Predictive Ridesharing based on Personal Mobility Patterns

Roman Roor[1], Michael Karg[1], Andy Liao[2], Wenhui Lei[2] and Alexandra Kirsch[3]

***Abstract*— For digital mobility assistants it is advantageous to know users' mobility habits to be able to infer the most probable departure time and next destination. Different approaches are known to face this challenge, but most of them either have a very static feature model and limited extensibility capabilities or they are very complex and require exponential amount of training data for every added feature. This paper introduces a flexible and extendible mobility model – to represent a user's movement and habits – using a Variable-order Markov Model (VOMM) based on users' mobility patterns enriched with different temporal context information. Since this model uses a tree like data structure, it is possible to find patterns of different lengths in the same training data. Spatio-temporal next location prediction is based on the Prediction by Partial Matching (PPM) algorithm. We examine several classification and regression based machine learning algorithms for probability fusion of next location candidates and possible departure times to obtain the most accurate joint probability for the predicted location. The resulting prediction accuracy is between 60% and 81%.**

I. Introduction

Big cities are growing fast and the population is rising. For the first time in history, in 2007 more people were living in cities than in rural areas [1]. Among others, in some Asian countries, e.g. China and India, the population is increasing faster than the urban transport infrastructure can be extended. Commuting from place A to B is challenging in such cities, especially when using a car. A lot of traffic, particularly in the rush hours, can be seen as a result of urbanization and rising car ownership. Increasing traffic, on the one hand, is causing a higher risk for traffic jams and, on the other hand the air pollution caused by car exhaust gas increases dramatically. Increasing air pollution and rising number of car owners are also one of the reasons for road space rationing (also called driving restriction) in various Chinese cities, such as Shanghai and Beijing. This policy regulates, which car is allowed to enter a common road space based upon the last digit of the license number on certain established days. Thus, not every car owner is allowed to use the own car on specific days. This issue especially affects commuters without sufficient access to a public transportation network.

As a solution we propose a ridesharing based approach to still be able to commute to work without spending high amount of money or forgo on the comfort of using a car. Thanks to the widespread adoption of smartphones with integrated localization capabilities, interested users can easily track their mobility habits. An interesting aspect of a predictive system is that it can proactively trigger actions based on the user's habits. Predictive ridesharing strongly benefits from removing the effort of manually entering rides into the system but instead offering automated entries based on a user's habits. Benefits are time saving by removing periodic ridesharing related tasks, traffic reduction by increasing the number of riders per car, splitting costs and being eligible for other benefits, such as car pool lanes. In predictive ridesharing, the system predicts when drivers and riders from the same area will travel to similar destinations and match them. Drivers just need to accept automatic matches by confirming e.g. a push-notification of a smartphone app. Taking the effort out of every day tasks by proactively suggesting solutions based on the user's habits is an exciting aspect of mobility pattern prediction.

To proactively trigger actions based on the users' daily movements, it's advantageous for the ridesharing assistant to know the users' mobility habits and infer information about further movements, such as next location and departure time. This knowledge can help the assistant to announce ridesharing demands on an online ridesharing platform and match the right drivers and riders for carpooling with the same commute demands and almost no interaction for the users. Thus, we propose a model for spatio-temporal next location and departure time prediction. This model incorporates different spatial and temporal context information. While predicting next location and departure time, sub-probabilites of the different used features will be merged, using machine learning algorithms, into a joint probability to gain the most probable result. Knowledge of recent movement habits enable the assistant to respond to fast habit changes and the ability to still offer the best ridesharing assistance. Therefore it's beneficial to use a simple data structure regarding model updates to keep the most recent model of the users' movements.

Our contributions to spatio-temporal next location prediction with Prediction by Partial Matching (PPM) on a Variable-order Markov Model (VOMM) are a) a model for predicting arrival and departure time with fine-grained resolution as well as location prediction, b) analysis of the variety of temporal context information trees and c) a model for merging different predictions based on different features to obtain a joint probability for predicted next location including arrival and departure time.

II. Related Work

The modeling of mobility patterns is an active research topic. The entire field covers a variety of applications, like the identification of significant locations, activities and means of

[1]BMW AG, Munich, Germany {roman.roor, michael.karg}@bmw.de

[2]BMW Technology Office China, Shanghai, China {andy.liao, wenhui.lei}@bmw.com

[3]Department of Computer Science, Eberhard Karls Universität Tübingen, Tübingen, Germany alexandra.kirsch@uni-tuebingen.de

transportation detection as well as social aspects influencing a person's mobility. For our mobility habit model the work in location prediction is of special relevance.

In case of everyday use of location prediction it is important to detect a user's significant location changes. Continuous usage of Global Positioning Systems (GPS) can drawn a smartphone's battery fast. In [2], [3] the authors apply battery saving strategies to reduce battery consumption of localization up to 87% whereat over 90% of significant location changes can be detected. In addition Papandrea et al. proposes a mobility model that can detect the user's behaviour to adapt the location monitoring parameters and predict the next action [3].

