Transportation Research Part C 58 (2015) 681–705

http://dx.doi.org/10.1016/j.trc.2015.02.007

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Personalised assistance for fuel-efficient driving

Ekaterina Gilman *, Anja Keskinarkaus, Satu Tamminen, Susanna Pirttikangas, Juha Röning, Jukka Riekki

Department of Computer Science and Engineering and Infotech Oulu, University of Oulu, P.O. Box 4500, 90014, Oulu, Finland

A B S T R A C T

Recent advances in technology are changing the way how everyday activities are performed. Technologies in the traffic domain provide diverse instruments of gathering and analysing data for more fuel-efficient, safe, and convenient travelling for both drivers and passengers. In this article, we propose a reference architecture for a context-aware driving assistant system. Moreover, we exemplify this architecture with a real prototype of a driving assistance system called Driving coach. This prototype collects, fuses and analyses diverse information, like digital map, weather, traffic situation, as well as vehicle information to provide drivers in-depth information regarding their previous trip along with personalised hints to improve their fuel-efficient driving in the future. The Driving coach system monitors its own performance, as well as driver feedback to correct itself to serve the driver more appropriately.

Keywords:
Green driving
Driving assistance
Ubiquitous computing
Context-awareness
Context reasoning

1. Introduction

The number of cars is increasing. Cars make our lives more convenient but have shortcomings as well. First, it is well known that car emissions have high effects on air pollution (European Environment Agency, 2011). Another worrying issue is that, among transport accidents, passenger car accidents have one of the leading positions (Eurostat, 2009). Different factors affect both of these characteristics, like driver experience, car model, traffic situation, and weather conditions. Often, a driver cannot understand or distinguish all these factors in order to optimise his driving. Our research aims to deliver to the driver information on his driving behaviour and instructions for safer and more environmentally-friendly driving.

Many researchers have investigated what affects fuel usage and emissions of a car, can the driver affect these parameters and how. Sivak and Schoettle (2012) classify the driver decisions which affect fuel consumption into strategic (vehicle selection and maintenance), tactical (route selection and vehicle road), and operational (driver behaviour). The authors discovered that vehicle selection has a dominating effect on fuel economy, but the remaining factors can contribute, in total, to about 45% reduction in fuel consumption. Ericsson (2001) has investigated the factors of driving behaviour which affect fuel consumption. From her findings, the most influential factors include stops during run, extreme acceleration, and late change from 2nd to 3rd gear. In turn, driving behaviour depends on many factors, among others are street and traffic environment (Brundell-Freij and Ericsson, 2005). For instance, the density of junctions controlled by traffic lights seems to have a high effect on driving behaviour and hence on fuel consumption and car emissions. Thus, route selection is an important factor having influence on fuel efficiency (Sivak and Schoettle, 2012; Brundell-Freij and Ericsson, 2005). Route selection refers to

* Corresponding author.
E-mail addresses: ekaterina.gilman@ee.oulu.fi (E. Gilman), anja.keskinarkaus@ee.oulu.fi (A. Keskinarkaus), satu.tamminen@ee.oulu.fi (S. Tamminen), susanna.pirttikangas@ee.oulu.fi (S. Pirttikangas), juha.roning@ee.oulu.fi (J. Röning), jukka.riekki@ee.oulu.fi (J. Riekki).

http://dx.doi.org/10.1016/j.trc.2015.02.007
0968-090X/© 2015 The Authors. Published by Elsevier Ltd.

planning a trip in such a way that the number of stops, high speeding, etc. are minimised. Ericsson et al. (2006) have calculated that a fuel-efficient route can save about 8% of fuel. Also, their study demonstrated that a fuel-efficient route is about the same as the shortest one. No significant fuel reduction effect was found for the fastest route option. Aggressive driving is another factor affecting fuel-consumption (Sivak and Schoettle, 2012; Brundell-Freij and Ericsson, 2005). Aggressive driving means certain actions increasing the risk of road accident, like excessive speeding and improper turning. In fact, aggressive driving is found to be one of the main causes of car accidents (The AAA Foundation for Traffic Safety, 2009). Based on the related work, we may conclude that it is possible to minimise fuel consumption by discovering the most relevant factors and informing drivers how to improve their driving behaviour with respect to these factors.

On the other hand, drivers vary a lot. They have different driving experience, preferences, and habits. Hence they require tailored solutions to explain what can be improved in their driving style and how (Gonder et al., 2011). Moreover, different external factors, like traffic fluency situation, road quality, and weather may affect performance of drivers. Hence, the overall situation should be assessed to promote more fuel-efficient driving. This information is referred as context, and systems able to capture the context and react on its changes are called context-aware (Dey, 2001). In this article, we argue that context-aware driving assistant systems provide more adequate feedback to drivers regarding fuel-efficient driving.

In this article, we propose a reference architecture for context-aware driving assistance systems. This architecture is aligned with a Meta-level control framework presented by Gilman and Riekki (2012). Their framework emphasises the necessity for self-introspective functionality for personalised and adaptive systems. This framework adds a controlling and monitoring layer to such systems. Moreover, it emphasises monitoring the overall interaction to gather feedback about how well the system supports its users in their tasks. For instance, with this kind of functionality, the system would notice that a driver constantly ignores certain advice and would perform actions to resolve such cases.

The proposed architecture is exemplified with a driving support system called Driving coach. This system teaches a driver to drive better. Better driving in this context means: (1) avoiding aggressive driving, (2) trip planning, and (3) driving in a fuel-efficient manner. The system is based on real-time information, obtained from on-board sensors and external services. The driver gets feedback about his driving after each trip: comments and recommendations what to do differently in order to drive better. The key characteristics of our system are:

1. Fusion of on-board information and real-time information from third party services.
2. Identification of personal driving factors affecting the fuel use in certain situations.
3. Adaptation of the system's decision-making with respect to a driver's progress and responses to recommendations.

Although many applications have been developed for driver assistance, our application is unique in combining these three characteristics.

The contribution of this article can be summarised as follows: First, we apply the Meta-level framework to create a reference architecture for context-aware driving assistance systems. Second, we propose Driving coach, which serves as the implementation use case for this reference architecture. Driving coach is a fully implemented and functional prototype which gathers diverse data from real trips (driving data, weather data, traffic situation data, and digital map data), analyses these data and presents feedback to the driver after the trip.

The rest of the article is structured as follows: First, we present a review of related research in Section 2. Section 3 presents our vision to equip driving assistance systems with Meta-level control. Section 4 discusses the big picture of Driving coach. Then, we provide details on data used in Section 5. Section 6 describes Driving coach core. Web interface of the system is presented in Section 7. Finally, we conclude the paper with Section 8.

2. Related work

Recently, keeping in mind that driving behaviour affects fuel consumption significantly, car manufacturers have started to invest in the development of on-board systems that provide drivers feedback about their driving (e.g., SmartGauge[1], ECO ASSIST[2]). These systems provide visual feedback about whether driving is fuel-efficient together with statistics about fuel consumption and possible savings. Another illustrative approach is ECO Pedal[3] from Nissan, which provides physical feedback with a pedal push-back control mechanism when a driver accelerates too heavily. More detailed analysis of trips is provided by Fiat eco:Drive[4] system. This solution gathers statistics about trips and provides explanatory feedback about how to drive more fuel-efficiently. On-board diagnostic scanners are becoming the most common tools for monitoring driving behaviour, as they can be bought separately and plugged into on-board diagnostic ports. Kiwi Drive Green[5] system serves as an example of such a tool. Kiwi device plugs into an on-board diagnostic port to obtain sensor information about the vehicle. The device analyses driving behaviour and delivers this information to the driver. The systems listed above are oriented for real-time driver awareness about fuel consumption and hence car emissions. Only eco:Drive system from Fiat provides explanatory feedback.

[1] http://en.wikipedia.org/wiki/Ford_Fusion_Hybrid#SmartGauge_for_eco_driving.
[2] http://world.honda.com/INSIGHT/eco/index.html.
[3] http://www.nissan-global.com/EN/NEWS/2008/_STORY/080804-02-e.html.
[4] http://www.fiat.com/ecodrive/.
[5] http://www.plxdevices.com/product_info.php?id=SCANKIWI.

In addition to special devices, modern smartphones can be used to monitor driving behaviour. Modern smartphones are equipped with sensors, like GPS receivers and accelerometers, that can provide data for driving support systems. Glass of water[6] application from Toyota is one good example of using the smartphone as a platform for driving assistant applications. This application motivates drivers to drive in a smoother manner by using the metaphor of a glass full of water. Obviously, the more aggressive driving is performed, the less water is left in the glass. Hence, the application persuades for less aggressive driving, which has a direct effect on fuel consumption and road safety. Another application of this kind is BlissTreck[7], motivating the driver to avoid stops and high speeds. GreenMeter[8] application provides detailed real-time driving statistics.

Research community has provided their own solutions to tackle fuel-efficient driving. For instance, a fuel-efficiency support tool by van der Voort et al. (2001) provides real-time user advices. This tool calculates minimal fuel consumption for manoeuvres carried out and if actual fuel consumption deviates from this optimum, the support tool presents advices to the driver. Another support solution is presented by Wu et al. (2011), their fuel economy optimisation system calculates optimal acceleration and deceleration values and advises the driver based on these values. Both systems utilise on-board sensor technologies to monitor driving behaviour. Vagg et al. (2013) consider light acceleration and early upshifts as the main sources to reduce fuel consumption and provide online feedback when drivers deviate from the optimal driving behaviour. Hellström et al. (2009) use external information to minimise fuel consumption. Their system uses road geometry information in order to optimise the velocity trajectory with respect to trip time and fuel economy. Mensing et al. (2013) suggest a vehicle model and optimisation method for optimal vehicle operation. Their system uses road constraints for optimisation. Guan and Frey (2012) suggest a driving assistant system that uses an adaptive power train model to generate fuel-efficient online guidelines for the driver. Abdullah et al. (2008) present a fuzzy-logic based controller for autonomous intelligent cruise control. Another cruise control system is presented by Khayyam et al. (2012). Authors present an adaptive neuro-fuzzy inference system that reduces energy consumption of a vehicle and improves its efficiency by controlling the vehicle speed based on the present speed and the predicted future slope information. Few works report dynamic adaptation. For instance, Syed et al. (2009) present a fuzzy-logic based learning algorithm, which estimates driver preferences. This algorithm is used for a driving advisory system for a hybrid vehicle. With the help of this algorithm, the system learns drivers' intentions by monitoring their driving style and behaviours and balances the competing requirements for fuel consumption and vehicle performance. Paz and Peeta (2009) also propose to utilise fuzzy-logic rules in a real-time route guidance system. Araujo et al. (2012) suggest a smartphone application coaching the driver to systematically reduce fuel consumption. The authors use vehicle data and data from smartphones to feed the system classifiers. Fuzzy logic rules implement the decision logic of the system.

As can be seen, research community looks ahead for providing real-time statistics on driving efficiency. Researchers are keen to educate drivers to drive better; hence, data analysis is performed and tips are generated for drivers. From car manufacturers' applications, only Fiat considers these aspects. However, external information is not used much yet, like weather information and road condition information. This information may have a significant effect on fuel-consumption (e.g., driving on icy road vs. dry road). Moreover, performing some manoeuvres can be dangerous in certain road conditions (e.g., on wet or slippery road). External information, like road profile, would allow analysing a driver's behaviour in more detail; hence, more accurate decisions could be made. Another case not addressed much in related work is personalisation. In our opinion, driving assistant systems should speak the same language as the driver. For example, when a system recommends a fuel-efficient route, it should take into account if the driver prefers the route with fewer traffic lights. The work presented in this article attempts to fill these gaps. We use external information for trip analysis. Moreover, Driving coach monitors driver progress and generates advices which work the best for the driver.

3. Meta-level framework applied to context-aware driving assistant system

3.1. Context and context-awareness

In order to be able to provide relevant feedback about a trip, a driving assistant system should capture and understand the situation a driver is in. Such a situation is described by diverse information, like weather, traffic situation, and driver-related information. This information that characterises the situation can be generalised as context. In turn, applications which use context to deliver relevant information or services to their users are called context-aware (Dey, 2001). Context-awareness is a desired feature for a driving assistant system, especially for real-time assistance, as context-awareness facilitates adaptation to the driving situation and provisioning of more relevant guidance. For instance, driving in low speed can be judged differently when the road surface is slippery and when there is a good grip. We summarise driving situation context with Fig. 1. Driving situation can be characterised with driver context, environmental context, and vehicle context. Some of this information is static (like vehicle characteristics), and some other is highly dynamic (environmental context). Moreover, driver context affects overall driving situation considerably, as drivers differ a lot in their experience, preferences, etc. Context information is retrieved with sensors and from third party services.

