Travel time prediction

Leonid Dashko
dashko@ut.ee
Institute of Computer Science, University of Tartu
Supervisor: Amnir Hadachi

April 29, 2017

I. Introduction

Travel time information plays an important role in Transportation Systems, in particular in the fields of Advanced Traveler Information Systems (ATISs) and Advanced Traffic Management Systems (ATMSs). The forecast of travel time is one of the crucial components of these systems. Considering the perspectives of travelers the information about travel time may help in making better road choice, estimate the forecasted travel time, and reach the destination point faster by correcting the route.

Travel time prediction may also be helpful for the transportation agencies to control the traffic and reduce the congestion.

The main objective of this research is to observe existing approaches to make a travel time prediction and a try to apply Deep Belief Network (DBN) algorithm for travel time prediction.

Besides, in many states, in the USA, relevant traffic information is also posted on variable message signs (VMSs) that are placed along the highways that help road users to see the predicted travel times for better planning their trips and choosing their route of travel. [1]

The traffic data may be collected from various sources [2]:

- automatic number plate recognition systems (ANPRs)
- automatic vehicle identification systems (AVIs)
- inductive loops
- global positioning system (GPS)
- cellular geo-location
- video cameras
- ground-based radio navigation (the data is collected by communication between probe vehicles and a radio tower infrastructure)
- combined approach (the data is filtered from different sources)
- other approaches

These sources of the data may be used in ITS depending on the specific task. Particularly, the GPS data is most suitable for the travel time forecast because each GPS record has the following information: latitude, longitude, timestamp, speed (optional), other meta data (optional).

If we do not have information about road segments in the data set, we may find closest road segment and bind this GPS record to a particular road segment. This step is known as map-matching and it will be discussed later.

Additionally, there are various factors that may affect the accuracy of travel time. For example, if the travel time prediction depends on vehicle speed, traffic flow, and occupancy, then these factors will be highly sensitive to traffic accidents and weather conditions and the result from this prediction may be distorted. That is why it is very difficult to reach high accuracy in prediction. However, some patterns can still be discovered such as daily/weekly/seasonal patterns. As an example, weekly patterns produce high traffic activity during the weekends meanwhile the activity is low during the weekdays; key anomalies from daily patterns may be seen during rush hour and late night traf-

fic; seasonal patterns distinguish winter and summer traffic. Therefore the time-dependent features play a crucial role in travel-time modeling.

II. Overview of methodologies for travel time prediction

Travel time forecast is the task of regression because in the end, we should have an approximation of total journey time based on current data (e.g. the vehicle has only passed half of the trip). There is number of widely-used approaches that can be used to deal with task:

- **Linear model** - predicts the travel time of one trip departing at current time by combining the latest calculated travel time of the trip and the historical mean travel time of the same trip departing at the same time;
- **K-nearest-neighbors (KNN)** - finds the most similar historical **k** days to the current day, and takes the mean travel time at current time of that **k** days as the travel time at the current time of the present day;
- **Kalman filter** - estimates the predicted travel time and updates the prediction continually as a new observation becomes available;
- **Support vector regression model (SVR)**;
- **Gradient boosting regression tree** - combines simple regression trees;
- **Deep learning algorithms and Neural Networks (NN)** - purely supervised: Logistic Regression, Multilayer perceptron, Deep Convolutional Network (DCN); the unsupervised and semi-supervised: Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN), others;

The usage of the specific algorithm also depends on the task, for example: if we do real-time processing, then we need a fast algorithm that may produce a satisfied approximation, but for post-processing, we can use almost any of them.

III. Literature Review

In the research paper [4], the researchers explore a deep learning model named the LSTM neural network model for travel time prediction. The data set of the real travel time of 66 routes provided by Highways England were used. The data set was split into three parts: training set (80%), the validation set (10%) and test set (10%). Specifically, the travel times in this data set are 15-minute interval average travel times for each route. Here travel time prediction is defined as predicting future time periods travel times $\{x'(t+1), x'(t+2), ...\}$ from the acquired historical travel times $\{x(0), x(1), x(2), ..., x(t1), x(t)\}$.

In the experiment, the authors used LSTM neural network that has a structure of regular unidirectional (not bidirectional) recurrent neural network (RNN). The LSTM has three gates namely input gate, forget gate and output gate, which control the information flow through the cell and the neural network. Particularly, at time "t", the input of the network is the observed history data $x(t)$ and the output is the predicted future data $x'(t+1)$.

The evaluation results have shown that the 1-step ahead travel time prediction error is relatively small, the median of mean relative error (MRE) for the 66 links in the experiments is 7.0% in the test set. Meanwhile, the MRE for 2,3,4-step ahead predictions produce wrong travel-time predictions.

In the article [5], the researchers investigated the impact on predictive performance of a using local linear regression with using additional sources of traffic information. The main contribution of the paper is the extension of local linear models with higher order autoregressive travel time variables, namely vehicle flow data (vehicles/hour), speed (km/h), and density data (vehicles/km).