Noulas et al. try to extract a set of features from 35m Foursquare check-ins made by 1m users to capture user movements and intentions. Due to the sparse check-in data an accuracy of 50% could be achieved using M5 model trees [4]. Foursquare and Gowalla check-ins are also exploited in [5] using a Hidden Markov Model (HMM) to predict whether the users' next location will be a never observed or already known one. A classification error rate of almost 20% could be achieved. Ye et al. also try to predict a users' next location based on Gowalla check-ins. They use a mixed HMM, but first they predict the category of the next location to restrain the solution space and in the next step the next location within the predicted category [6]. This approach predicts the next category with an accuracy of 44%.

Similar to [6] Lee et al. also predict the next location based on the predicted action at the next location using a dynamic Bayesian network (DBN) [7]. In this case students were asked to provide a diary (time, location, action) about their movements at a university campus over several weeks. Based on this dataset an accuracy of over 72% was achieved. A similar low accuracy is shown in [8] based on the same idea as [5], [6] using Mixed Membership Stochastic Blockmodels (MMSB).

The model approaches by [9], [10] proofs higher accuracy using next location prediction without prior context recognition based on GPS and/or other smartphone sensor data. Scellato et al. achieved a prediction accuracy up to 90%.

Markov chains show lower next location prediction accuracies on GPS or WiFi data. In [11], [12] accuracies of 48% (WiFi, indoor) and 80% (GPS, outdoor) are shown respectively.

Another popular approach is based on VOMM for predicting discrete sequences over a finite alphabet using PPM [13]. VOMMs are more flexible and easier to train than HMMs [14]. Also the computation of transition probabilities is not needed, since probability computation is done by PPM in another step. Bapierre et al. extend the VOMM with a temporal and a social component and use it for spatio-temporal next location prediction [15]. This approach was very limited to only one coarse-grained temporal tree without result fusion. Different possible departure time possibilities have not been considered. Due to a limited amount of co-locations for the social context component, no serious accuracy results are evaluated.

Fig. 1: Prediction timeline. The usual arrival, dwell, departure, travel time will be considered for next location prediction.

In this work we use approaches like the ones described in [13], [15] to build a model for predicting arrival and departure time but with fine-grained resolution as well as location prediction. Further, we provide an analysis of the variety of temporal context information trees. To our knowledge, none of the existing approaches allow for the explicit continuous retraining of the model and sub-result merging while predicting next location. Thus, we propose a model for merging different predictions based on different features to obtain a joint probability for predicted next location and departure time.

III. PREDICTION ALGORITHM

A. Prediction Concept

The prediction goal is to determine the user's most likely transition to the next location given the current location and time. The steps and components involved in the prediction are visualized in fig. 1. Upon arrival at the current location the departure time prediction is triggered. Then, to estimate the arrival time at the next location candidate the travel time to that location has to be added. Each visited location is logged in a tree (see fig. 2) and contains the arrival time of each visit. This information can later be used by the prediction component to derive how likely a visit at the predicted time would be. Two other temporal aspects – dwell time and departure time – are treated analogously.

The spatial prediction component determines the number of times the next location candidate has been observed after the current sequence of locations.

B. Variable-order Markov Model

In the field of lossless compression the VOMM has it's classical application [16]. The VOMM extends the concept of Fixed Order Markov Movels (FOMMs) and is a powerful approach to model and predict sequential data [13]. Its main advantage is the ability to detect patterns of varying context length.

The following example is taken from [17]: the sequence $xxyzxxxyzxxxyzxxxyz...xxxyz$ of random variables from the alphabet $\{x, y, z\}$ is represented by a VOMMs with upper bound order of 2 by the conditional probabilities:

- $Pr(x|xx) = 0.5$
- $Pr(y|xx) = 0.5$
- $Pr(z|xy) = 1.0$
- $Pr(x|yz) = 1.0$
- $Pr(x|zx) = 1.0$

With an upper bound model order of 3, every successor symbol can be determined with a 100% confidence, such as by replacing $Pr(x|xx) = 0.5$

with $Pr(x|zxx) = 1.0$ and $Pr(y|xx) = 0.5$ with $Pr(y|xxx) = 1.0$. In contrast, a VOMM of order 1 requires the 9 conditional Probability components: $\{Pr(x|x), Pr(x|y), Pr(x|z), Pr(y|x), Pr(y|y), Pr(y|z), Pr(z|x), Pr(z|y), Pr(z|z)\}$. A VOMM of order 2 requires 27 evaluations: $\{Pr(x|xx), Pr(x|xy), \ldots, Pr(z|zz)\}$. And a fixed order of 3 demands 81 evaluations: $\{Pr(x|xxx), Pr(x|xxy), \ldots, Pr(z|zzz)\}$. This exercise exemplifies the amount of training data a FOMM requires to evaluate all the conditional probabilities. The same problem can be expressed by a VOMM with 5 conditional probabilities, which is a great reduction in model parameters [16].