[6] http://www.aglassofwater.org/en.
[7] http://www.blisstrek.com/.
[8] http://hunter.pairsite.com/greenmeter/.

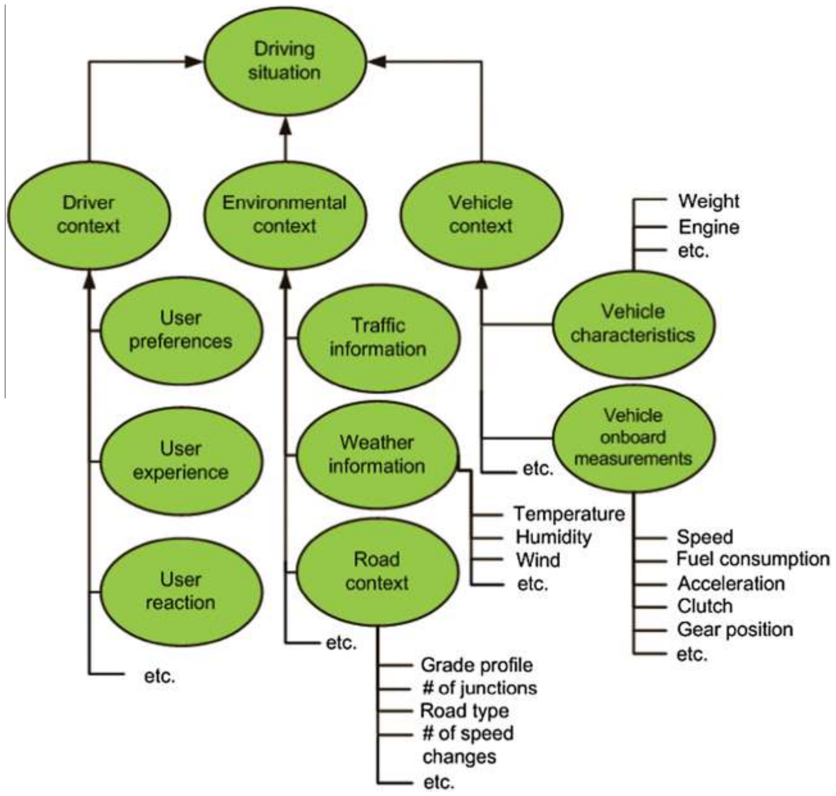


Fig. 1. Context characterising driving situation.

Different combinations of contexts can build more meaningful high-level contexts. Often, such high-level context is more useful for applications. In order to be able to provide the relevant support for a driver, a system must be able to recognise context, interpret and analyse it correctly, and provide corresponding feedback.

3.2. Reference architecture for Meta-level functionality of driving assistant systems

A context-aware driving assistant system, as any context-aware system, can be represented as an Action-Perception loop system (Fig. 2). Such a system senses the environment, vehicle and driver. Then, the system recognises context from sensor data and infers actions to perform as a response to the changes in this context. The system actions, in turn, can change the environment and driver behaviour. This can trigger new system actions and the cycle repeats. These main steps in the Action-Perception loop are included in Ground and Object levels (Fig. 2).

However, in complex and highly changing situations of the traffic domain, self-introspection capabilities can improve the quality of decision-making in context-aware driving assistant systems. These capabilities can be implemented by an additional layer (Meta-level in Fig. 2) which adds monitoring and control over the reasoning processes of the system (Cox and Raja, 2008, 2011). Meta level aims to improve the quality of decision-making by monitoring and evaluating how well decision-making progresses and if any changes are required to improve it (Cox and Raja, 2008). It should be noted, however, that in the traffic related domain, user satisfaction is an important criterion which can be used to evaluate the decision-making performed by the system. Hence, Meta level also monitors Ground level in order to optimise the decision-making. This monitoring plays also another useful role, as it allows giving more concrete explanations to the user about the decisions made by the system.

Gilman and Riekki (2012) propose a general framework to implement Meta-level functionality in context-aware applications. We adapted their general framework to design the reference architecture for context-aware driving assistance systems, see Fig. 3.

Ground level is presented with Actuators and Perceptors. Perceptors sense the environment (e.g., weather, traffic situation on the roads) and user (e.g., driving related measurements). Generally, Perceptors are represented by sensors, user interfaces, and services like weather service. Actuators change the environment (e.g., turn off the air conditioner in the car) and deliver information to users (e.g., show a message to the user). Actuators are represented by specific hardware and software, external services and user interfaces.

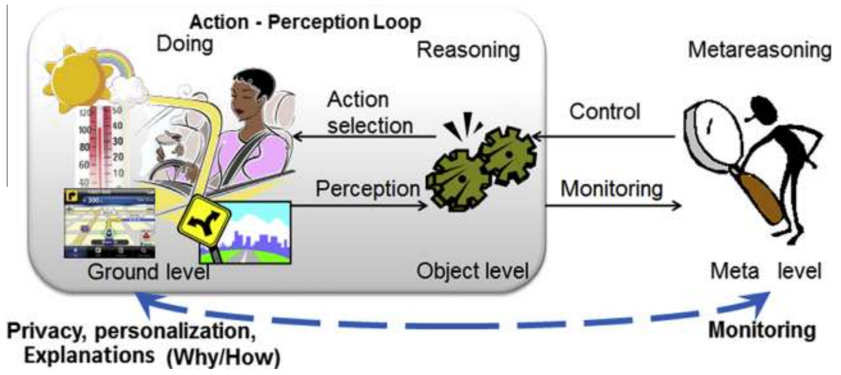


Fig. 2. Meta-level concept visualisation, modified from Cox and Raja (2008) and Gilman and Riekki (2012).

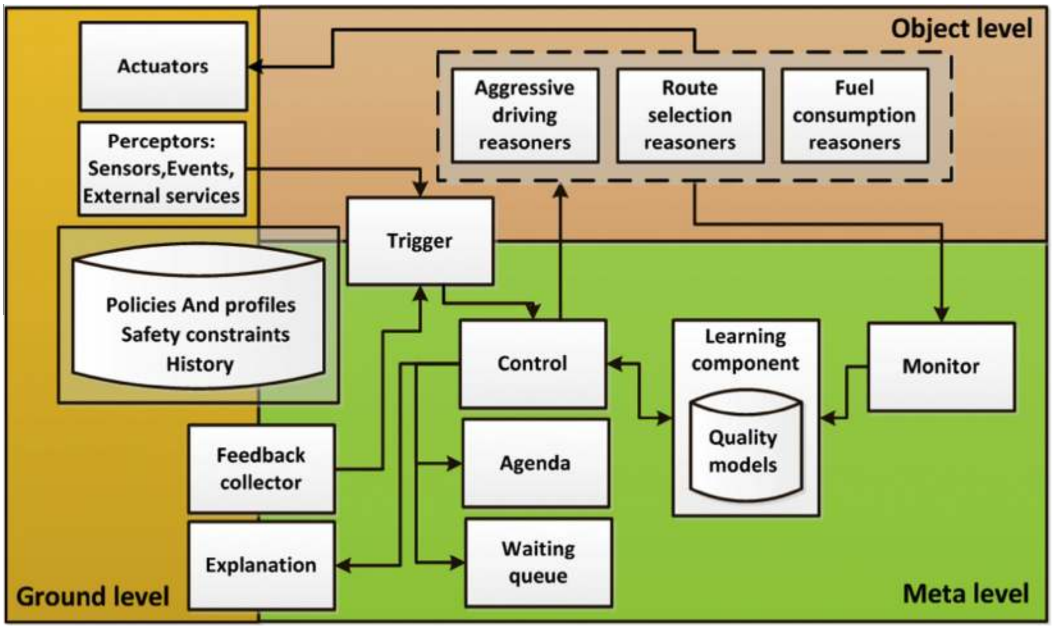


Fig. 3. Reference architecture for context-aware driving assistance systems, adapted from Gilman and Riekki (2012).

Object-level comprises the decision-making tasks of the driving assistance system. More specifically, it includes the set of reasoners required to evaluate the trip, as can be seen from Fig. 3. Object level informs its decisions to Actuators.

Meta-level tasks monitor and control the execution of the Object-level tasks. Trigger component receives the events happening in the environment and delivers them to Control component. The events can describe a new trip to evaluate, weather change, or traffic situation warning, for examples. Based on the incoming event, Control component estimates which event can be postponed and which should be handled immediately. To handle the event, Control component can alter the execution of an Object-level tasks, e.g., pause the reasoning process and tune parameters for reasoners. Control component consults Quality models when it makes control decisions. Quality models present best fits and best configurations of Object-level tasks and strategies for different contexts (Gilman and Riekki, 2012). Machine learning techniques of Learning component are used to build Quality models. Not urgent events wait for their processing in Agenda. The Waiting queue holds the events which cannot be handled at the moment, but can be processed when the context changes, e.g., a certain resource becomes available. Policies and Profiles database stores all the information regarding the driver, his performance and preferences. Safety constraints contain general terms of safety and driving rules in order to check that the advice given by the system is safe in the context. For instance, if the road is slippery, it can be unsafe to recommend to drive faster. History Storage gathers trip evaluation reports together with their contexts. We have placed these components so that each level can access them based on needs. For instance, History may serve as the data for the learning mechanism and may allow checking that the advice given by the system does not conflict with the previous one. Moreover, this facilitates analysis whether the driver accepts the advice given by the system. Feedback collector collects feedback from the user. Explanation component, in opposite, provides cause-effect explanations to the user why the system behaves in a certain way (Gilman and Riekki, 2012).

Monitoring and controlling system operation at a separate level (Meta-level) produces a clear design as each layer is responsible for certain types of tasks. This, in turn, improves modularization and simplifies system maintenance tasks. Moreover, such modularization allows simplification of Object-level components. Reusability is another advantage: the same Meta-level components can be used by several applications when interfaces are well defined. Still another benefit is customisation. Meta-level has a higher-level view on a particular situation; hence, tailored decisions can be made about objet-level tasks.

4. Driving coach

This article presents a driving assistant system, which serves as the example application of the architecture presented in the previous section. We call this system Driving coach. As our prototype does not yet support real-time operation, Driving coach supports drivers after the actual trip. Hence, not all the components of the reference architecture are required to implement the system.

The system architecture of the Driving coach is presented in Fig. 4. As can be seen, the Driving coach application is made of three blocks. These blocks are Driving coach back-end, RESTful client connector, and client applications. Driving coach back-end implements the functionality of the system. RESTful client connector provides interfaces for the client applications allowing them to concentrate on usability issues. Client applications deliver Driving coach information to users.

Driving coach back-end is the core of the system and consists of three main components: Data suppliers, Storage, and System core. Data suppliers feed the system with heterogeneous context information, such as weather and traffic situation during the trip. Storage is the database gathering all information in one place. System core component performs analysis in order to provide feedback to users.

Driving coach system implements the reference architecture as follows: Ground level is presented with Data suppliers (Perceptors, Fig. 2) and client applications (Actuators, Fig. 2). Storage system accumulates Policies and Profiles, Safety constraints, and History (Fig. 2). As can be seen from Fig. 4, main functional blocks of System core are Trip evaluation, Comment generation, and Model learning. These blocks cover both Object and Meta levels and are discussed in more detail in Section 6. The current version of Driving coach does not implement Agenda and Waiting queue elements. This is due to the fact that the only event triggering the system decision making is a new trip made by a driver. As this trip immediately gets analysed to provide feedback, there is no need for Agenda. Moreover, there is no need for Waiting queue, as all required context is directly available. Driving coach is implemented in Java, with SWI Prolog and R tool for statistical computing.

