Real-Time Public Transportation Prediction with Machine Learning Algorithms

Dancho Panovski and Titus Zaharia
IP Paris, Télécom SudParis, ARTEMIS Department, UMR CNRS 5157 SAMOVAR, Évry, France
E-mail: {dancho.panovski, titus.zaharia}@telecom-sudparis.eu

Abstract—As part of Intelligent Transportation Systems (ITS) public transportation plays a critical and essential role for the mobility in every modern city. In this paper, we introduce a novel method for the real-time prediction of bus arrival times in the various bus stops over a given itinerary. The proposed approach exploits machine and deep learning algorithms, including optimal least square (OLS) linear regression, support vector regression (SVR) and fully-connected neural networks (FNN). The experimental results obtained show that the FNN approach outperforms, in terms of mean absolute prediction error, both SVR (by 7.62\%) and OLS (for 15.74\%).

I. INTRODUCTION

Today, the concept of modern and smart cities cannot be envisaged without mentioning the ITS (Intelligent Transportation Systems) [1]. Since the beginning of this important domain, in the early 1970’s, there has been a lot of progress, innovation and development. In recent years, the emergence of ITS-related applications has been catalyzed by the rapid growth of available data, as well by the increase of the computational power that is naturally needed to process such data. Both public institutions [2] [3] and private companies [4] [5] are proposing various solutions for gathering and processing of its-related data. Among the various use cases considered, let us mention trip planning, daily commute, real-live traffic information, trip cost, multimodality and so on. Usually as end results, the corresponding services are provided in a form of phone app or web portal, where the users can get the desired information, mostly for free.

II. RELATED WORK

Acquiring and exploiting traffic-related data is neither easy nor cheap, due to some government constrains, privacy issues and legislations. In order to overcome such difficulties while ensuring an adequate level of realism, we have created our own, artificial data, with the help of traffic simulation platforms. There are two main branches of traffic simulators, including microscopic [6] and macroscopic [7] approaches. While macroscopic simulators are focusing on more global values, like global traffic density or vehicle distribution, the microscopic simulators focus on each individual vehicle in the system, which is presented and tracked in details with the various attributes (velocity, speed, location, time …). Among the various traffic simulators available, we can encounter both open-source [8] [9] [10] and commercial solutions [11] [12] [13]. In this paper, we have adopted the microscopic SUMO (Simulation of Urban, MOBility) simulator [14], which is an open-source software with big developer community. SUMO offers the advantage of a microscopic, multi-modal traffic simulator that can simulate various scenarios with different type of traffic data and produce end result-data in readable formats like XML.

As our goal is to predict the bus arrival time in different location (e.g. bus stops) over a given itinerary, we have considered ML (Machine Learning) algorithms. Among the classical ML models used for traffic prediction purposes, let us mention OLS (Optimal Least Squares) [15], SVRs (Support Vector Regressors) [16], k-NN (k-Nearest Neighbors) [17], Random Forest [18]. In a different meaner, DL (Deep Learning) [19] [20] techniques have become more popular, thanks to the rapid growth of the GPU units. Deep Learning techniques like BPNN (Back-Propagation Neural Network) [21] were used for predicting the bus traffic. A CNN (Convolutional Neural Network)-based algorithm [22] has been used to learn the traffic as an image. Short-term forecasting for on-demand ride services were predicted successfully with the LSTM (Long-Short Term Memory) [23] method. As an emerging trend, let us mention the GNN (Graph-based Neural Networks) approaches [24].

In this paper, we are going to focus on real-time prediction of bus arrival time at the desired station, while including the real-time traffic density map on the bus itinerary. The contribution of the article is twofold. First, it concerns the modeling and aggregation of the data in a specific data structure, so-called TDM (Traffic Density Matrix). Second, it concerns the exploitation of this structure for prediction purposes, with the help of ML techniques.

The rest of the paper is organized as follows. Section III describes the simulation scenario in details. In Section IV the TDM (Traffic Density Matrix) structure is described with the resulting data model. Experimental results are presented in Section V. Finally, Section VI concludes the paper and opens perspectives of future work.

III. SIMULATION SCENARIO

The simulation scenario has been developed for two bus lines in city of Nantes, France, namely bus 79 (Beausejour - Orvault) and bus 89 (Beausejour – Le Cardo), both illustrated in Fig 1. These are two real bus lines that are situated in same geographical area inside the city of Nantes. The fact that both bus itineraries are in the same geographical areas makes them well-suited for simulation, since, with one simulation it is...
possible to produce data for both bus lines. All the necessary data used for creating the bus itineraries and bus stop locations were recovered from TAN (Transport de l’Agglomération Nantaise) service [25] and Nantesmetropole [26] – open data repository, while the map was loaded from OSM (Open Street Map) [27].