Nomenclature:

- Σ finite alphabet
- $q_t \in \Sigma$ random variable q at time t from alphabet Σ
- s context
- n upper bound model order
- $P(q_t|s)$ conditional probability for symbol given context
- $s \in \Sigma^*$ the * sign represents a sequence of states of any length

The VOMM has a main difference in contrast to the FOMM, because n is only an upper bound for the VOMM, but it's the required context length $|s| = n$ for the FOMM.

A VOMM tree has a depth of $n + 1$. The symbol q is denoted by the leaves and based on the context $s = (q_{t-n}, \ldots, q_{t-1})$, that represents the path from the root to a leaf. To construct the tree it's either necessary to create new nodes for symbols, that has never been observed after the given context or increment the count of the already existing node as shown in fig. 3a. Note that after processing the first three symbols (*Home*, *Work* and *Gym*), only *Home* is fully finished. On the path each node represents a sub-pattern for a path leading up to it from the root. After processing a sequence of 8 locations the final tree structure is depicted in fig. 3b. The location's popularity can be determined by the count attribute of the first layer nodes. Thus, for instance, only having the root as context an estimation of the next location still can be done. To determine the probability of the next location it's necessary to traverse the tree according to the given context, e.g. $s = (Home, Work)$, and then calculate the conditional probability based on the leaf node's count.

To train the VOMM three main tasks exists: counting, smoothing and modelling. As mentioned before the occurrence counts of symbols after their context determines the probabilities. Smoothing describes how the problem of not seen symbols is handled, also called *zero-frequency problem*. To solve this issue the PPM allocates a certain probability share for unobserved symbols. In the last step, the modelling, the actual VOMM is constructed. After the first context sequence starting with *Home* is processed the next (sub-)sequence starts with *Work* and creates the next path originating from root.

C. Prediction by Partial Matching

The PPM algorithm predicts a symbol based on the context represented by the VOMM [13]. The probability for a symbol

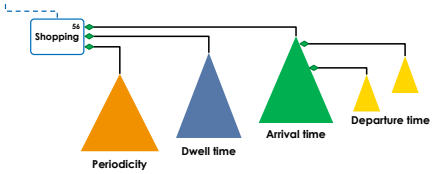
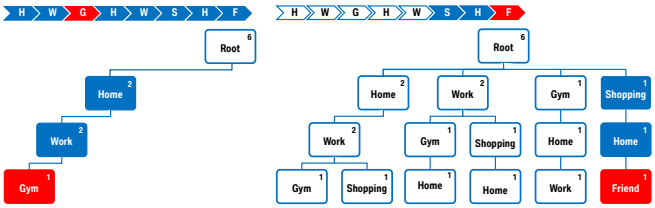


Fig. 2: Context tree extension with temporal context information. Every spatial node keeps several temporal context trees as attributes.



(a) after three locations (b) fully processed

Fig. 3: Constructing the VOMM context tree of order 2 from *HWGHWSHF*. An occurrence counter is present in every node. Fig. (a) shows the context tree after the very first three locations were processed. In (b) the context tree is fully build. Subsequently every time a branch has reached the model order n, a new branch is created adopting the last two locations q_{t-1}, q_t and adding the next location q_{t+1}.

q appearing after the context s of the model order n is calculated by the count of that sequence divided by all symbols appearing after the context and all of their counts:

$$\tilde{P}(q|s) = \frac{C(sq)}{|\Sigma_s| + \Sigma_{q' \in \Sigma_s} C(sq')} \tag{1}$$

$C(sq)$ denotes the occurrence count of symbol q following sequence s and Σ_s is the set of symbols following a context s. The $|\Sigma_s|$ in the denominator acts as a Laplace additive smoothing factor to account for possible symbols appearing after the context that have not been observed in training. If q_1 with a count of two, $C(sq_1) = 2$, is the only successor of context s, the conditional probability is not 100%, but $P(q_1|s) = \frac{2}{1+2} = 67\%$. The remaining 33% are allocated for yet unseen symbols.

For symbols that have not appeared after the context s, the PPM subsequently escapes to shorter contexts by removing the first element of s until q can be observed or an empty context has been reached. The conditional probability for an empty context is defined as $P(q|\epsilon) = \frac{1}{|\Sigma|}$. In general, a PPM allocates a probability mass of $P(escape|s)$ to all symbols that have not been observed after the context s and distributes the remaining proportion between the observed successors to the context.

$$\tilde{P}(escape|s) = 1 - \sum_{q \in \Sigma_s} \tilde{P}(q|s) = \frac{|\Sigma_s|}{|\Sigma_s| + \Sigma_{q' \in \Sigma_s} C(sq')} \tag{2}$$

The complete expression for the conditional probability for an arbitrary symbol q is therefore:

$$P(q|s) = \begin{cases} \tilde{P}(q|s) & q \in \Sigma_s \\ \tilde{P}(escape|s) * P(q|sub(s)) & q \notin \Sigma_s \end{cases} \quad (3)$$

$sub(s)$ is the sub-sequence that results from removing the first element of s.

