idea of considering further information about the circumstances in a city. However, our approach considers more detailed information about specific districts rather than general information like holidays, which usually hold for an entire city.

Similar to the approach presented by Tiedemann et al. [9], Hössinger et al. [10] 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.

Located in Spain (Barcelona), Caicedo et al. [11] developed 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. Similar to our work, Caicedo et al. [11] take further factors into account, such as duration of stay and capacity of every operating parking facility.

Rajabioun and Ioannou [12] 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, 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 follow-up paper, Rajabioun and Ioannou [13] propose a multivariate autoregressive model that considers the temporal and spatial correlations of parking availability when making predictions. The authors hold that the model, which is integrated into a parking guidance and information system, recommends parking locations with high accuracy.

In contrast to Rajabioun and Ioannou [12], Chen [14] tackles the parking problem by aggregating parking lots. Their findings show 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 evaluated 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 [15]. The findings of Chen confirm our idea building models for smaller areas, such as one district and one parking lot. However, our work goes one step further by additionally transferring the trained model to districts where no data is available.

Richter et al. [16] 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 clustering are tried out. Hierarchical clustering with complete linkage is employed. The models are evaluated on street data from the SF*park* project[15] and reach a prediction success rate of about 70%.

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 [17] [18]. 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 **??** 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 [19] 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 1.

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. 1).

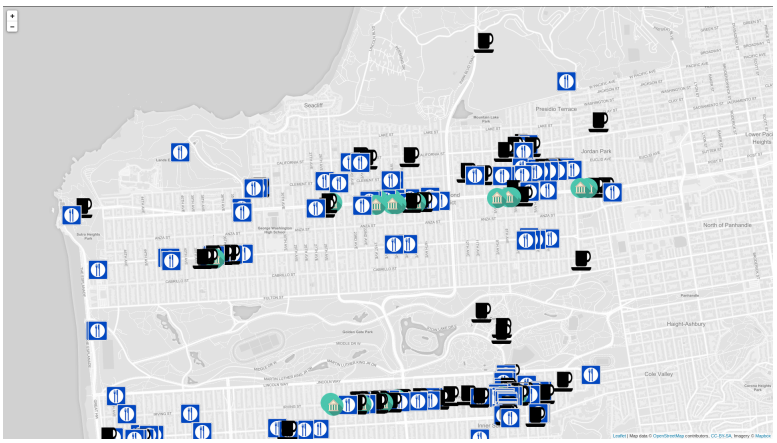

Fig. 1. A map indicating public amenities (cafes, restaurants, banks) found at points of interest in OSM.

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

Table 1. 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.)

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* values and popularity of the place for specific points in time. The average values are based on the users' smart phone GPS sensors (cf. 2). 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

Fig. 2. An example of visiting duration information found on Google Maps.

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 as part of the CityPulse project [19]. 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 6.3.

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 6.4.

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. 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 3 for a general representation.

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

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

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.

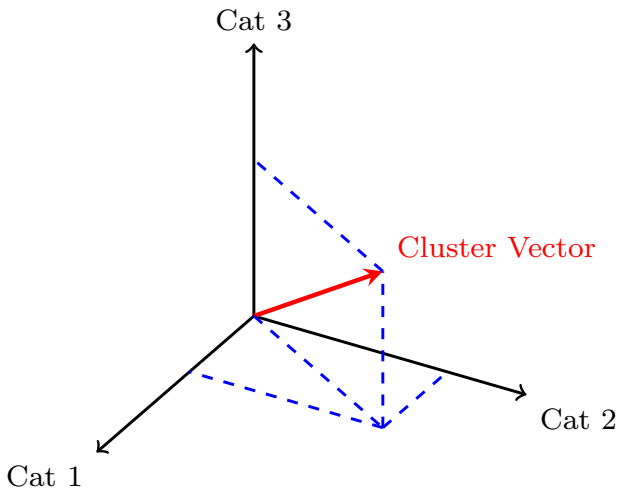

Fig. 3. An example of a cluster vector for three categories.

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 4 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 $\mathcal{C}_i$.