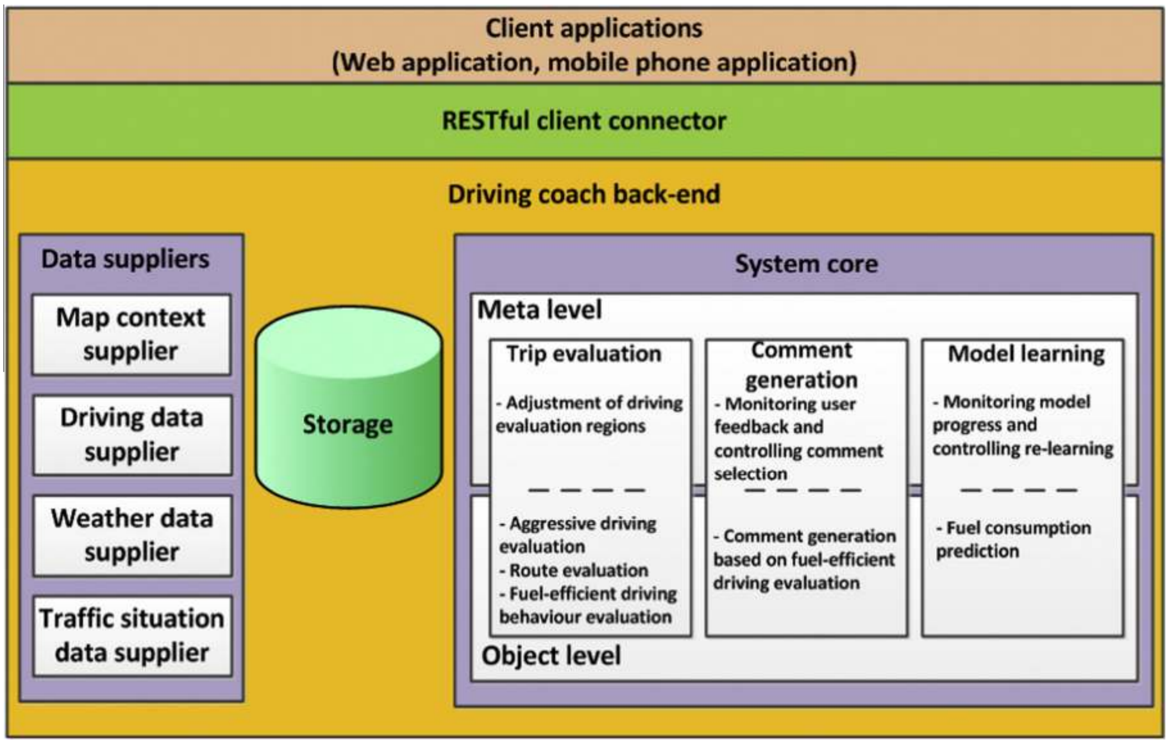

Client applications
(Web application, mobile phone application)
RESTful client connector
Driving coach back-end
Data suppliers
Map context supplier
Driving data supplier
Weather data supplier
Traffic situation data supplier
Storage
System core
Meta level
Trip evaluation
- Adjustment of driving evaluation regions
- Aggressive driving evaluation
- Route evaluation
- Fuel-efficient driving behaviour evaluation
Comment generation
- Monitoring user feedback and controlling comment selection
- Comment generation based on fuel-efficient driving evaluation
Model learning
- Monitoring model progress and controlling re-learning
- Fuel consumption prediction
Object level

Fig. 4. System architecture.

5. Driving coach: Data suppliers and Storage

Data suppliers are based on a large pilot that we have implemented to collect data from the region covering the city of Oulu in Finland. This pilot contributes to Data to Intelligence[9] project.

Map context of the driven route can be very useful to reveal and explain certain driving patterns. Moreover, knowing the road characteristics is important for personalised route recommendations. In order to retrieve map information about roads, we use Digiroad[10], which is a database of Finnish road network maintained by National Land Survey of Finland, the Finnish Transport Agency and individual municipalities. The Digiroad database contains up-to-date information about the road network centreline and road type. The database contains also road attribute information that can be used to develop services like route planning and navigation. Examples of attribute data include information about bridges, bus stops, speed limits, driving directions and restrictions, road quality, and traffic lights.

The road network database used in our case is the latest update of database on Northern Ostrobothnia. We have restricted the database to the local region, in order to speed up queries and for other practical challenges related to a large and heavy database. Fig. 5 demonstrates the road network used for analysis.

We have created a road network graph from the data of Digiroad database where each edge represents a road segment between two intersections. Moreover, we process the available attribute data in a way that it corresponds correctly to the created road network graph. The resulting road network is stored in Storage.

Driving Data are used as follows: First, these data allow getting a general picture of driving style of a driver, as well as observing aggressive driving behaviour occurrences, like very high accelerations or decelerations and high speeds. Second, we retrieve map related features, available from Map data supplier, of the route driven. This allows us to layout the driving style to the map to reveal interesting driving patterns and perhaps provide more information on the cause of such patterns. Also, this allows us to evaluate the route selection strategy used by the driver. Moreover, presenting the data on a common coordinate system allows comparing drivers and calculating features for a set of drivers.

Driving data are retrieved by a Driveco[11] device which collects data from the OBDII diagnostics connector and transfers these data via GPRS/3G connection to a dedicated service. This information is structured into trips and route points. A trip is defined as a run between two stops where the engine is off. Each trip contains the vector of properties measured for the whole trip, such as the whole distance travelled. In addition, each trip is presented with a set of measured points, so called route points. Route points contain the vector of measured properties for one location, such as speed, fuel spent, time and distance driven together with geographic coordinates. More details are given in Table 1.

There is no specific sampling rate for the route points, but a new route point is generated when some significant change in driving behaviour is registered, such as a turn. Fig. 6 presents a trip example, together with registered route points.

Raw driving data retrieved from the Driveco device were filtered and map matched. Data filtering ensures that data for analyses is of sufficient quality, as sensor data are error prone. We make sure that trips fall into local region (Fig. 5) and contain not less than five route points. Also, when necessary, we rearrange route points of the trip to make sure that time, fuel and distance increase monotonically.

In order to retrieve map related information about the route, it needs to be laid out on the road network of the map (described in the previous section). The procedure of placing the driven route to the road network is called map-matching. An incremental map-matching algorithm (Brakatsoulas et al., 2005) was implemented for Driving coach. We have incorporated into this basic incremental algorithm details about road directions and bi-directional Dijkstra Shortest Path implementation of pgRouting package to handle the cases where route points are very far from each other. After this procedure, route points have coordinates aligned to the road network. As a result, raw data from the Driveco device are filtered and map-matched, and delivered to Storage in the format of Table 1 with aligned route points and updated coordinates.

Weather data can be used to identify certain driving patterns observed in specific weather conditions. Moreover, driving behaviour can be judged differently based on road and weather conditions. Weather information is provided by Digitraffic[12] service, which offers real time and historical information about the traffic on the Finnish main roads. Road weather information is collected with road weather stations and provided via a Web service interface. We have selected one road weather station, close to the city centre, as a data supplier, see yellow dot in Fig. 5. We query road weather update information every half an hour. Weather properties used in the analysis include date and time, air, surface and dew point temperature, relative humidity, surface condition describing how good is the grip with the surface of the road, precipitation intensity and type, visibility, friction coefficient, and amount of snow, water and ice on the road. This information is delivered to Storage.

Traffic situation data can be used to evaluate the route selection decision made by a driver, as well as accumulate the statistics about a certain road, e.g., how fluent is the traffic of a certain road based on the time of day. Traffic situation data we retrieve from Oulunliikenne[13] service providing information about traffic situation for major roads in the local region.

[9] Data to Intelligence project, URL http://www.datatointelligence.fi/.
[10] Finnish Transport Agency, "Digiroad-the Finnish road network in digital format," URL http://www.digiroad.fi/dokumentit/en_GB/documents/.
[11] http://eco.driveco.fi/www/.
[12] http://www.infotripla.fi/digitraffic/doku.php?id=start_en.
[13] http://www.oulunliikenne.fi/#/autoilu.

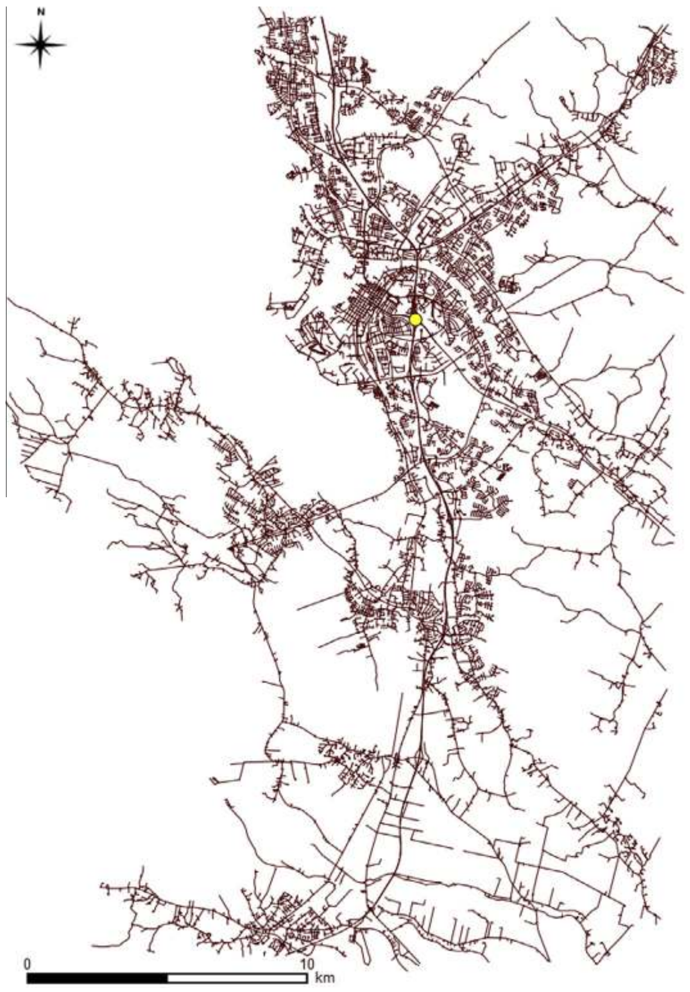


Fig. 5. Road network used for Driving coach.[14]

Table 1
Subset of measured properties for Driving behaviour.

Data structure	Properties used for analysis
Trip	Trip id, start and end time of the trip (Unix timestamp), start and end route point, total time (s), total distance (m), total fuel (ml), total CO_2 (g)
Route point	Point id, trip id, latitude, longitude, start time of the trip (Unix timestamp), speed (km/h), average speed (km/h), rpm, average rpm, distance from start of the trip (m), fuel from start of the trip (ml), time from start of the trip (ms)

Oulunliikenne categorises traffic flow to three levels: smooth (green line), constrained (yellow[14] line) and bad (red line). These categories are visualised on the map, see Fig. 7(a). We retrieve these data every half an hour.

To use data from Oulunliikenne service in our system, we process them as follows: We match the retrieved coordinates from Oulunliikenne to corresponding road segments of the road network. As a result, traffic situation information for road segments of the road network is delivered to Storage in the following format: {road segment, direction, time, traffic situation indicator}. Fig. 7(b) indicates all the roads for which we have historical traffic situation information. Fig. 7(c) and (d) show an example of information retrieved for a specific road (marked as red, the arrow shows the direction). The amount of smooth (green), constrained (yellow), and bad (red) traffic during one hour is visualised as one pie chart.

[14] For interpretation of color in Figs. 5,6,7,12, the reader is referred to the Web version of this article.

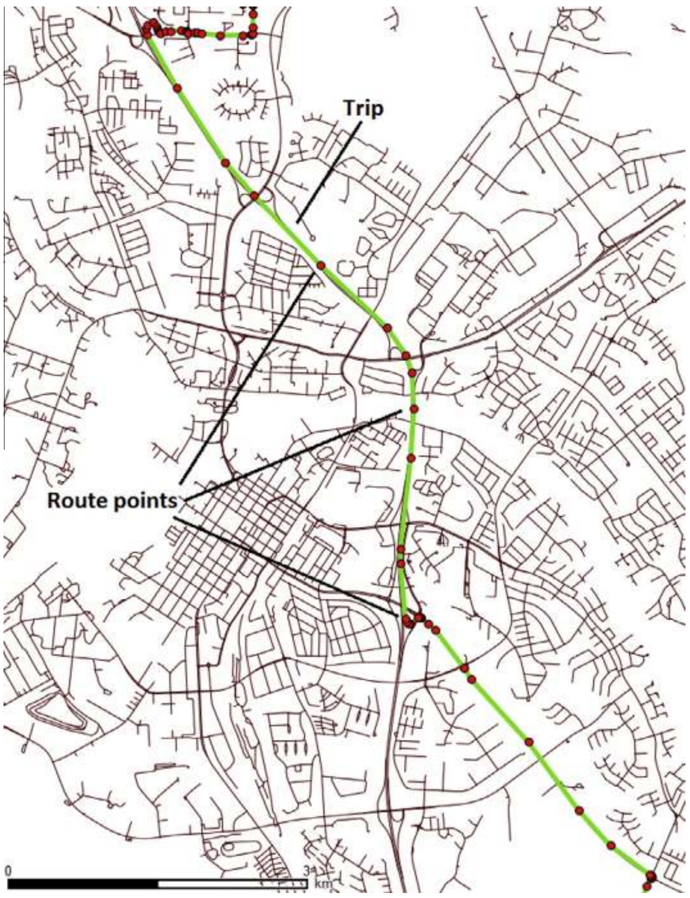


Fig. 6. Example of a trip (green) and its route points (red).[14]

Storage. PostgreSQL DBMS is selected to aggregate the data coming from Data suppliers. To visualise results, we use Quantum GIS version 1.8.0 "Lisboa". This toolset was chosen because it is freely available, provides all the desired functionality (e.g., PostGIS 2.0 extension to operate with spatial information), and allows easy integration with self-written programming code, Steiniger and Hunter (2013). It was decided to accumulate all the information in one Storage due to complexity of analysis, as well as inability to retrieve the historical information from context provisioning services. The raw and processed data from each data supplier are stored under a separate schema in the database; this provides modularity and maintenance advantages. Also, because of heavy data pre-processing of driver trips, it was decided to store driving-related information for each user in a separate schema. In our case, this solution provides benefits for parallel access to the data tables for different users, which could be problematic otherwise because of heavy data pre-processing and table blockings. Storage notifies the System core when a new trip has been processed and can be used for analysis.