Another big advantage of this approach is the low complexity. The VOMM context tree can be learned in $O(g)$ time, where g is length of training data.

D. Context Tree

For spatio-temporal predictions the mobility model requires an extended tree data-structure for different temporal information. Yet, a person's mobility behavior is governed by many more factors than just the previous $|s|$ locations. Especially temporal features like the usual arrival time, duration of stay at a location, departure time and travel time to the next location are valuable insights when predicting the most probable next location and departure time. Thus, the spatial context tree is extend with these additional temporal features. Every location node within the tree has several temporal trees as attributes, as shown in fig. 2. The spatial context tree determines the basic structure of the entire context tree. The other features can be regarded as supportive properties to the location node.

1) Arrival Time: Arrival time trees store the time of occurrences in a calendar like structured way. To any arrival time node there is a departure time tree of the same structure associated. This way for every arrival time at a location, the most probable departure time can be derived from the associated departure time tree. The first layer in fig. 4a divides the week in weekday and weekend. It is further sub-divided into day of week, hour and 10-minute time slots to account for finer grained patterns and departure time predictions. The location node *Shopping* in fig. 4a has been visited 56 times, which implies also 56 arrival times as shown in the calendar time tree's root node. The tree shows the shopping patterns of Monday and Thursday around 18:00 and Saturday at about 15:30 easily comprehensible. The departure time trees that are attached to every arrival time node are omitted for clarity.

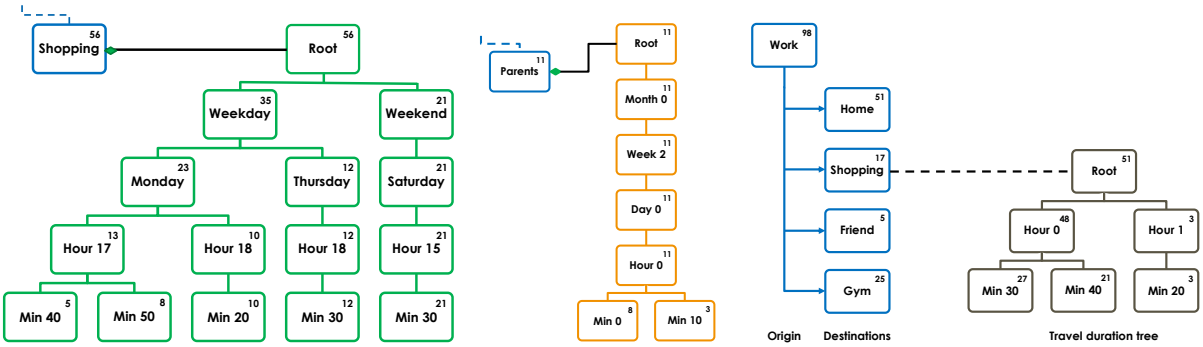
In contrast to the location tree, the paths through the tree do not describe a sequence but different layers of abstraction. This plays an important role, in particular when *escaping* from unobserved data points. The probability for observed nodes is still computed as in the location tree eq. (2). In case of *escape* the predictions lead to the parent node and not to a path of shortened context. For the arrival time tree in fig. 4a, this means a probability estimation for the unobserved time Monday 17:20 has to escape to the *Hour 17* node, which aggregates all the departure times of its children nodes. So, even without having observed an arrival at 17:20 before, the most likely departure time is estimated based on the 17:40 and 17:50 arrival time nodes's most probable departure times which are summarized in the *Hour 17* node.

2) Dwell Time: The dwell time tree encodes the duration that someone spends at the location it is attached to. Structurally, it is similar to the arrival time tree. As the dwell time at a location is normally limited to a couple of hours, the hour and minute-slice levels are information. It is evident that the regular working time is about 8 hours with some days, maybe Fridays, around 6 hours. The dwell time information is especially helpful with irregular arrival times but mostly constant resting duration. Take, for instance, as assembly line worker, who has varying work start times during the week and different schedules assigned every month. The most probable departure time is difficult to derive based on the very noisy arrival and departure time trees. Assuming fixed shift duration it is straightforward to estimate the departure time based on the current arrival time. For a Monday shift start at 8:00 the most likely departure time based on the dwell time tree will be predicted to be around 16:00. For a late shift on Tuesday starting at 14:00 the same mechanism will predict 22:00 as departure time. If the shifts turn the following week, the same mechanism will still work, as it is not bound to the day of the week like the calendar like time trees.