The data of the period of two years used in this paper ([5]) provided by UK Highways Agency (HA) from inductive loop sensors which measure vehicle speed, flow and density at specific locations. Additionally, the heterogeneity of the road segments plays an impor-

tant role when the models are evaluated. For example, relative errors become less informative (50% error on a 100m road segment is not comparable to a 50% error on a 10 km segment) that is why researchers have to put a focus on absolute errors more often.

The experiment has shown that the model of the spatiotemporal distribution of travel times separately for each link by using local linear regression can be used to optimally balance the use of historical and real-time data because the use of real-time data is very accurate for shorter journeys.

The article [3] describes the concept and application of Deep Belief Networks (DBN) to predict travel times. Researchers state that in their model used DBN as a stack of Restricted Boltzmann Machines (RBM) to automatically learn generic traffic features in an unsupervised fashion, and then a sigmoid regression was used to predict travel time in a supervised fashion.

By contrast to the conventional time prediction methods which require the training each road link separately, the DBN method can collectively train the traffic data on the entire road network all at once.

The application of DBN for time travel forecasting (the features are learned on a layer-by-layer basis) can be described in three steps:

1. The input vector is a vector of historical travel time on an observation route. Particularly, the input vector for predicting the travel time on a route at time T_{t+d} will be a vector of the form $\{T_t, T_{t-d}, T_{t-2d}, ..., T_{t-(k-1)d}\}$, where T_t is the travel time at the current time t, d is the length of the prediction interval (e.g., 15 minutes, 30 minutes, etc.), and k is the number of past observations on a route.
2. On the second step, the DBN will take care of extracting important features from the inputs by itself.
3. Once the DBN determines the features to be used, they will be sent to the third part of the prediction model, which is the sigmoid regression layer.
4. Finally, the prediction model will return the predicted travel time on a specific route at time $t + d$ as an output. Researchers used the real travel time data collected by the Caltrans Performance Measurement System (PeMS) where the traffic data are collected from over 42000 inductive loop detectors in real-time.

In this article, the researchers used the data of 9 months out of 12 months as a training set while the last three months of the data are used as testing set. The analysis of the travel time data on over five hundred different routes of the 3514-km long network, which covers the entire State of California showed that the proposed method can achieve great performance in terms of prediction accuracy. For a practical 15-minute prediction interval, the predicted travel time values are accurate to within 1 minute of the actual values on most of the routes (mean absolute percent error was up to 10%). Additionally, the model demonstrates quite a good result for the 60-minute interval where the maximum error was about 7.26 minutes on a 131-km route.

IV. Evaluation of travel time prediction algorithms

Usually, for the evaluating forecast accuracy of the model, the researchers use the following metrics most often:

- **MAE** - mean absolute error;

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  where n - total number of observations,
  y_i is the actual value of examined travel time,
  $\hat{y}_i$ is the forecaster value of travel time.

- **MAPE** (%) - mean absolute percent error;

$$MAPE(\%) = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i - \hat{y}_i}{y_i} \times 100\%$$

- **RMSE** - root mean square error;

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i}^2)}$$

V. Case study

To begin my experiment I started searching for the data set and picked the data set providing a minute by minute traffic flow for Junction 37 of the M1 ([6]) that contains information about traffic flow (vehicle count), average speed (kph), occupancy (%) by lane. Later I discovered that this data set is not suitable for travel time prediction as far as we need at least information about vehicle location on the road segment and its speed.

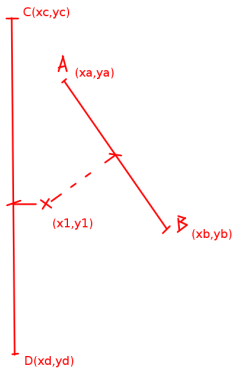

Figure 1: *Illustration of the problem of detecting the closest road segment*

Another data set was provided by my supervisor that contains ∼600,000 GPS records about the movements of 10 Estonian people during 8 days between April-March 2015. Every GPS record has such information as person_id, timestamp, latitude, longitude, the speed of the vehicle, other metadata.

On the next step, I have tried different map-matching approaches to tie every GPS point to the road segment that it is related to:

1. I used OpenStreetMap (OSM) API ([7]) to find closest road segments around 10-20-30 meters near the GPS point (to test the queries to OpeenStreetMap with projecting the results on the map I used [8] as well). It works fine, but I am limited to run a big amount of requests. I tested it on the dataset with a size of 16k points, but OSM API blocked my requests because of too many requests. I've set timeout and tried to use threading for this, it did not help inasmuch after 10-15 consecutive requests I received a blocking for "n" time period.

2. I tried to extract all road segments through OSM API within a certain range I defined the range latitude_min, longitude_min, latitude_max, longitude_max to capture all road segments inside this square. As a result, I got ∼46,500 points that represent road segments. Then, I built a graph by using "Networkx" library in Python with points of road segments. Moving further, I created a test point that represents GPS coordinate from the dataset and calculated the Euclidean distance between GPS point and other points that represent the road segments. However, this experiment did not produce a good result because some of the points were tied incorrectly.