6. Driving coach: System core

System core of Driving coach implements the application logic. The incoming trips get processed immediately as the System core gets notified whenever a new trip was made by a driver. The core has access to all the processed information from Data suppliers in Storage. At the same time, the core monitors the quality of solutions proposed and adapts system behaviour if required. Hence, the System core of Driving coach implements both Object-level and Meta-level functionality. As can be seen from Fig. 4, its main functional blocks are Trip evaluation, Comment generation, and Model learning. Below, we will describe how these tasks are supported on both Object and Meta levels.

6.1. Object-level functionality

Object-level functionality implements trip analysis regarding driving behaviour, aggressive driving, and route selection strategy. We fuse all the data provided by Data suppliers to grasp the big picture of the trip. In other words, we analyse

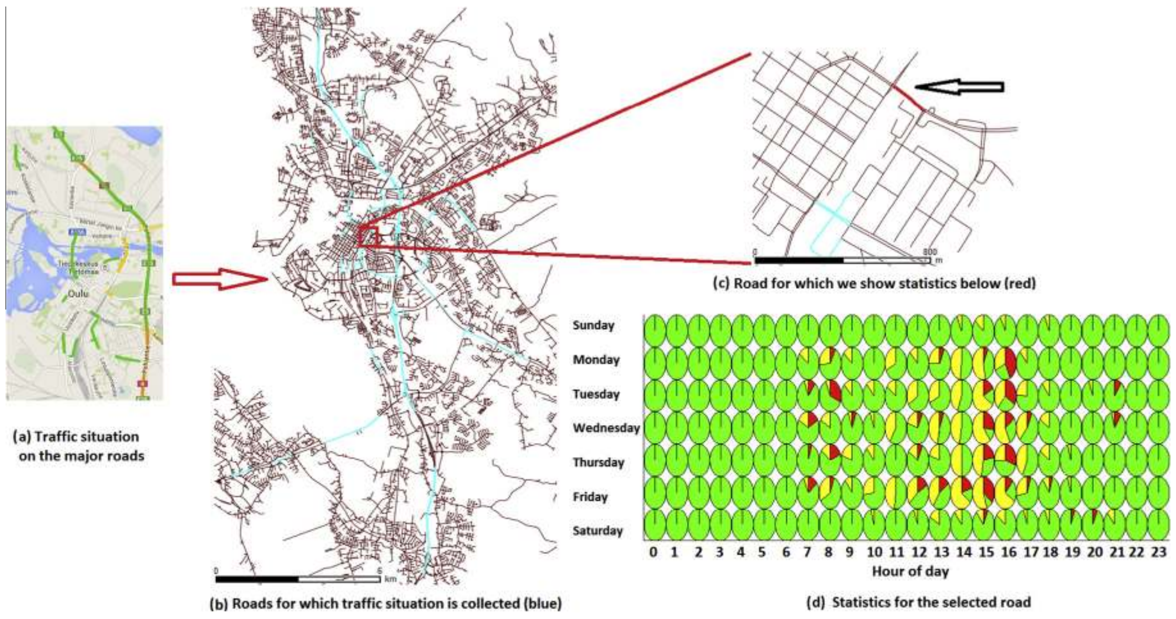

(a) Traffic situation on the major roads
(b) Roads for which traffic situation is collected (blue)
(c) Road for which we show statistics below (red)
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of day
(d) Statistics for the selected road

Fig. 7. Traffic situation information from Oulunliikenne service.[14]

the actual driving behaviour, retrieve map characteristics and traffic situation for the driven route, as well as what kind of weather was during the trip.

Aggressive driving evaluation. Driving coach uses threshold values to detect aggressive driving occurrences based on data from Driving data supplier. Three cases indicate aggressive driving: high acceleration (more than 2.5 m/s^2), high deceleration (less than -2.5 m/s^2), as well as high speed (more than 110 km/h). Moreover, we remember the coordinates for such cases. This can be useful for future research to find out connections between aggressive driving occurrences and other context information. Current implementation has limited capabilities in grasping aggressive driving behaviour because of sparse measurements; hence, precise coordinates of aggressive driving occurrences could be missed. We will use an improved version of Driveco device with increased sampling rate in our future research. Another possible solution to overcome this issue is to utilise mobile phones with embedded sensors and additional hardware and software to detect more cases which can be considered as aggressive driving (Johnson and Trivedi, 2011; Eren et al., 2012).

Route evaluation. To evaluate the route, we retrieve its map characteristics (road network from Map context supplier), as well as traffic situation (data from Traffic situation data supplier). Retrieved map characteristics of the route are shown in Table 2. Traffic situation for the driven route is presented as a vector with three values, telling the proportion of the length of the route driven with smooth, restricted and bad traffic. The route with less traffic lights, pedestrian crossings and smooth traffic consumes less fuel (if other route characteristics are the same) (Ericsson et al., 2006). Driving coach provides information about a trip after it has been accomplished; hence, real-time route guidance is not supported. Current implementation of Driving coach presents features of the driven route to the driver, rather than evaluates it.

Fuel-efficient driving behaviour evaluation. Driving coach system uses selected factors from Ericsson (2001) to evaluate driving behaviour in terms of fuel efficiency, see Table 3 for factors used. These factors are calculated from Driving data supplier data for the whole trip, as soon as it has been stored in Storage. The system evaluates each trip with respect to the trips driven by the driver during the last two weeks. This window allows positioning the current trip to the current driver's progress and weather conditions.

Driving behaviour for the registered trip is evaluated for each factor presented in Table 3. The evaluation is performed by calculating a performance indicator value for each factor. The four values of performance indicator are: "quite bad", "fine", "good", "very good". Together, these values form the performance vector of the trip in question. To determine performance indicator values, we use the following procedure: We collect information for all the trips driven during the last two weeks and build distributions for the factors. Driving factors usually have a distribution of a bell-shaped form, meaning that there are fewer trips at the beginning and the end of the factor value spectrum. Based on these distributions, we determine the four regions for each factor which would indicate factor values. The first region is constituted with the factor range from the beginning of the spectrum till the first quartile (region 1 of Fig. 8). The second region is formed with the factor range

Table 2
Route map properties retrieved from the route driven.

Property	Comment
speedNpercentage where $N \in \{20, 40, 50, 60, 80, 100, 120\}$	% of the trip distance driven on the road with speed limit N km/h
traffic_lights	Number of traffic lights per meter in the driven route
num_crossings	Number of crossings per meter in the driven route
num_crossings_ped	Number of pedestrian or bicycle crossings per meter in the driven route
tettypeN where $N \in \{1, 2, 3, 4, 8, 14, 17, 18\}$	Type of the road (describing physical or traffic-type attributes) telling % of the trip driven on *N*, where $N \in$ {motorway, part of a multiple carriageway which is not a motorway, part of a single carriageway, roundabout, slip road, cycle path, semi-motorway, rest area}
vtypeN where $N \in \{1, 2, 3, 4\}$	% of trip driven on road type *N*, where $N \in$ {road, street, private, pedestrian or cycle}
ftypeN where $N \in \{1, 2, 3, 4, 5, 6, 10\}$	Functional class of the road (describing the service level of the road to the traffic) telling % of trip driven on *N*, where $N \in$ {regional main (class I) street, regional main (class II) street, local main street, connecting road, feeder street/class I private road, class II private road, cycle or pedestrian path}

Table 3
Factors used for fuel-efficient driving behaviour evaluation (Ericsson, 2001).

Factor	Comment
stop factor	% of time with the speed < 2 km/h
speed0_15	% of time with the speed < 15 km/h
speed15_30	% of time with the speed 15–30 km/h
speed30_50	% of time with the speed 30–50 km/h
speed50_70	% of time with the speed 50–70 km/h
speed70_90	% of time with the speed 70–90 km/h
speed90_110	% of time with the speed 90–110 km/h
speed110	% of time with the speed over 110 km/h
engine_speed1500	% of time with the engine speed < 1500
engine_speed1500_2500	% of time with the engine speed 1500–2500
engine_speed2500_3500	% of time with the engine speed 2500–3500
engine_speed3500	% of time with the engine speed > 3500
speed_oscillation	Speed oscillations (local max and min) per driven meter
accel_moderate	% of time when multiplication of speed and acceleration is 3–10 m^2/s^3
accel_high	% when the acceleration > 2.5 m/s^2
rpa	Relative positive acceleration
decel_av	Average deceleration

from the first quartile till the factor value having maximum in density distribution (region 2 of Fig. 8). The third region is formed with the factor range from the maximum density distribution till the third quartile (region 3 of Fig. 8). Finally, the rest of the data constitute the fourth region (region 4 of Fig. 8). Fig. 8 gives an example for splitting the factor distribution into four regions with the procedure defined above. These regions are used to grade the factors of the current trip with performance indicator values. In fact, the same process can be used for fuzzification of the driving factors related variables into the fuzzy system used for trip evaluation.

The factors of the current trip are labelled with the numbers of the regions in which they fall in two week trip set. Then, we assign performance indicator values based on the region numbers. Depending on the factor and its effect on fuel consumption (Ericsson, 2001), we want to either reduce the region 1 or region 4 occurrences. That is, the performance indicator “quite bad” corresponds to either region 1 or region 4. For instance, for a stop_factor, region 4 presents cases when trips were driven with a high percent of time with very low speed, which is bad for fuel consumption. Hence, if the stop_factor of the current trip falls into region 4, it will be evaluated with performance indicator “quite bad”. However, if the factor for speed50_70 falls into region 4, we grade it with the performance indicator “very good” and want to reduce the region 1 occurrences. Hence, fuel-efficiency trip evaluation produces a vector with values representing the factors and their performance indicators.

Comment generation based on fuel-efficient driving evaluation. Comments regarding the driving behaviour are formed with a rule-based system. This system assigns comments to factors based on their performance indicators, as well as weather information. A rule-based system was selected as it provides a formal way to encode the expert knowledge, as well enough expressive power for end users and maintenance advantages for developers (Bikakis and Antoniou, 2010; Gilman et al., 2010). Driving coach uses SWI-Prolog system for these purposes. We have generated several comments regarding the same situation, so that the same comment would not be repeated too often. Content-wise comments form three groups: general, environment-related, and money savings related. This division was made in order to make some analysis in the future about which kinds of comments persuade drivers the most. Hence, the system can adapt its advice generation strategy to prefer giving advice from a certain group. Table 4 gives some examples of the comments.

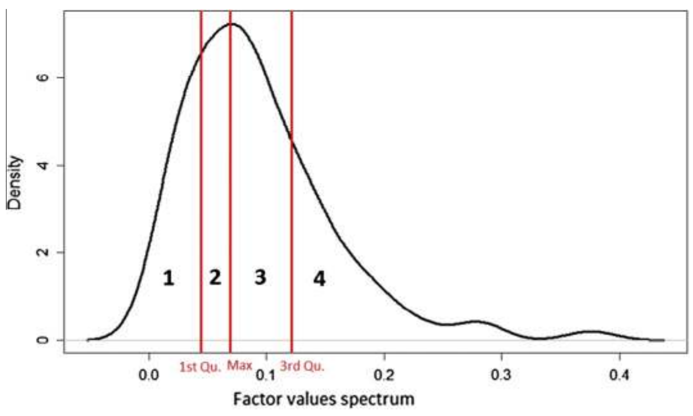


Fig. 8. Example of dividing a factor into four regions, region numbers are shown.

Table 4
Examples of comments.