3) Periodicity: Periodicity describes by which locations are visited. Similar to the dwell time the periodicity tree encodes a duration. Yet, the time is measured between the different arrival times at the associated location. As the time between visits of the same location can greatly vary, the periodicity tree incorporates additional layers for months, weeks and days passed since the last occurrence. The periodicity information allows to represent time-relative patterns, such as going to the gym every second day. Based on the time since the last visitation of a location, the periodicity tree estimates the probability of going there next. Fig. 4b illustrates the pattern of visiting your parents every other weekend. If it has only been a week since the last visit, predictions based on the tree will result in a very low probability, as there are no counts for the *Week 1* node. Yet, if a next location prediction is triggered on a Saturday morning and the estimated arrival time at the *Parents* location is close to the two weeks after the location's timestamp of the last visit, the periodicity based prediction will result in a very high probability.

4) Travel Duration: The travel duration is based on the historic data, how long it took the user previously to get from one location to another. The initial travel duration can be estimated by using a routing service with the coordinates of the two locations. For each destination candidate, the travel duration is added to the estimated departure time from the current location to come up with an arrival time prediction. This estimated arrival time is then checked with the destination candidate's periodicity and arrival time tree for its likelihood.

While the previous trees are directly assigned to a location node, or in case of the departure time tree to an arrival time node, the travel duration trees are embedded in a mapping from origin to destination location. Thus, every instance of a location, independent of its sequence position in the location tree, shares the same travel duration trees. Fig. 4c shows the



(a) Arrival time tree (b) Periodicity tree (c) Travel duration mapping

Fig. 4: Each tree contains different temporal context information and is attached independently to one location node. Fig. (a) depicts an arrival time tree for location node *Shopping*. The similar structured departure time trees, that are attached to every arrival time node, are omitted for clarity. Periodicity tree (b) contains temporal context for visiting parents every other weekend. Travel duration mapping (c) from origin *Work* to all its successors. Travel duration trees are added to every origin-destination mapping, for instance, *Work* $\rightarrow$ *Home* mapping.

travel duration mapping from origin *Work* for all locations. Each origin-destination pair maps to a travel duration tree that is identical in structure to the dwell time tree. The example shows the commuting time from *Work* to *Shopping* of normally 30 to 40 minutes with some outliers at an hour and 20 minutes, for instance, due to bad traffic.

E. Model Update

The model is based on the sequence of stops. A stop describes the visit of a location and consists of:

- a unique location identifier
- longitude and latitude
- street address
- arrival timestamp
- departure timestamp

As soon as the user arrives at a new location, a new stop is created. The stop detection is part of the pre-processing (clustering algorithm) and, thus, not further described. At this time the departure time is not yet known and the departure time field remains empty. The departure time and dwell time trees can only be updated after the user left the location and the stop's data is completed. The remaining trees are refreshed upon arrival at the location. First, the location tree is updated. The new stop is added to the existing stop sequence that contains the previous stops. Together, they form the location context. In fig. 3a, the new stop is *Gym*, which is added to the existing sequence *Home* $\rightarrow$ *Work*. For the first element of the location context, the predecessor node is identified and the algorithm decides whether to add the new node as a child or in case the predecessor is already a leaf node to start a new tree path from the root. For every stop we check if the location node already exists in that layer of the tree. If this is not the case, a new location node is added as child to its predecessor. Otherwise, the existing node is updated. In both cases, the node's access counts from the root node to the current location node are updated. In the arrival time tree, the arrival timestamp is transformed into the different levels of abstractions given by the tree structure. As with the location tree we check if the temporal information is already encoded in the tree and otherwise new tree nodes are created. The counters of the entire path from the arrival time tree root to the minute node is incremented to encode the arrival time's probability. Next, the periodicity tree is updated in a similar way by calculating the duration between the current stop's arrival time and the timestamp of the location's last visit. As soon as the user leaves the current location and the departure time is known, the dwell time tree is updated by calculating the duration between the stop's arrival and departure time and incrementing the counter of the corresponding tree nodes. Further, the departure time trees for each node associated with the current stop's arrival time are updated. The entire update process is repeated for every stop in the location context.

F. Merging of individual Features

The next location prediction is based on the probabilities of spatio-temporal features. For the current location, all other known locations are potential successor candidates. Every candidate is evaluated regarding the following features:

- location
- arrival time: calendar based
- arrival time: calendar based, evaluated by periodicity
- arrival time: dwell based
- arrival time: dwell based, evaluated by periodicity

As the correct candidates for previous location transitions are known, a classifier we trained on the mentioned features to come up with a model that identifies the most promising next location candidate. For evaluation, the following classifier and regression algorithms will be considered using the implementation and default parameters from Weka[1]:

- Decision Tree
- Random Forest
- Naive Bayes

[1] `http://www.cs.waikato.ac.nz/ml/index.html`

- AdaBoost
- Support Vector Machine (SVM)
- Random Forest Regression
- Linear Regression
- SVM Regression

Besides the next stop's location, also the departure time is of interest. The PPM comes up with two departure times, based on the following features:

- departure time: calender based
- departure time: dwell based

A likelihood is associated with each of these times, which allows a machine learning algorithm for classification to be trained on these two temporal features to come up with the most probable departure time. The five classifier mentioned earlier in this section were used for classification. In this case regression algorithm were not used for departure time prediction. For finding the most probable next location candidate the classification is not enough, as more than one or no candidate at all could be classified *true*. Therefore, the classification results are further ranked by their classification confidence of their affiliation of the specific class to identify the most probable candidate. The same applies for departure time candidates. A classifier weights all sub-probabilities, given by the VOMM's spatial and temporal context information, and builds an individual model to obtain a joint probability for each next location candidate and the departure time.