The simulation scenario was executed with the following parameters, visually represented in Fig 2. First, the 2D map was converted and imported in SUMO from OSM. Since SUMO does not support importing the public transportation from external sources, all the bus itinerary and bus stop were manually added. The total number of simulations was set to 4000.

Each simulation contains 3 bus runs at different start time period (3000, 7000, 11000) in seconds. After further analysis it was decided that each bus run will be considered as separate run with starting time t=0 seconds, which led to a total number of 12000 runs.

The global macro parameter that controls the simulation is the total number of vehicles inserted into the system. Each consecutive simulation will differ from the previous one from the number of vehicles that are inserted into the system, this parameter is within range of [11000, 18000] interval (number of vehicles). Vehicles are inserted into the system in timely manner (Fig 2), so the stability range can be established between 2500 and 15000 seconds. For each vehicle inserted a shortest path algorithm – Dijkstra [28] is calculated from O/D (Origin/Destination) matrix.

The simulation also involves the following three static parameters:

1. The number of pedestrians, set to 3600 and intended to increase the realism of the simulation, which also takes into account the pedestrians traffic lights.
2. The simulation end time was set to 19000 seconds; in order to make sure that all the vehicles are out of the system.
3. The time spent by at the bus at each bus stop was set to 20 seconds.

The whole simulations scenarios and machine learning algorithms were executed on the PC with the following configuration: OS - Windows 10 Pro, RAM - 32GB, CPU - i7 6700HQ (3.40 GHz), GPU - NVidia GTX 1080Ti. The Scikit-learn [29] and PyTorch [30] machine learning frameworks were used and Python has been used as coding environment.

Let us now introduce the data development model and the TDM (Traffic Density Matrix).

IV. DATA MODEL – TRAFFIC DENSITY MATRIX

The initial goal is to improve the public transportation system by predicting bus arrival time at desired station while taking into account the real-time traffic situations on the whole bus itinerary.

In order to do so, the traffic situation at each measurement station is represented with the help of the TDM (Traffic Density Matrix), illustrated in Fig. 3.

Measurement stations are points in space that detects and counts how many vehicles pass at a certain period of time. The TDM(i,j) entry thus provides the number of vehicles that are present in the i-th bus station at time j. In real-life scenarios such sensors are most often the loop detectors, but other sensors can perform similar task, like camera for vehicle counting. For the convenience of the whole simulation, each bus stop location was considered as a measurement station. In this particular scenario, the detection radius around the considered measurement station was set to 100 meters. This representation can be interpreted as a traffic density matrix evolving over time, which is traversed by the considered buses, as described in greater details in our previous work [31].

In contrast with our previous work [31], where a global approach, taking into account the totality of bus stops over a given itinerary, has been considered for prediction purposes, here we propose a different, per bus approach, which leads to a
significantly less complex representation. The objective is to inform the user, in real-time, of the arrival of the buses in his location. To this purpose, we estimate, for a given bus stop provided as input, the expected arrival time of the next bus.

In order to do achieve this goal, we sample the TDM and construct vectors that are further used for prediction purposes. Such a vector is composed of the \( N_{\text{stops}} \) values from the TDM matrix that correspond to a given time \( R_{\text{time}} \), which represents the current time.

In total, for the whole bus itinerary there are \( N_{\text{stops}} \) bus stops. In our experiments, we have focused on bus line 89, which included 36 bus stops, since its prediction is more difficult due to the presence of traffic jams [31]. We have chosen 5 different bus stop stations, evenly sampling the itinerary, and denoted as \( R_{\text{stop}}: 11, 16, 21, 26 \) and 31. These are the bus stops that will be used to predict bus arrival time in seconds, Fig. 4.

![Fig. 4. Predicting bus arrival time at desired station](image)

In order to construct the training vector, set, for each current time, randomly chosen and denoted by \( t_{\text{crt}} \), we determine the location of the closest bus \( C_{\text{stop}} \) prior to the current bus location \( R_{\text{stop}} \). During the construction of the training data set, we make sure to retain only valid trials, i.e., values for which in the simulations, at the current time there exists a bus prior to current bus location (Fig. 4).