Comment
"You could save more fuel if you would stop less, even when driving in bad weather conditions"
"Yes, it was bad weather for driving, but making less stops is beneficial for the environment"
"Perhaps, you should consider route planning, as having many stops consumes fuel and hence your money"
"It was a comfortable driving weather, but perhaps, you should think about environment more and reduce the amount of stops"

Fuel consumption prediction. In order to estimate the possible fuel savings, it is necessary to create models to evaluate the response variable based on given parameters. Different models were proposed by researchers for fuel prediction, e.g., Mensing et al. (2013) and Guan and Frey (2012) suggest use of ready-made formulas, including vehicle characteristics, Parlak et al., 2006 use a neural network to predict fuel consumption from vehicle characteristics. Zhou et al. (2013) suggest a fuel consumption model based on driving patterns; however, it does not demonstrate such a high performance in comparison to vehicle-based models. Current implementation of the Driving coach solves a regression task and contains three types of models, namely, linear regression model (Type I), decision tree model (Type II) and neural network model (Type III). These models are created in R with the corresponding packages (Kuhn, 2008). For all the models, fuel spent for each meter of the trip is the response variable. There are two models created for each type. The first model (Model A) contains driving related explanatory variables from Table 3 and weather related explanatory variables from Section 5. The second model (Model B) uses explanatory variables of Model A, as well as route map related explanatory variables from Table 2. This division to Model A and Model B is done purposefully for the cases where driven route was not map matched. This is related to uncertainty of the data that is very common in real world situations. So, by having two kinds of models, Driving coach system handles such uncertainty cases. Based on the model selected, we can alter some values of the current trip, calculate fuel consumption with the altered values, and provide feedback to the driver about possible savings of fuel if he behaves differently.

To avoid cold start, we provide the drivers with the models built from observed data for one driver from February till November 2013. After data cleaning and map matching, we resulted with 624 trips which were driven with distance between 1 and 40 km. We have established these limits as driving very small distances can be very difficult to predict with the measurement methodology we have. Moreover, driving more than 40 km in one trip is unlikely in selected city area. We have calculated route map properties of Table 2, as well as fuel-efficient driving behaviour factors from Table 3. We are interested in how much fuel was spent for each driven meter of the trip; hence, the variable we are interested is fuel (ml/m). Data for initial model were randomly divided into two sets: training set (80% of all data) and test set (20% of all data). The training set was used to train the models, when the test set was used to evaluate the models. We have manually constructed initial models based on distributions of explanatory variables and selected the best ones based on models performance. Initial models are presented in Tables A1–A3 of Appendix.

6.2. Meta-level functionality

Meta-level functionality in our system, as can be seen from Fig. 4, covers adjustments of the driving behaviour evaluation regions, monitoring user feedback and controlling comment selection, monitoring model progress and controlling re-learning.

Adjustment of driving behaviour evaluation regions. Bad driving weather may force drivers to drive worse than they would do otherwise. For instance, a very slippery road or heavy snowfall may increase the value for low speed factor due to safety reasons. At the same time, having high speeds in bad driving weather conditions may be dangerous and considered as bad driving although the same speeds would be acceptable in good weather. Also, one may want to evaluate an experienced driver more strictly than a novice one. To take these issues into account, we use a fuzzy logic rule-based system. This fuzzy system shifts the regions of the factors in order to release or strengthen the evaluation of the factor for the registered trip, see Fig. 9 for an example. For instance, if based on the original adjustment, a factor value falls into region 3, it may fall into region 2 or region 4 after weather-based adjustment.

Fig. 10 demonstrates the fuzzy weather variables used by the system. We have constructed them with the information given by Finnish Meteorological Institute. Table 5 gives an example of rules used to adjust the borders of the regions. These rules are used for low speed factors. In the case of low speed, the system relaxes factor judgement regions. This means that when bad driving weather is encountered, the system shifts the factor region borders to the right and relaxes the evaluation. So, a high amount of low speed during bad weather is judged less strongly as having the same amount of low speed during good weather. We have several different sets of rules which are applied for low speed factors, high speed factors, and factors related to acceleration and deceleration. The same approach can be used to adapt to user progress, for example. The fuzzy inference system has been implemented with Sets package of the R statistics toolkit (Meyer and Hornik, 2009).

Monitoring user feedback and controlling comment selection. As presented above, the comments are divided into three groups: general, environment-related, and money savings related. The system monitors which advices persuade a driver to drive in a more fuel-efficient manner. Driver responses to the comments are analysed at two time scales: immediately and weekly. Immediate response tells whether a driver followed comments already in his next trip. Week response tells if the driver's skills improved so that the improvement is visible in the commented factor. These responses are represented numerically with a positive number for improved skill and a negative number for not improved skill. Generally, these responses may reveal the usefulness of the advice (whether advice led to an improvement of driving behaviour) generated for the driver. Hence, the system favours the comments which are considered useful by the driver. The Driving coach presents other comments as well, but less frequently. SWI-Prolog rules are used to control comment selection. Table 6 demonstrates SWI-Prolog rules, which replace the given comments with more appropriate ones.

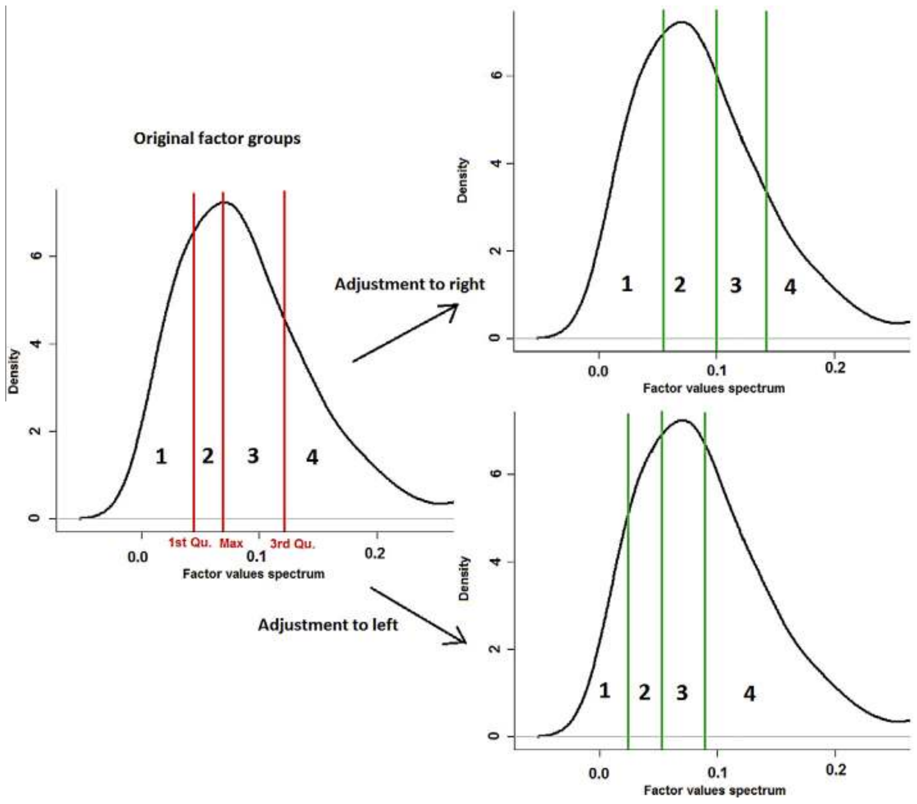


Fig. 9. Factor group adjustment example.

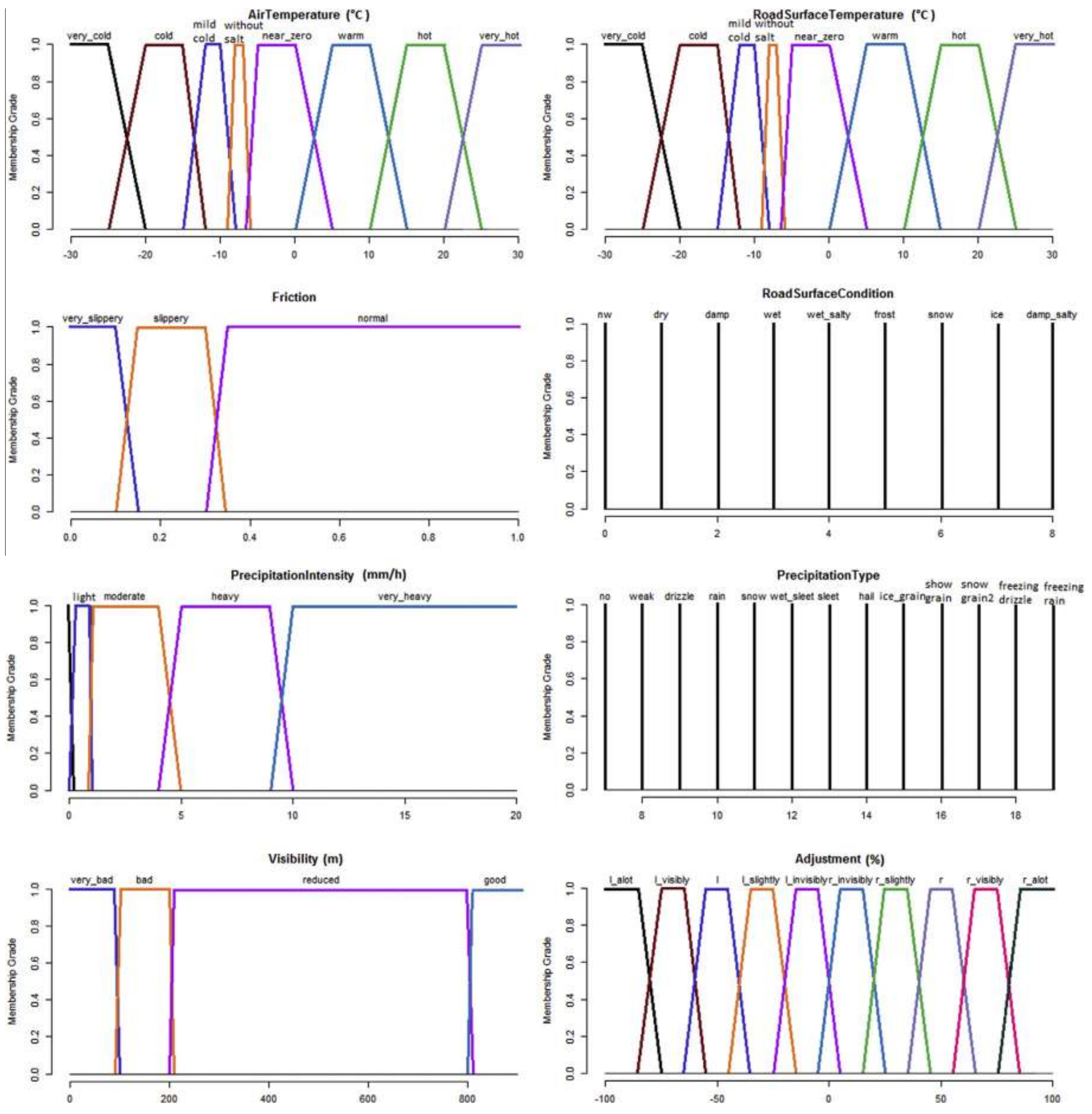


Fig. 10. Fuzzy variables of weather used for factor region adjustments.

Monitoring model progress and controlling re-learning. An appropriate fuel-consumption model is required in order to give adequate economy feedback to the driver (Fuel consumption prediction paragraph of Section 6.1). Hence, the Driving coach system monitors the models learnt for the driver, selects the most appropriate one for the given context, and commands the re-learning process if the model does not perform well enough.

The Driving coach system progresses together with the driver. This means that the system evaluates the driving based on recent driving style, always pushing the driver to drive better. Learnt models can become irrelevant based on two factors: significant change of driving behaviour, and significant change of driving conditions. Driving behaviour changes when a driver's skills change or the driver changes his preferences. In such a situation, the learnt model does not correspond well enough to the situation anymore and a re-learning process has to be initiated. Another issue is the change of the driving conditions, like weather. In countries with snowy winters, a weather change can make the current model irrelevant. For instance, friction coefficient may not be very useful for the model during the summer; however, it is considered to be very important during winter trips. Therefore, monitoring the performance of models is important.

The issues described above are relevant for many systems and are related to adaptive learning and concept drift adaptation (Tsymbal, 2004; Gama et al., 2014). That is, systems should detect and adapt to evolving data over time in order to

Table 5
Factor group borders adjustments.