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:

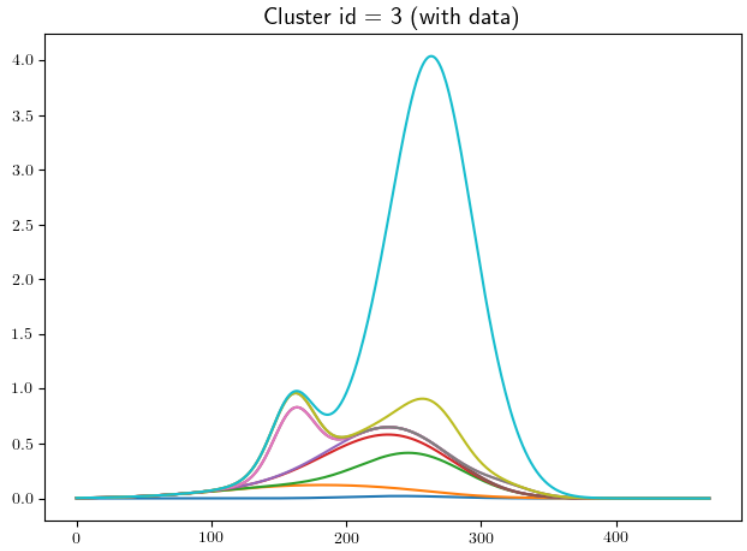

Fig. 4. The summing of Gaussians resulting in a cluster Gaussian.

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

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 $3\times$ the largest standard deviation to it, as it is known that within $3\times$ 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 $3\times$ 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 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. Evaluation

For evaluating our approach, we require 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 data were available. Therefore, we first calculated the similarity score between a cluster c and all other clusters. Afterwards, we calculated the estimation errors when applying the model of cluster c to all other clusters.

6.1. *SFpark Parking Data*

The SF*park* data are visualized in 5 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.

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 2.

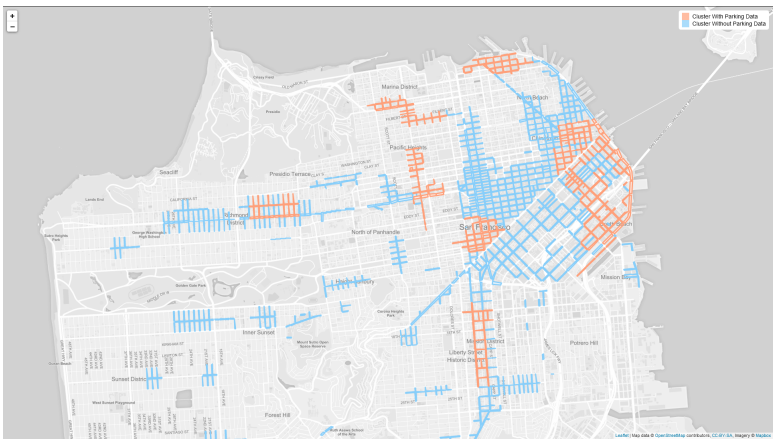

Fig. 5. The blocks accounted in SF*park*. The light blue ones are blocks *without* parking data, the light red ones are *with* parking data.

Table 2. List of all OSM amenities found in the SF*park* blocks.

arts_centre	dojo	marketplace	shelter
bank	embassy	music_rehearsal_place	shop
bar	fast_food	music_school	spa
biergarten	grocery	nightclub	stripclub
bureau_de_change	gym	pet_grooming_shop	studio
cafe	hookah_lounge	pharmacy	training
clinic	ice_cream	police	veterinary
clothes_store	karaoke	post_office	vintage_and_modern_resale
community_centre	lan_gaming_centre	pub	
dentist	laundry	restaurant	
doctors	library	salon	

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 set 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. We assign to it distances of 100m, 200m, and 400m.

6.4. *Clustering*

As established in 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 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. 6).