Finally, the input vector (Fig. 5) includes the TDM \( N_{\text{stops}} \) values at the current time \( t_{\text{crt}} \), the current bus stop \( R_{\text{stop}} \) and the closest bus stop \( C_{\text{stop}} \). The output value is simply the expected arrival time from the ground truth. For each predicted bus stops, 10 bus runs are extracted from TDM, so:

\[
12000_{\text{sim}} \times 5_{\text{stop}} \times 10_{\text{runs}} = 600000 \quad (1)
\]

The total number of generated train/test dataset inputs is 600000.

**V. EXPERIMENTAL RESULTS**

For comparative methods three different machine learning algorithms were considered: OLS (Ordinary Least Squares), SVR (Support Vector Regression) with 2nd degree polynomial kernel and FNN (Feedforward Fully Connected Neural Network).

For the learning stage, obtained data has been randomly split: Training dataset 80%; Tests dataset 20%, from which 10% used as validation, and 10-fold cross validation was performed for determining the randomness of the data.

The architecture of the proposed neural network is presented in Fig. 6.

![Fig. 6. Feedforward fully connected neural network with 3 hidden layers](image)

This is 4-layer Feedforward Fully Connected Neural Network with 3 hidden layers of size \((20 – 200 - 20)\) and ReLU as activation function. MSE (Mean Square Error) was used as loss function and SGD (Stochastic Gradient Descent) algorithm (with learning rate of 0.001) for optimization has been considered for the learning stage (Fig. 7/C).

The obtained prediction results are summarized in Table 1, while Table 2 presents the results from our previous work [31] (different data structure) vs. current performance.

### Table 1

<table>
<thead>
<tr>
<th>MACHINE LEARNING ALGORITHMS PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11)</td>
</tr>
<tr>
<td>OLS</td>
</tr>
<tr>
<td>SVR</td>
</tr>
<tr>
<td>FNN</td>
</tr>
</tbody>
</table>

\(g\text{MAE} = \text{Global Mean Absolute Error}, \ c\text{TIME} = \text{Computational Time in (s)}, \ OLS = \text{Ordinary Least Squares}, \ SVR = \text{Support Vector Regression}, \ FNN = \text{Feedforward Fully connected Neural Network}\)

### Table 2

<table>
<thead>
<tr>
<th>PREVIOUS WORK VS CURRENT WORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT PERFORMANCE</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>OLS</td>
</tr>
<tr>
<td>(g\text{MAE})</td>
</tr>
<tr>
<td>(c\text{TIME})</td>
</tr>
</tbody>
</table>

\(g\text{MAE} = \text{Global Mean Absolute Error}, \ c\text{TIME} = \text{Computational Time in (s)}, \ OLS = \text{Ordinary Least Squares}, \ SVR = \text{Support Vector Regression}, \ FNN = \text{Feedforward Fully connected Neural Network}\)

From Table 1, we can observe that FNN produce better results (15.76 % improvement then OLS and 7.62 % then SVR) in all bus stops except in one instance (bus stop 11) where SVR has a MAE of 43.5 against 54.8 of FNN. MAE (Mean Absolute Error) is presented in seconds and represent the actual average time of the bus arriving at the desired station. The fastest method is OLS but as a penalty its prediction performance is the worst.
In Fig. 7/A histogram show the MAE for every method at each bus stop station. While in Fig. 7/B predicted scatter chart values of all errors are presented for the bus stop 26, where the predicted performance between 3 different methods can be clearly seen. In Fig.7/D shows the trend (MAE performance in seconds) for bus arrival time at bus stop station from 11th to 31th.

VI. CONCLUSION AND PERSPECTIVES

In this paper we have, considered the issue of real-time prediction of bus arrivals in the various bus stops over a given urban itinerary. To this purpose, we have taken into account the current traffic conditions, modeled with the help of the traffic density matrix. Concerning the machine learning techniques considered, we have retained and compared linear regression, support vector regressors and an original feed-forward neural network model. Experimental results obtained show promising performances, with global average prediction errors of 163 seconds when the neural network approach is applied.

In order to improve the accuracy of the system, our future work will concern adopting more complex deep learning techniques such as LSTMs or GNNs, and also exploring some other possibilities of data models.

REFERENCE

[12] "Paramics, Quadstone. &quot;Paramics microscopic traffic simulation software.&quot; [Online]. Available: https://scholar.google.fr/scholar?q=Paramics%2CQuadstone.&quot;+%22Paramics+microscopic+traffic+simulation+software%22+%22%20Par amics+microscopic+traffic+simulation+software%22+%22%282012%29.& btmG=hl=en&as_sdt=0%2C5.