R code and Interpretation

```
low_speed_rules<-set(
  #slipperiness
fuzzy_rule(friction %is% very_slippery, adjust %is% right_alot),
fuzzy_rule(friction %is% slippery, adjust %is% right_visibly),
fuzzy_rule(roadSurfaceCondition %is% ice, adjust %is% right_visibly),
fuzzy_rule(roadSurfaceTemperature %is% near_zero_without_salt, adjust %is% right_alot),
fuzzy_rule(roadSurfaceTemperature %is% near_zero_without_salt && !(precipitationIntensity %is% no), adjust %is% right),

  #heavy precipitation
fuzzy_rule(precipitationType %is% freezing_rain, adjust %is% right_alot),
fuzzy_rule(precipitationType %is% snow||precipitationType %is% hail, adjust %is% right),
fuzzy_rule(precipitationType %is% ice_grain||precipitationType %is% snow_grain1||precipitationType %is% snow_grain2,
   adjust %is% right_visibly),
fuzzy_rule(precipitationIntensity %is% heavy_rain||precipitationIntensity %is% very_heavy_rain, adjust %is%
   right_visibly),

  #bad visibility
fuzzy_rule(visibility %is% very_bad, adjust %is% right_alot),
fuzzy_rule(visibility %is% bad, adjust %is% right_visibly),
fuzzy_rule(visibility %is% reduced, adjust %is% right))
```

```
  #slipperiness
If the road is very slippery then adjust borders of the regions to the right a lot,
If the road is slippery then adjust borders of the regions to the right visibly,
If the road surface is icy then adjust borders of the regions to the right visibly,
If the road temperature is near zero without salt then adjust borders of the regions to the right a lot,
If the road is near zero without salt and there is some rain then adjust borders of the regions to the right,

  #heavy precipitation
If there is freezing rain then adjust the borders of the regions to the right a lot,
If there is snow falling or hail then adjust the borders of the regions to the right,
If there is ice grain or snow grain then adjust borders of the regions to the right visibly,
If there is very heavy or heavy rain then adjust borders of the regions to the right visibly,

  #bad visibility
If visibility is very bad then adjust the borders of the regions to the right a lot,
If visibility is bad then adjust the borders of the regions to the right visibly,
If visibility is reduced then adjust the borders of the regions to the right
```

Table 6
SWI-Prolog rules telling whether to change the feedbacks given to the user to other ones, with better immediate and week response.

SWI-Prolog code

```
%Collect all the feedbacks generated for the given User
checkFeedbacksGiven(User):- findall(N,feedback(User,N),R),
    forall(member(X,R), check_feedback(User,X)).
%Check each feedback
check_feedback(User,Id):-
%we generate random number to keep the probability for all the advices to be given
N is random(3),
%however, system favours useful advices, as the probability to get the random number < 2
%from {0, 1, 2} is higher
((N < 2) -> (feedback(User,Id),
%getting the immediate and week responses for feedback
  (%if feedbacks are given, retrieve them
    score(User,Id,ImmediateResponse,WeekResponse) ->(
      advice(Id,Factor,Level,Weather,_,_),
%if responses are negative, try to replace the advice with the one having better response
((ImmediateResponse < 0, WeekResponse < 0) -> (retractall(feedback(User,Id)),
    findall(O,advice(O,Factor,Level,Weather,_,_),R),
    find_replacement(User,R,_,NewId),
    assertz(feedback(User,NewId)),!);
    %if one of responses is positive, keep current comment
  true));
%if feedbacks are not given, keep current comment
    \+(score(User,Id,ImmediateResponse,WeekResponse)) -> true));
%if randomly generated number = 2 (less probable), keep current comment
   (N > 1) -> true
).
```

provide required performance. One of the challenges for such systems is to distinguish a real change in the data from a sudden deviation or anomaly, known as outlier. Several approaches have been suggested to determine when the old model should be reconsidered (Koskimaki et al., 2008; Gama et al., 2014).

Driving coach separates the treatment of weather change and driving behaviour change. This is done with the assumption that driving behaviour related models can differ for different weather and we wanted to be clear what causes the model degradation: weather or driving style change. However, more formal analysis should be conducted towards this design decision, and this is left for the future work.

Weather context and behavioural change serve as examples of model selection and model re-learning in Driving coach. Weather context is used to select the model for economy feedback generation. Hence, the system finds the models that functioned well for other trips having the same weather as the trip in question. For this task, we need to define context similarity. This means to define the measure which numerically represents how close one context is to another. Several metrics have been suggested (Chen, 2005; Boriah et al., 2008). Driving coach uses cosine similarity for continuous contexts (like temperature) and specifically defined distance matrices for categorical contexts (like precipitation type). The model giving the best result in terms of root mean square error gets selected and is used for generating economy feedback for the user.

Behavioural change is detected by using a threshold error value. When the error predicted with a model exceeds a threshold, we raise a flag indicating that probably the situation has changed and hence the model does not operate well. Several subsequent trips are monitored to detect the tendency (that the error is still larger than the threshold). This is required in order to evaluate whether the flag was raised for an outlier. When enough data have been gathered, the model can be retrained. Table 7 demonstrates SWI Prolog rules used for behavioural change detection. Schematically, the overall process is presented with Fig. 11.

Table 7
SWI-Prolog rules telling if the fuel prediction model should be retrained.

SWI-Prolog code

```
%Check if the model should be relearnt
potentiallyRelearn(User,ModelName,TripStartTime,Error):-
  %Check if this model already performed worse than allowed (that the model has a flag)
  findall(N,model_flag(User,ModelName,N,_),R),
  length(R,ModelFlags),
  (
      (ModelFlags > 0) -> ( %the model has flag already
      %getting the time of previous bad performance
      previous_occurence(User,ModelName,PreviousTime),
      %finding how many times the model can perform worse than allowed
      minimum_occurrences_allowed(User,Occurrences),
      %finding the time interval considered for change in behaviour
      days_interval(User,Days),
      %calculating the time difference between current and previous model with bad performance
      Z is TripStartTime – PreviousTime,
      %calculating the allowed interval in seconds
      Interval is Days*24*3600,
    ( %if bad model performance was observed within
      %time interval considered for behaviour change
      (Z =< Interval) -> (
        %if number of bad model performance does not exceed allowed
        %occurrences then set up the new flag for this model
        (ModelFlags < Occurrences-1) ->
          (assertz(model_flag(User,ModelName,TripStartTime,Error)));
        %if number of bad model performance equals to allowed
        %occurrences then this model should be learnt again
        (ModelFlags =:= Occurrences-1) ->
          (assertz(learn_model_final(User,ModelName,TripStartTime,Error)), retractall(model_flag(User,ModelName,_,_)))
      );
      %if bad model performance was observed within time interval exceeding allowed,
      %then set up the new flag for the model.
      (Z > Interval) -> (retractall(model_flag(User,ModelName,_,_)),
          assertz(model_flag(User,ModelName,TripStartTime,Error))
      )
    )
  );
%This model performed well before, but not this time, hence we set up the flag for it
(ModelFlags =:= 0) -> (assertz(model_flag(User,ModelName,TripStartTime,Error)))
).
```

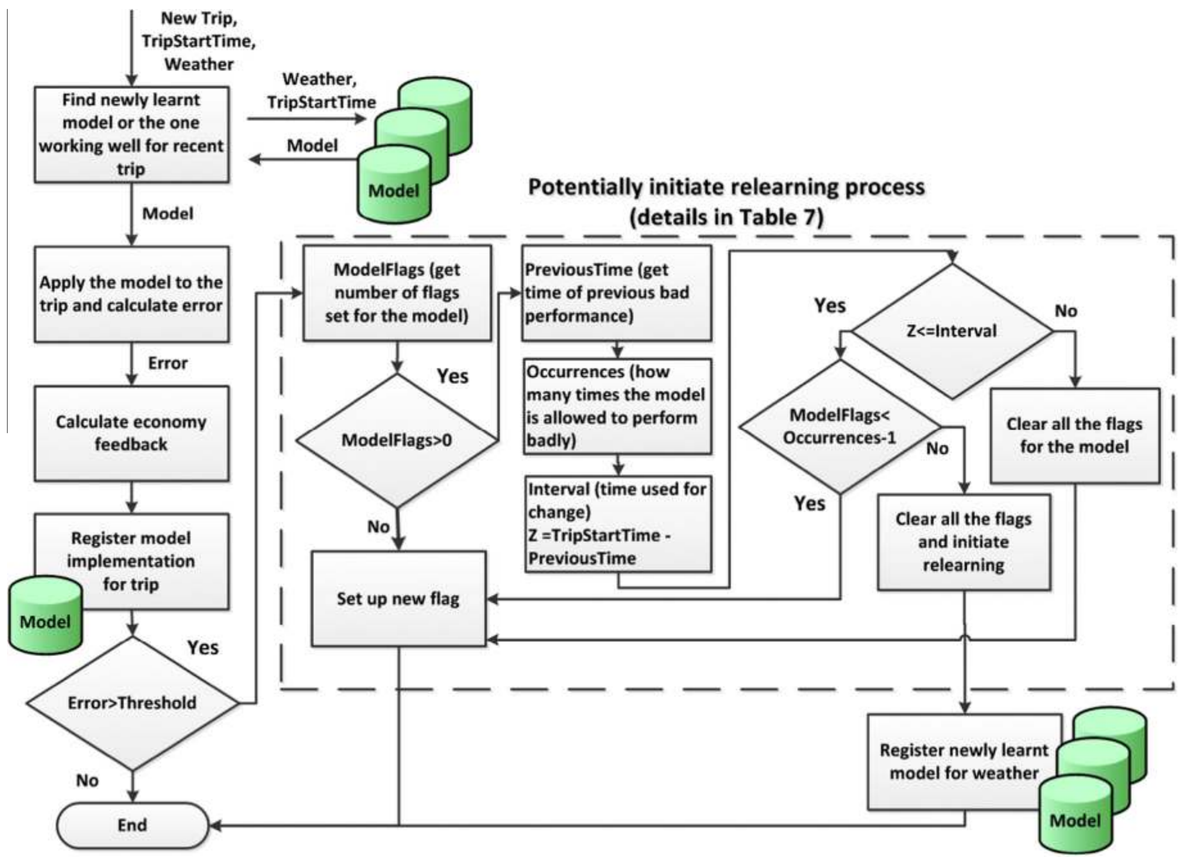

New Trip, TripStartTime, Weather
Find newly learnt model or the one working well for recent trip
Weather, TripStartTime
Model
Model
Model
Apply the model to the trip and calculate error
Error
Calculate economy feedback
Register model implementation for trip
Model
Error>Threshold
Yes
No
End
Potentially initiate relearning process (details in Table 7)
ModelFlags (get number of flags set for the model)
ModelFlags>0
Yes
No
Set up new flag
PreviousTime (get time of previous bad performance)
Occurrences (how many times the model is allowed to perform badly)
Interval (time used for change) Z =TripStartTime - PreviousTime
Z<=Interval
Yes
No
ModelFlags< Occurrences-1
Yes
No
Clear all the flags for the model
Clear all the flags and initiate relearning
Register newly learnt model for weather
Model

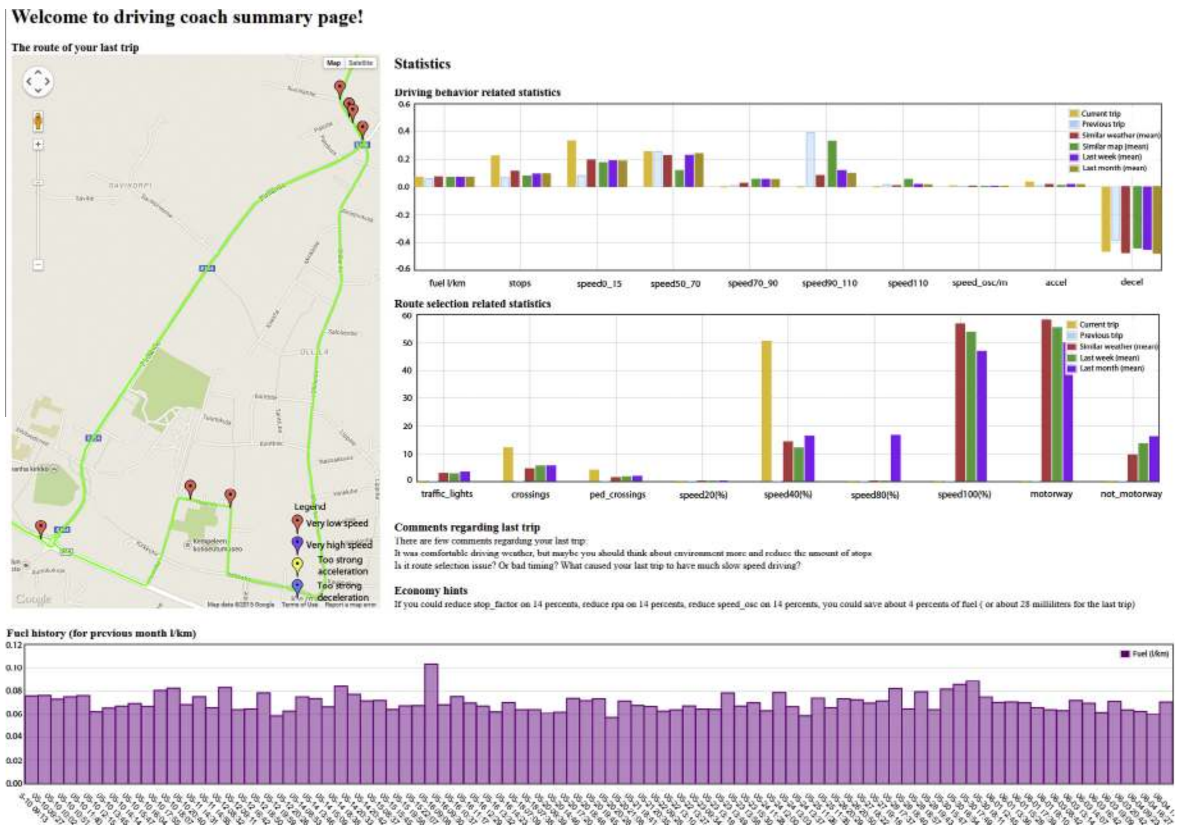
Fig. 11. The process of selecting and possible relearning for fuel economy prediction.