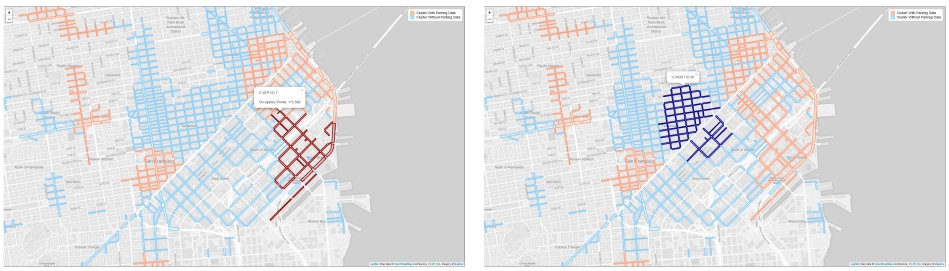

Fig. 6. Highlighted cluster *with* parking data on the left side and a cluster *without* parking data on the right side.

The resulting aggregated blocks is 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 3 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.

Alongside this information, we need the amenity categories in order to derive the *cluster vectors*. As defined in 5.6.1, the categories are based on the visiting duration mean. We split them in three categories: under half an hour, 31 to 90 minutes and more than 1.5 hours. The assigned partitions for every amenity are shown in 3.

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

Table 3. 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 is expressed and standard deviation are expressed in minutes. The assigned category for cluster vectors is included.

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

6.6. *Model Training and Evaluation for Clusters* with *Parking Data*

We use four methods to train models for the clusters *with* parking data: *decision trees*, *support vector machines*, *multilayer perceptrons*, *gradient boosted trees*. 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. However, aggregating traffic data on the district level, which is available for the occupancy data as well, does not provide 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.

The evaluation here consists in comparing the test-errors of cluster models with the independently-computed similarity values. More specifically, a *source* cluster's model will tested on a *target* cluster. The resulting error (measured as root mean square) will be compared to the similarity (cosine or EMD) between the *source* and the *target* clusters.

6.7. *Aggregating training data*

As indicated in 5.7, the training data is aggregated across all blocks so that it can be applied later on other clusters. 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. This means that the original collection of data records shrinks, which should improve the training time. In **??**, the expansion/shrinking rate is shown for various number of clusters.

In 7 a screenshot of the application showing a sample of the results is illustrated. 8 displays the presented table in more detail.

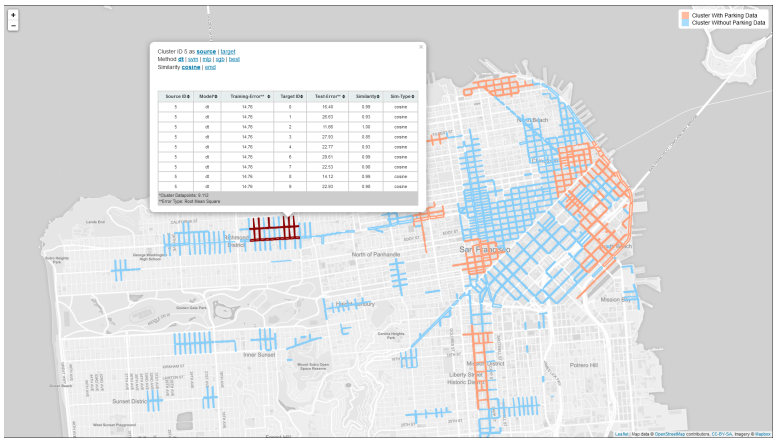

Fig. 7. Selected cluster *with* parking data and the pop-up table in the Leaflet application.

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. 8. The pop-up table for the Leaflet application view of 7.

6.8. *Best Model Method*

Models were trained using four methods: decision trees, support vector machines, multilayer perceptrons, gradient boosted trees. **??** 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.

6.9. *Clustered- vs. Total Models*

One assumption of our approach is that models originating from smaller clusters are better at predicting occupancy than models trained with the entire city area. We have compared the two types of models during our tests. For each target cluster $\mathcal{C}_{target}$, we determine the source cluster $\mathcal{C}_{source}$ whose model has the best estimation error when applied on $\mathcal{C}_{target}$. We also train a model containing the entire city area with parking data $\mathcal{A}$ minus $\mathcal{C}_{target}$'s data and compute this model's estimation error on $\mathcal{C}_{target}$. As we can observe from **??**, in the case of 8 clusters, and in **??** for 16 clusters, the cluster's models estimations are superior to the ones of the total model with very few exceptions.