IV. EVALUATION

A. Database

The used database for evaluation contains GPS logs recorded by a co-worker in the time span from 1 August 2015 until 1 December 2015. 369 stops have been recorded in that time at 98 unique locations, mostly centered around Munich but also include trips to northern Germany and a vacation in Portugal. Note that the *first* stop at each of the *98 unique locations* (27% of the training data) of the 369 stops in total have been impossible to predict. A stop at a new location is predictable only after the first occurrence of this location in the recorded data. Thus, the evaluation of the prediction accuracy is made on the 271 predictable stops. Once a stop is completed – a stop is completed after the user has left this stop – the habit model is updated and trained on all previous stops.

B. Prediction Performance

To measure the prediction performance we need update the VOMM with the first stop of the 369 stops. To simulate that this is the current location at that the user just has arrived the departure time is deleted to be able to predict the next location and departure time.

The prediction result is the most probable next location ranking. The deviation from the correct result is specified in the number of ranking positions. The departure and arrival times deviation is calculated in minutes.

Additionally, all candidates that were considered as successor locations are logged with their different spatio-temporal feature values and labeled, as shown in table I. These additional features are one of our contributions. This training data result shows two predictions. The first prediction has two location candidates and the second prediction has three location candidates. Predictions are separated by a double empty row. The column *correct candidate* indicates whether next locations candidate is the correct next location or not respectively by $\{1, 0\}$. Self transitions are not allowed in this model. Thus, all identical origin and destination locations are filtered. The correct location candidate can also occur in differently escaped levels, e.g. *Work* $\rightarrow$ *Gym* $\rightarrow$ *Home*, *Gym* $\rightarrow$ *Home* and *Home*. To

TABLE I: Next location candidates single feature probabilities per prediction. Two predictions are shown separated by a double empty row. The first prediction has two location candidates and the second prediction has three location candidates. The last column indicates whether next locations candidate is the correct next location or not respectively by $\{1, 0\}$.

locationP	dwellP	calendarP	dwell Periodicity P	calendar Periodicity P	correct candidate
0.12500	0.00010	0.00010	0.00001	0.00001	1
0.06250	0.00010	0.00010	0.00001	0.00001	0
0.04167	0.00050	0.00050	0.00001	0.00001	1
0.04167	0.00010	0.00010	0.00001	0.00001	0
0.01563	0.00050	0.00050	0.00001	0.00001	0

The last evaluation step is to set the correct departure time for the current stop, which simulates the user leaving the location. The evaluation then continues with the next step and incorporates all previous knowledge. The VOMM improves its predictive performance by increment the relevant counts of location node visits. With regard to our first contribution, the classifiers which predict the most probable next location and departure time based on the VOMM's output gets better through the growing number of training data (see fig. 5).

Table II shows a benchmark of different classifiers and regressors based on the training set for next location candidates. This approach was further compared with regression models that were trained on the same data. Only the class attribute was changed to the numeric values 0 and 1 instead of *false* and *true*. The prediction results have been verified with 10 folds cross validation. All evaluated classifier except the Naive Bayes perform with almost the same accuracy of 98% (see table II). The Naive Bayes gains an accuracy of only 95%. The SVM with a polynomial kernel, in this case, is identical to a Zero-R classifier which simply predicts the majority class. Even with using a cost sensitive evaluation and increasing the costs for false negatives the number of true positives could not be increased without dramatically increasing the number of false positives with the other classifiers.

Table III shows the next location *prediction* benchmark of all classifiers and regressors that were successively retrained after each stop. The Random Forest classifier based on 100 trees, each trained on a subset of $int(log_2(predictors)+1) = 3$ features, achieves the best perfect prediction performance

TABLE II: Next location candidates classification accuracy and regression (Regr.) error

Classifier	Accuracy-%	Relative absolute error-%	Root relative squared error-%
Decision Tree	98.7085	–	–
Naive Bayes	95.3155	–	–
Random Forest	98.6519	–	–
AdaBoost	98.5499	–	–
SVM	98.4649	–	–
Random Forest Regr.	–	60.3059	82.7705
Linear Regr.	–	82.8586	83.0316
SVM Regr.	–	50.8814	100.697

TABLE III: Prediction accuracies of different classifier and regression (Regr.) models with respect to different deviation (Dev.) counts for next location prediction. Perfect prediction means no deviation.