If the system initiates a relearning process, it goes as follows: All the trip data within the specified time or trip window are collected and divided into two sets: training set (80%) to train the model and test set (20%) to evaluate its performance. The amount of trips required for model training is set by the number of explanatory variables. Based on our experience, we subjectively ranked which variables to use for training the models, so we don't feed too large number of possible predictors if the amount of data is small. Of course, to use more explanatory variables, one approach is to extend the time window. After selection of explanatory variables, we train three models (3 model types for either Model A or Model B, depending on model which needs relearning), described in Section 6.1. To select which explanatory variables to keep in the linear regression model (type I), all-subsets regression (from leaps R package) with AIC are considered. Regression tree model (type II) uses in-built selection of predictors and no selection is done for neural network model (type III). The model with best performance on the test set is kept for the future. Some examples of relearned models are presented in Table A4 of Appendix.

7. Driving coach: Web client application and results

Our data retrieval procedure does not allow us to provide feedback during driving; hence, we implemented a Web page summarising and visualising to the driver the information regarding trip driven. This page can be checked after a trip with any device connected to the Internet. This summary page gives the driver a statistical overview of the last trip regarding different aspects of driving, as well as hints to save fuel (see Fig. 12). Web client application is implemented with PHP and deployed on Apache Web server.

As can be seen from Fig. 12, a map demonstrates the route driven, as well as indicates the places where some low speeds and aggressive behaviour was observed. Also, the route segments with poor traffic fluency can be demonstrated to the driver on this map. The right part of the page summarises the trip regarding driving behaviour (upper plot), and selected route characteristics (bottom plot). These plots demonstrate the last trip, together with the same information for the previous trip, averaged values for the trips driven with the same weather and route, as well as averages for the last week and month. This wide range of statistics allows the user to evaluate the trip more thoroughly and look for some patterns in his driving behaviour. Comments summarise the trip evaluation and economy hints give suggestions on what could be improved in order to save some fuel. The bottom plot gives the historical information to the user, so he is able to see the progress.



Comments regarding last trip
There are few comments regarding your last trip:
It was comfortable driving weather, but maybe you should think about environment more and reduce the amount of stops
Is it route selection issue? Or bad timing? What caused your last trip to have much slow speed driving?

Economy hints
If you could reduce stop_factor on 14 percents, reduce rpa on 14 percents, reduce speed_osc on 14 percents, you could save about 4 percents of fuel (or about 28 milliliters for the last trip)

Fig. 12. Web interface of the Driving coach system.[14]

System performance analysis revealed the following information: Information gathering and processing from Data suppliers occur in background and do not affect system performance. However, the map-matching task performed by driving data supplier may take some time depending on the route. Another time-consuming task is retrieving map properties for the driven route, as the road network and road attributes are stored in different tables of the Storage, which have to be merged and searched. Model re-learning is yet another task requiring time; however, this task is performed in the background after the actual decision about the current trip was delivered to the user; hence, it does not affect the overall system performance. Other components of the system demonstrate acceptable processing times. As the Driving coach does not provide real-time driving assistance, we may conclude that delays outlined above, caused by certain components, are acceptable.

At the moment of writing, only one user had a possibility to check the system (with real data coming from Data suppliers), and first user impressions were quite positive.

To evaluate if the proposed solution for fuel economy model learning is appropriate, a comparative study was conducted. In this study, the system collected fuel economy model performance for the same driver and for the same trips with the initially learnt model ensemble as a baseline and with an approach suggested in Section 6.2. In order to operate with real data for real trips and real user, the user started collecting the real driving data (with Driveco device installed to the car, refer to Section 5) at the very beginning of the system design and implementation from February 2013 till November 2014. Gathered data from 2013 were used to build initial fuel economy models. Data from 2014 were used for testing the proposed system. To make the model evaluation comparable, we set the same ensemble size for both cases ($N = 3$). We use model ensemble in Driving coach because of possible measurement outliers and complexity of the domain in general. Model ensemble allows selecting the model with the best performance for final fuel economy recommendation. Hence, when initial models are used, the one demonstrated the smallest fuel prediction error is used for recommendation; the same approach is used for selection of the best model with changed context. To detect driving behaviour change, we use the following settings (refer to Section 6.2 for details): threshold = 5, occurrences = 20, interval = 14. Basically, this means that if within two weeks (14 days) the model gives high fuel prediction error (more than 5%) about 20 times, hence, we register driving behaviour change. For this study, 80 most recent trips are used for relearning. These trips are randomly divided into train and test sets (80% and 20% correspondingly). We compare fuel prediction model error in percent, as Fig. 13 demonstrates.

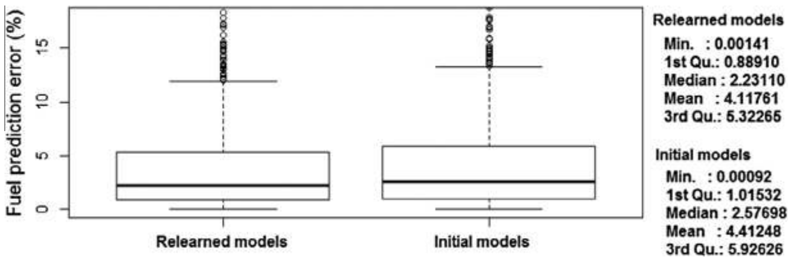


Fig. 13. Fuel prediction model performance comparison (1077 trips, $N = 3$).

As can be seen from Fig. 13, generally, there is no difference in terms of fuel prediction error for relearned models in comparison with initial ones. Fig. 14 demonstrates the models performance based on months and air temperature distribution. From this figure, an interesting observation may follow. The system had to accumulate good performance models for different kinds of weather. And at the beginning, the system did not perform better than initial models because of high variance in air temperature and overall weather situation (March and April cases). With the time, the system collected some models and was able to perform better than initial models (May case). However, June required accumulating the different weather models again (May still had some very different weather conditions, see Fig. 14); hence, relearned models demonstrated a worse result than initial ones. From July, relearned models demonstrated better performance in terms of fuel prediction error than initial ones, even when the weather differed a lot (October and November cases). This is due to the fact that models from diverse weather conditions were accumulated. Cases like June may indicate that the amount of trips used for relearning (80 in this experiment) could be reduced. All these fluctuations together sum up to a more or less similar result observed from Fig. 13.

Conducted analysis revealed that in terms of fuel prediction the suggested approach may work better than initial models if some time is given to accumulate the models. However, conducted analysis has the limitation that the same driver was used to train initial models. This implies that initial models grasped driving patterns of this driver very well. Longer

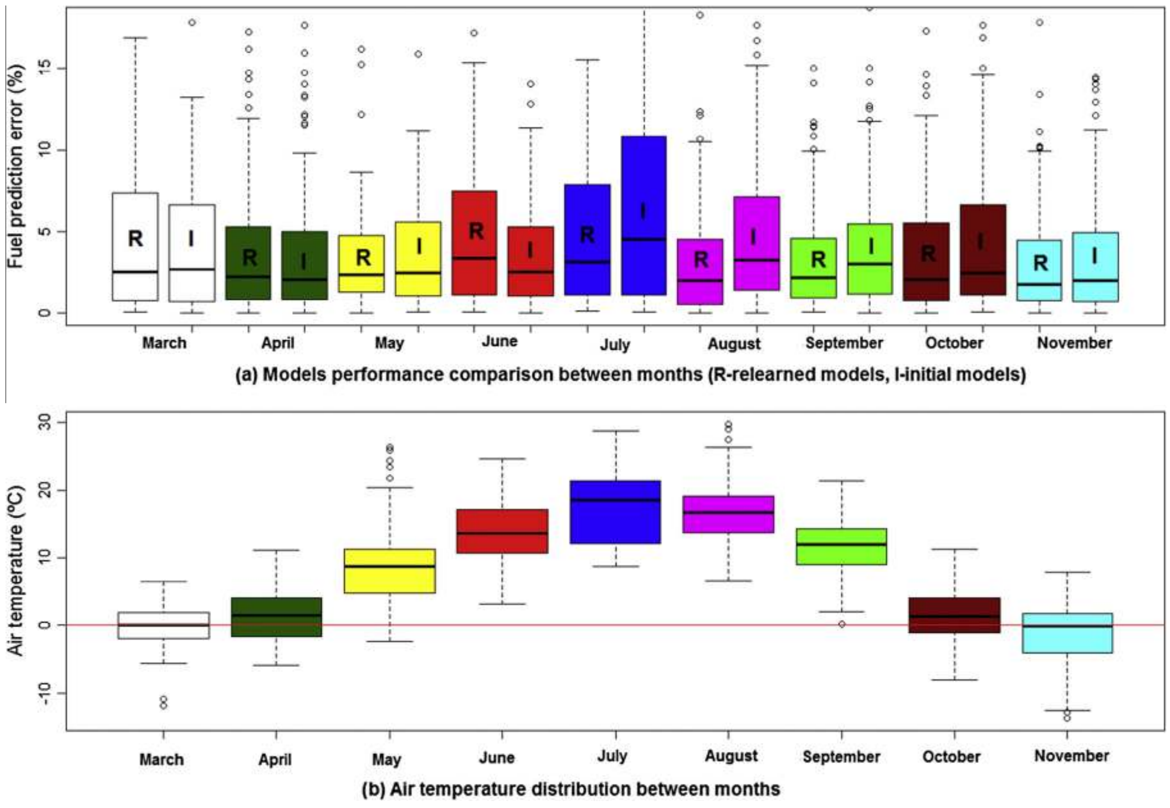


Fig. 14. Performance of models (a) and air temperature distribution (b) between months (1077 trips, $N = 3$).

studies are required to prove conclusions. Finally, parameter selection requires a comprehensive study, as this affects results a lot.

Generally, in order to analyse how well the proposed system supports drivers, a long time user test is required with different drivers and different weather conditions. Moreover, the user interface requires work to present the essential information clearly without too long learning curve.

8. Conclusions and future work

Development of driving assistant tools is an actively researched field in both industry and academia. We study this problem from the ubiquitous computing point of view, where the driving situation is considered as context and the driving assistant system uses this context to support the driver. Moreover, we emphasise the importance of self-introspection capabilities for driving assistant systems, as well as use of diverse information sources for more thorough analysis. We proposed a reference architecture for a context-aware driving assistant system, as well as prototyped it with a driving assistant system called Driving coach. Driving coach is a fully functional system, which collects diverse information, analyses collected data and provides a driver with personalised hints to improve the driving.

Driving coach does not provide real-time support, as some related work like van der Voort et al. (2001) and Wu et al. (2011); from the time perspective, it is closer to Fiat solution[4], where the trip can be observed afterwards. Such a solution gives rich capabilities for analyses as well as capabilities to integrate heterogeneous data sources. Driving coach uses diverse real context information to perform the analysis, namely weather, road geometry, driving behaviour, and traffic situation information. With such a rich set of context sources, it is unique, even though other work was reported on fusing different sources of information, e.g., by Mensing et al. (2013). Moreover, the system looks for personalisation and adapts itself to serve the user better. In this sense, it is somewhat close to, e.g., Syed et al. (2009).

The current version of Driving coach partially implements the Meta-level control framework, suggested by Gilman and Riekki (2012). Implementation of this framework in the traffic related domain is challenging. First, the data retrieving procedure we used limited us for developing an offline support system. Even though an offline system provides advantages of thorough trip analyses, it makes it challenging to retrieve user feedback, and hence, to evaluate how well the decisions of the system support the user. Second, the domain itself limits the diversity of devices and user interaction modes.

Despite the challenges, separation of the tasks of a ubiquitous driving assistant system to Object and Meta level seems to be feasible. First, it provides maintenance advantages for the system, as each layer encapsulates the tasks it is responsible for. Moreover, the system can be tailored to different environments and users without changes in the core Object-level tasks. This can be achieved by setting the rules the Meta level uses for controlling the Object level tasks. Also Meta-level layer can be shared between different systems in a ubiquitous environment, improving reusability.