3. I found some solutions that are written in Java, but they only correct the GPS points instead of defining a road segment which is connected to the GPS point.

4. I downloaded the map of road segments in Estonia as file "estonia.osm.pbf" and I tried to install and use OSM locally, then I might not have limitations on the number of concurrent requests to the local API. However, I did not succeed in setting up the local environment.

5. I have installed OpenJUMP, converted my points to the *.gpx format and imported them. Then I installed the RoadMatcher tool that might solve map-matching problem. Although I could not import my road segments into OpenJUMP and run the RoadMatcher (due to inability to import road segments), the OpenJUMP was a good tool for plotting a big amount of points.

To illustrate the problem of the map-matching, let us look at the figure 1. In the case of calculating Euclidean distance, the algorithm will give the closest line "AB" to the point (x_1, y_1)as a result. In fact, the line "CD" will be closer to the target point.

Finally, I came up with binding the GPS point to the road segment through finding the smallest distance from GPS point to the road segment between all road segments (each road segment is represented as an edge) which were in the graph by using this formula[9]:

$$distance(P_1, P_2, (x_0, y_0)) = \\ = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}$$

Where $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are the beginning and the end point of the road segment edge (x and y represent the latitude and longitude respectively),
(x_0, y_0) is target GPS point.

Before and after running the map-matching I tried to plot the data set. Inasmuch as the data set is quite big (∼300,000 points), Google Maps, Networkx library, Matplotlib, other tools could not plot such amount of points. The result of plotting the whole data set as points through OpenJUMP is shown on the figure 2. Additionally, I succeeded to plot ∼16,000 road segments on the map (figure 4).

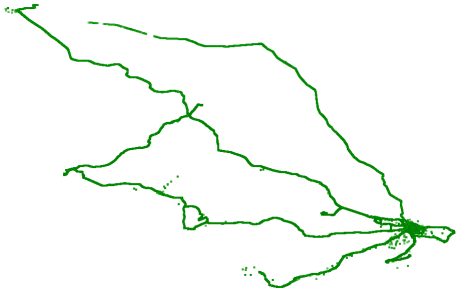

Figure 2: *The whole data set projection (only points)*

VI. Future work and conclusion

During this research, I familiarized with different approaches which are used for travel time prediction, how to evaluate the forecast accuracy of the model (section IV, worked with OSM and OpenJUMP, made the pre-processing steps for the real data. The next step for the future research will be applying the Deep Belief Network (DBN) to the real data and to validate the obtained result in comparison to other techniques for predicting travel time.

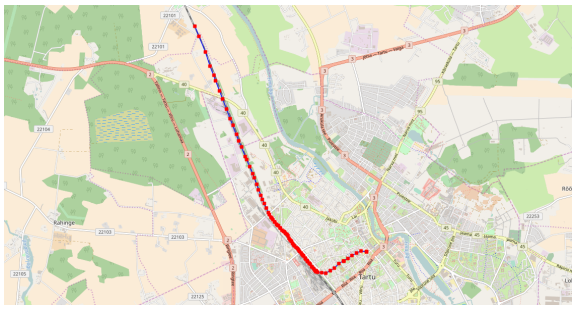

Figure 3: *Projection of 200 points from single person on the Tartu map*

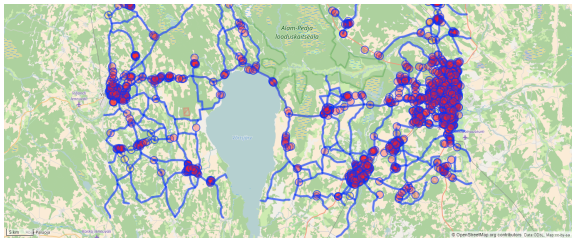

Figure 4: *Road segments for test dataset (approximately 16 thousand points)*

References

[1] Hao Chen, Hesham A. Rakha, Catherine C. McGhee "**Dynamic Travel Time Prediction using Pattern Recognition**"

[2] Office of Highway Information Management Federal Highway Administration U.S. Department of Transportation **"Travel Time Data Collection Handbook"** (Chapter 5) - 1998

[3] Chaiyaphum Siripanpornchana, Sooksan Panichpapiboon and Pimwadee Chaovalit "**Travel-Time Prediction With Deep**

Learning". Region 10 Conference (TENCON), 2016 IEEE

[4] Yanjie Duan, Yisheng Lv, Fei-Yue Wang "**Travel Time Prediction with LSTM Neural Network**". 19th International Conference on Intelligent Transportation Systems (ITSC) 2016 IEEE

[5] John Davies, Bla Fortuna, Alistair Duke, Sandra Stincic Clarke, Jan Rupnik "**Travel Time Prediction on Highways**". International Conference on Computer and Information Technology 2015 IEEE

[6] Traffic flow data set for Junction 37 of the M1 from inductive loops by the MIDAS

[7] OpeenStreetMap API

[8] Test queries to OSM online with plotting the result on the map

[9] Calculating the distance from a point to a line