Classifier	Perfect Prediction	Dev. of 1	Dev. of 2	Dev. of 3	Dev. > 3
Decision Tree	58%	12%	3%	3%	24%
Random Forrest	**60%**	10%	2%	4%	24%
Naive Bayes	55%	13%	5%	2%	25%
AdaBoost	58%	13%	5%	5%	**19%**
SVM	55%	13%	5%	2%	25%
Random Forest Regr.	58%	12%	4%	3%	23%
Linear Regr.	56%	13%	3%	4%	24%
SVM Regr.	57%	13%	4%	2%	24%

with 60%. One drawback of the Random Forest is its computational complexity. Compared to the similarly performing decision tree with a prefect prediction accuracy of 58%, the training and prediction of the 369 models takes 288 seconds (Random Forest) vs. 15 seconds (Decision Tree) on an average personal computer. Another strong contender is the AdaBoost boosting algorithm with Decision Stumps (one-level decision trees) as classifiers. Especially if small deviations from the perfect predictions are acceptable, it performs significantly better than all other classifiers with only 19% of the predictions deviating more than 3 spots from the correct solution. The computation time is with 25 seconds also on the lower end of the spectrum. The regression models' performance is not superior to the classifiers.

In spatio-temporal next location prediction, not only the correctly predicted next location is of relevance but also the predicted departure time to the next location, that is also one of our contributions. Table IV shows the predicted next location departure time benchmark results for the models retrained after each stop. In this benchmark the Naive Bayes performs best. The deviation of the predicted departure time with Naive Bayes classifier from the actual departure time is broken down in table V. These numbers include the departure from locations that are visited for the first time.

In fig. 6 the performance of the different features is outlined. The purely location based prediction determines the correct next location with a maximum deviation of 3 twice as likely as purely time based predictions based on any of the temporal features. Another contribution is to merge spatial and temporal prediction probabilities based on different features with the AdaBoost classifier into one

TABLE IV: Departure time prediction classification results.

Classifier	Accuracy-%
Decision Tree	68.29268
Random Forest	65.58265
Naive Bayes	69.37669
AdaBoost	68.02168
SVM	65.58265

TABLE V: Departure time prediction deviation results compared with actual departure time.

Deviation	$\pm$ 10 min	$\pm$ 30 min	$\pm$ 1 h	$\pm$ 2 h	$>$ 2 h
Value	10%	18%	15%	12%	45%

combined model the accuracy of the spatial prediction is improved by 9% which leads to the total performance of 219 out of 271 or 81%. One of the biggest advantages of the used PPM model is that it delivers accurate predictions right away without requiring a long learning process in comparison to other approaches, for instance, DBN [18] or HMM [19]. Fig. 5 shows how the prediction deviation develops over time. Even without much historical data, less frequent locations can be predicted accurately by our model. It is evident that the deviation is only influenced by phases of irregular behavior and does not depend much on the training duration.

V. CONCLUSION

In this paper we proposed a spatio-temporal next location prediction approach for a predictive ridesharing assistant based on the PPM algorithm. The main idea is to built a VOMM based on the location history and enhance the model with temporal context information to predict arrival and departure time with fine-grained resolution as well as the next location. Further main parts of this work are the analysis of the variety of temporal context information trees and a model for merging different predictions based on different features to obtain a joint probability for predicted next location including arrival and departure time. The output of this model can be used for the predictive ridesharing assistant to match

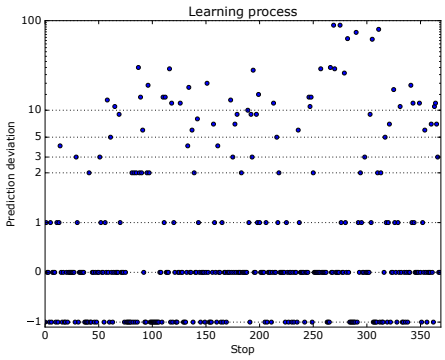


Fig. 5: Evolutionary prediction benchmark of VOMM over time. It shows the deviation of predicted next location after every model update. The model is updated after each stop. The negative value encodes locations that have not been seen before. Thus, a correct prediction is not possible.

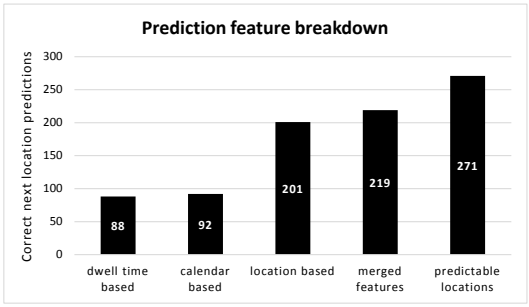


Fig. 6: Next location prediction performance (deviation of 3) of different context information.

driver and rider with the same next location and departure time. Ridesharing is one of the low hanging fruits to ease urban congestion if managed properly. In this paper we have shown how to antagonize, for instance, urban congestion using predictive ridesharing with little effort to the participants.