6.10. *Similarity Values vs. Estimation Errors*

The goal of our approach was to replace occupancy estimations for clusters where no parking data is available with estimations based on cluster similarity values. Among clusters *with* parking data, the real occupancy values are known. This enables us to compute estimation errors for cluster models, which can then be correlated with the similarity values between clusters. We use two correlation coefficients: the Pearson correlation coefficient and Spearman's rank correlation coefficient.

We have evaluated both cosine and EMD similarity values in 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 **??** the final results are shown. For each similarity measure (cosine, rank cosine, emd, rank emd), the correlation value between the similarity and parking occupancy values is averaged over all clusters. 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.

6.11. *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 9.

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

7. Extensions

In the following we present some extensions and variations that consolidate the approach presented, if not improve upon it.

Firstly, in 7.1) we investigate the results by skipping the aggregating step in machine learning training (recall 6.7). Instead of averaging the datapoints over timestamp per cluster, we build the cluster models using the entire occupancy data.

Secondly, in 7.2 we use another amenity value on which of the similarity functions are applied (recall 7.2). Instead of *amenity time spent*, *amenity area* will extracted and leveraged.

Thirdly, in 7.3 we question whether the similarity function approach is the most efficient and transfer its purpose to the machine learning phrase (recall 7.3). The model will receive absolute *cosine* and *emd Gaussian* values as additional features and its error will be compared to the previous models.

7.1. *Using all occupancy datapoints*

Recall that in 6.6, the training data for a cluster is aggregated per timestamp, i.e. the *price*, *capacity* and *occupancy* attributes (see 4 for an example). The training phase was subsequently more time-consuming as it involved digesting a significantly larger dataset (see **??** for the expansion/shrinking rates for various number of clusters). Among the resulting models, *extreme gradient boosting* proved to performed the best in terms of test errors (as in the aggregated points case). Otherwise, the models ended up poorer compared to the models with averaged values in terms of training and test errors. (see Table 5). Correlating the occupancy results delivered by theses models with the unchanged similarity values results in generally inferior numbers to correlations based on model from aggregated datapoints. Apart from the ranked emd correlation for 8 and 16 clusters and the ranked cosine for 16 clusters, all other correlation measures are not improved upon (see 6).

Table 4. 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 are blocks belonging to cluster 1 which are transformed into two entries by averaging *price rate*, *total spots* and *occupied* attributes per timestamp (subtable below)

Table 5. Train and test error for the ML models build with all datapoints

cluster size	datapoints	training error	test error
8	aggregate	**12.92**	**20.13**
8	all	19.41	26.86
16	aggregate	**13.52**	**20.52**
16	all	19.05	26.36
32	aggregate	**14.59**	**21.67**
32	all	18.54	27.50

Note: The same values are shows by comparison for the models build on aggregate datapoints for 8, 16 and 32 clusters; all models above were build using extreme gradient boosting

Table 6. Resulting correlation values for ML models built using all datapoints alongside correlations of models based on aggregate datapoints for 8, 16 and 32 clusters

cluster size	datapoints	cosine	rank_cosine	emd	rank_emd
8	aggregate	**-0.47**	**-0.50**	0.26	0.17
8	all	-0.41	-0.42	**0.34**	**0.25**
16	aggregate	**-0.11**	-0.11	**0.14**	0.10
16	all	-0.02	**-0.14**	0.13	**0.12**
32	aggregate	**-0.13**	**-0.06**	**0.09**	**0.07**
32	all	-0.01	-0.02	0.03	0.01

Note: All models above were build using extreme gradient boosting

7.2. *Amenity area as similarity basis*

Together with the similarity functions we considered *timespent* as the basis for creating the urban profile (recall 6.5). However, there are other measures that reflect the parking demand corresponding towards an amenity, one of them being area of the particular amenity, which is available in OpenStreetMap. OSM provides a *polygon layer* for a certain geographic bounding box, which contains information across all the surfaces in that region (recall 5.2). In extracting this information we had two options.

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 disadvantages: 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 case 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 (see 9). When calculating the coefficient of variation based on the matching of POI with the polygons, it came out 2.1.

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

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 (see 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 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.

Replacing the amenity area's key values in the similarity computation, i.e. category in cosine similarity, mean and standard deviation in earth mover's distance, we arrive at the following correlation values (see 8). It is clear that the area did not lead to an improvement of the correlation values.

Table 7. Amenity area values gathered 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: mean and standard deviation values were reduced by a 20x factor; categories are 0 - 35, 36 - 100, 100+

Table 8. The correlation results based on the amenity area values.

cluster size	amenity type	method	cosine	rank_cosine	emd	rank_emd
8	timespent	xgb	**-0.47**	**-0.50**	**0.26**	0.17
8	area	xgb	-0.20	-0.18	0.14	**0.22**
16	timespent	xgb	**-0.11**	**-0.11**	**0.14**	**0.10**
16	area	xgb	-0.05	-0.01	0.05	0.03
32	timespent	xgb	**-0.13**	**-0.06**	**0.09**	**0.07**
32	area	xgb	0.08	0.06	-0.05	-0.02

7.3. *Machine learning better than the similarity functions?*

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 find better similarities by leveraging unknown patterns in the city data. Hence, features representing cosine and emd are added to the model, as follows:

- a feature corresponding to the k categories the amenities are split in, specifically the magnitudes of the vectors for each category;
- a feature corresponding to the emd Gaussian value for that cluster, specifically the area under the emd Gaussian.

The subsequently built machine learning models cannot however be formed on a single cluster, as the additional features would be the same for all datapoints, of course. Therefore, models on $n-1$ out of n clusters will be build and tested on the remaining cluster (as it was done in the total models evaluation - recall 6.9). The resulting training and test errors are show in 9. (discussion)

Table 9. Models extended with cosine and emd features compared to previous models with regard to training- and testing errors

cluster size	model	training error avg	test error avg
8	xgb	17.90	**18.01**
8	xgb extended	**17.46**	19.08
8	dt	**18.85**	**18.52**
8	dt extended	19.84	19.46
16	xgb	18.42	18.26
16	xgb extended	**17.63**	**17.02**
16	dt	19.14	18.83
16	dt extended	**18.77**	**17.81**

Note: all models are total models

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 [20], Zurich [21], Santa Monica [22]. In Germany, Deutsche Bahn provides an API to obtain data from parking around train stations [23]. 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) **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 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.

(iii) **Integrate city data in machine learning models**. An alternative to building urban measures and similarity functions is to let machine learning figure out the similarities by itself. One can add the city data as further training information for clusters. The models are then applied on clusters without parking data and return occupancy estimations. The difference here is that models will be built for all clusters, including the ones without parking data. This also has the disadvantage of not using most of the parking data for training. The benefit of finding better similarities by leveraging unknown patterns in the city data may, however, outweigh this drawback.

(iv) **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.

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. 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.
5. 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.
6. 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.
7. 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).

8. 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.
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. 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.
11. F. Caicedo, C. Blazquez and P. Miranda, Prediction of parking space availability in real time, *Expert Systems with Applications* **39**(8) (2012) 7281–7290.
12. T. Rajabioun, B. Foster and P. Ioannou, Intelligent parking assist, in *Control & Automation (MED), 2013 21st Mediterranean Conference on* IEEE2013, pp. 1156–1161.
13. 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.
14. X. Chen, Parking occupancy prediction and pattern analysis, *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CS229-2014* (2014).
15. S. F. M. T. Agency, Sfpark - open data (2011–2013), `http://sfpark.org/how-it-works/open-data-page/`.
16. 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.
17. S. F. M. T. Agency, Sfpark (2011–2013), `http://sfpark.org`.
18. 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`.
19. 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/`.
20. 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`.
21. S. Zürich, Stadt zürich - open data (2015), `https://data.stadt-zuerich.ch/dataset/parkleitsystem`.
22. C. of Santa Monica, Santa monica - open data (2014), `https://data.smgov.net/Transportation/Parking-Lot-Counts/ng8m-khuz`.
23. D. B. GmbH, Db - parkplätze api (2016), `http://data.deutschebahn.com/dataset/api-parkplatz`.