For evaluation purposes the model is trained subsequently. Each previous stop candidate is labeled as wrong or correct candidate. The output of the model is a ranking of the most probable next location candidates. The best classifier to predict the next location is Random Forest with an accuracy of 60% without any deviation. AdaBoost achieves an accuracy of 81% for next location prediction with a maximum deviation of 3. The temporal features are used to derive the most probable departure time. In this challenge we got 69% accuracy picking the most reliable temporal prediction with a Naive Bayes classifier. This model predicts the correct departure time time within $\pm$ 10min in 10%, $\pm$ 30min in 18% and $\pm$ 1h in 15% of the cases. The VOMM possesses several advantages over FOMMs, for instance, a good prediction accuracy and the ability to detect patterns of different lengths. Thus, it's possible to detect more patterns without the need to increase the model order. In contrast to a VOMM the FOMM must have a higher model order to detect more patterns, which also implicate a higher amount of training data is needed. To deal with the cold-start and the zero-frequency problem the VOMM allocates a certain probability share for symbols that have not been contained in the training data. Adding more features causes an exponential grow of the transition matrix when using a fixed order model and the need of more training data. The VOMM's variable length pattern detection in conjunction with the escaping strategy allow the addition of extra features without increasing the training data demand which gives it a strong extensibility. High costs for model training can limit live tracking of a person's continuously changing lifestyle. The VOMM's quick trainability can process model updates within $O(n)$ and is therefore well suited for live tracking.

Our model's strong spatio-temporal prediction performance will be further improved by future work by a) implementing a forgetting factor to weight recent stops stronger and decay older, less frequently visited stop over time, b) advanced travel duration determination using a real time routing service to predict the most likely travel duration at the time and c) additional context information, which also influence mobility patterns, for instance, weather information and semantic classifications of a location.

REFERENCES

[1] G. Martine, A. Marshall, *et al.*, "State of world population 2007: unleashing the potential of urban growth," in *State of world population 2007: unleashing the potential of urban growth*. UNFPA, 2007.

[2] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," in *Proceedings of the 9th ACM conference on embedded networked sensor systems*. ACM, 2011, pp. 82–95.

[3] M. Papandrea and S. Giordano, "Location prediction and mobility modelling for enhanced localization solution," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 3, pp. 279–295, 2014.

[4] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *Data mining (ICDM), 2012 IEEE 12th international conference on*. IEEE, 2012, pp. 1038–1043.

[5] D. Lian, X. Xie, V. W. Zheng, N. J. Yuan, F. Zhang, and E. Chen, "Cepr: A collaborative exploration and periodically returning model for location prediction," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 1, p. 8, 2015.

[6] J. Ye, Z. Zhu, and H. Cheng, "What's your next move: User activity prediction in location-based social networks," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 171–179.

[7] S. Lee and K. C. Lee, "Context-prediction performance by a dynamic bayesian network: Emphasis on location prediction in ubiquitous decision support environment," *Expert Systems with Applications*, vol. 39, no. 5, pp. 4908–4914, 2012.

[8] J. Fukano, T. Mashita, T. Hara, K. Kiyokawa, H. Takemura, and S. Nishio, "A next location prediction method for smartphones using blockmodels," in *Virtual Reality (VR), 2013 IEEE*. IEEE, 2013, pp. 1–4.

[9] J. J.-C. Ying, W.-C. Lee, and V. S. Tseng, "Mining geographic-temporal-semantic patterns in trajectories for location prediction," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 1, p. 2, 2013.

[10] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "Nextplace: a spatio-temporal prediction framework for pervasive systems," in *International Conference on Pervasive Computing*. Springer, 2011, pp. 152–169.

[11] B.-K. Ang, D. Dahlmeier, Z. Lin, J. Huang, M.-L. Seeto, and H. Shi, "Indoor next location prediction with wi-fi," in *The Fourth International Conference on Digital Information Processing and Communications (ICDIPC2014)*. The Society of Digital Information and Wireless Communication, 2014, pp. 107–113.

[12] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*. ACM, 2012, p. 3.

[13] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order markov models," *Journal of Artificial Intelligence Research*, pp. 385–421, 2004.

[14] H. Bapierre *et al.*, "Context specific next location prediction," Ph.D. dissertation, Technische Universität München, 2014.

[15] H. Bapierre, G. Groh, and S. Theiner, "A variable order markov model approach for mobility prediction," *Pervasive Computing*, pp. 8–16, 2011.

[16] J. Rissanen *et al.*, "A universal data compression system," *IEEE Transactions on information theory*, vol. 29, no. 5, pp. 656–664, 1983.

[17] A. Shmilovici and I. Ben-Gal, "Using a vom model for reconstructing potential coding regions in est sequences," *Computational Statistics*, vol. 22, no. 1, pp. 49–69, 2007.

[18] J. Petzold, A. Pietzowski, F. Bagci, W. Trumler, and T. Ungerer, "Prediction of indoor movements using bayesian networks," in *Location- and Context-Awareness*. Springer, 2005, pp. 211–222.

[19] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden markov models," in *Proceedings of the 2012 ACM conference on ubiquitous computing*. ACM, 2012, pp. 911–918.