Presented solutions for both Object and Meta level functionality seem to be feasible; however, more thorough tests with different weather conditions are required to evaluate them. We find two weeks as a convenient window to obtain distributions for different driving factors. Such a time frame guarantees enough trips and demonstrates the most current driver's skills. However, the time window could also be configured based on the amount of trips. Usage of distributions to perform the driving trip evaluation with respect to driving factors seems to be a feasible instrument, as it demonstrates which factor values should be improved for a certain driver. Moreover, we consider shifting the factor region borders to release or strength evaluation very useful, especially, for bad driving conditions. Fuel prediction models used in the first implementation of Driving coach are somewhat simplistic, but even as such they are able to produce acceptable results. The proposed method for monitoring performance of fuel prediction models and controlling the relearning process aims to support adequate system behaviour in changing situations, namely user progress and weather changes. Initial experiments demonstrated that proposed solution provides better performance in comparison to static models if some time is given to accumulate models for changing weather conditions.

We see several directions for future work. First, to be able to sense a large amount of drivers, we need to think about more efficient solutions for data processing and analysis. Second, we will continue the work towards development of intelligent functionality of Driving coach. We will develop personalised cost function to evaluate the selected route and compare it with alternative routes to connect the origin and destination. With this approach, we will be able to evaluate whether the choice made by the driver is the best one. We will improve fuel consumption models (based on more detailed data) and generate a more flexible advice system. Also, as we already have many sources of information, we could use association rule mining techniques to discover driving related patterns, for example. Third, new functionality for client applications can be developed. For example, a driver could annotate a trip afterwards. This is useful for cases like, if a driver every morning gives a lift to the kids of a friend to the school, this causes stops which cannot be avoided. In order to help the system to understand such a situation correctly, the system could provide GUI tools for the driver to describe such cases. Hence, the learning algorithms would be aware of the situation. Fourth, a mobile client with a tailored user interface could be used for real-time interaction with the driver (when such facility will be available to Driving data supplier). However, this would set even more strict real-time requirements for the system. Finally, we would like to test our system with real drivers, to get their feedback about the system.

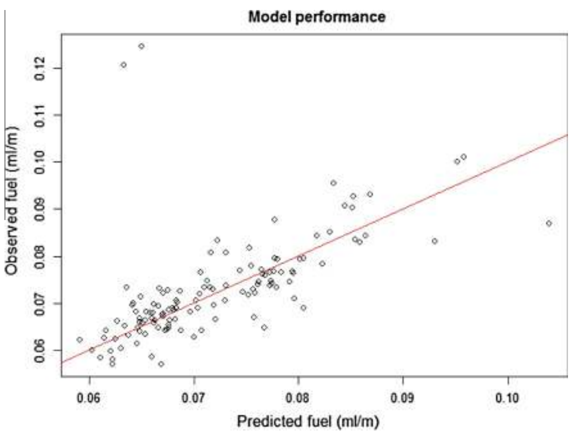
Acknowledgments

This work was supported by Tekes as part of the Data to Intelligence Program of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business). Also, we would like to thank Marjo Hippi from Finnish Meteorological Institute for her help regarding weather-based issues for the system.

Appendix A

See Tables A1–A4.

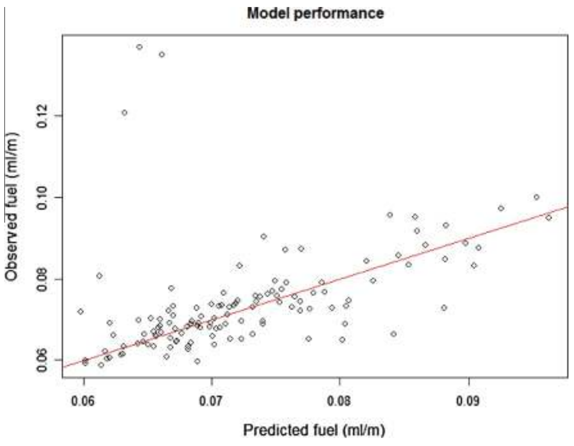
Table A1
Examples of initial linear models.

Model	Model characteristics (train set)	Model performance (test set)
A	Fuel ∼ speed50_70 + speed15_30 + stop_factor^2 + speed0_15^2 + speed_osc + surfaceTemperature	

Coefficients:	Estimate	t value	Pr(>|t|)
Intercept	6.259e-02	60.382	< 2e-16
speed50_70	-7.671e-03	-3.713	0.000228
speed15_30	2.145e-02	6.010	3.64e-09
stop_factor^2	7.649e-02	5.217	2.69e-07
speed0_15^2	5.520e-02	9.036	< 2e-16
speed_osc	1.397e+00	6.162	1.51e-09
surface Temperature	-2.125e-04	-8.773	< 2e-16

Residual standard error: 0.006224 on 488 DF
Multiple R-squared: 0.6669,
Adjusted R-squared: 0.6628
F-statistic: 162.8 on 6 and 488 DF, *p*-value: <2.2e−16

RMSE = 0.0087

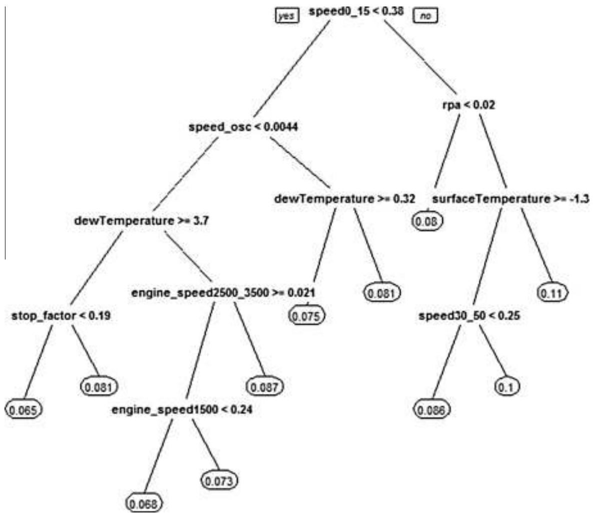
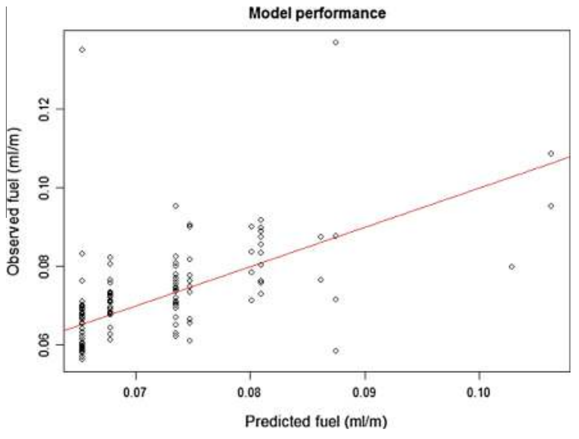
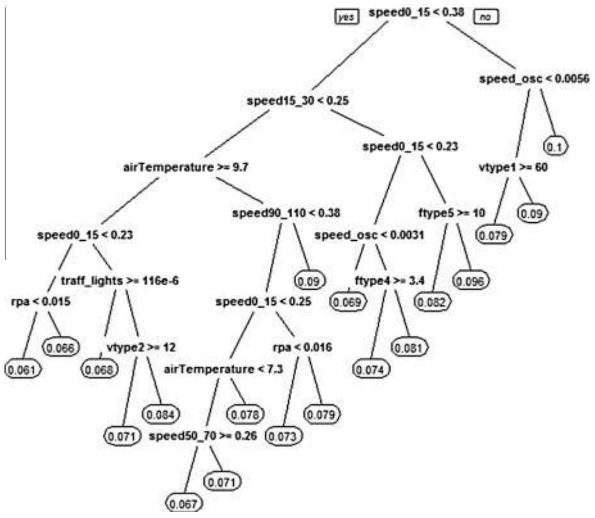
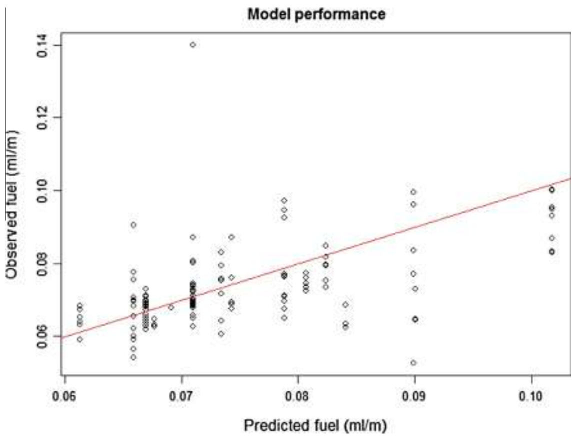
Model	Model characteristics (train set)	Model performance (test set)
B	Fuel ∼ speed50_70 + speed15_30 + stop_factor^2 + speed0_15^2 + speed_osc + surfaceTemperature + traff_lights	

Coefficients:	Estimate	t value	Pr(>|t|)
Intercept	6.281e-02	69.535	< 2e-16
speed50_70	-7.819e-03	-4.217	2.95e-05
speed15_30	1.874e-02	5.669	2.46e-08
stop_factor^2	6.558e-02	5.229	2.53e-07
speed0_15^2	6.147e-02	9.766	< 2e-16
speed_osc	1.506e+00	7.238	1.79e-12
surface Temperature	-2.148e-04	-9.891	< 2e-16
traff_lights	-7.524e-01	-1.979	0.0484

Residual standard error: 0.005499 on 487 DF
Multiple R-squared: 0.7239,
Adjusted R-squared: 0.72
F-statistic: 182.4 on 7 and 487 DF, *p*-value: <2.2e−16

RMSE = 0.0117

Table A2
Examples of initial regression tree models[a].

Model	Model characteristics (train set)	Model performance (test set)
A	 yes speed0_15 < 0.38 no speed_osc < 0.0044 rpa < 0.02 dewTemperature >= 3.7 dewTemperature >= 0.32 surfaceTemperature >= -1.3 0.08 stop_factor < 0.19 engine_speed2500_3500 >= 0.021 0.075 0.081 0.11 speed30_50 < 0.25 0.065 0.081 engine_speed1500 < 0.24 0.087 0.086 0.1 0.068 0.073	 RMSE = 0.0105
B	 yes speed0_15 < 0.38 no speed_osc < 0.0056 speed15_30 < 0.25 speed0_15 < 0.23 0.1 airTemperature >= 9.7 vtype1 >= 60 speed90_110 < 0.38 ftype5 >= 10 0.09 speed0_15 < 0.23 speed_osc < 0.0031 0.079 traff_lights >= 116e-6 0.09 ftype4 >= 3.4 0.096 rpa < 0.015 speed0_15 < 0.25 0.069 0.082 0.066 vtype2 >= 12 rpa < 0.016 0.081 0.061 0.068 airTemperature < 7.3 0.074 0.084 0.078 0.079 0.071 speed50_70 >= 0.26 0.073 0.071 0.067	 RMSE = 0.0109

[a] Tree plots of Tables 2 and 4 are drawn with rpart.plot package: Stephen Milborrow. Rpart.plot: Plot rpart models, 2014. R package.

Table A3
Examples of neural network initial models.

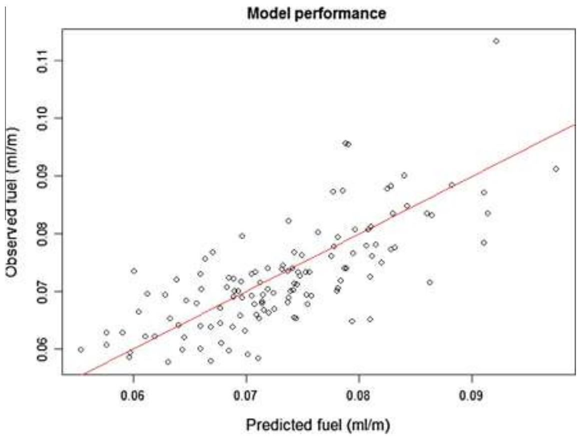
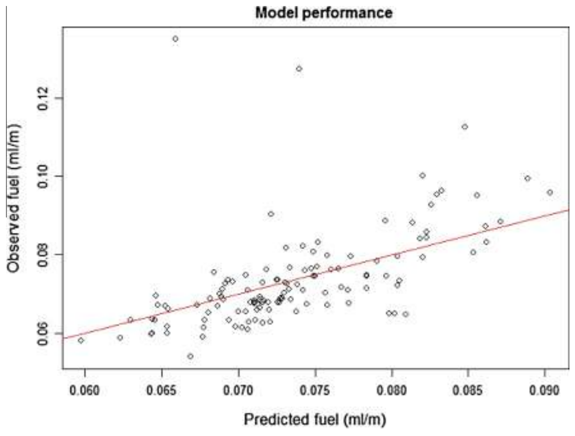
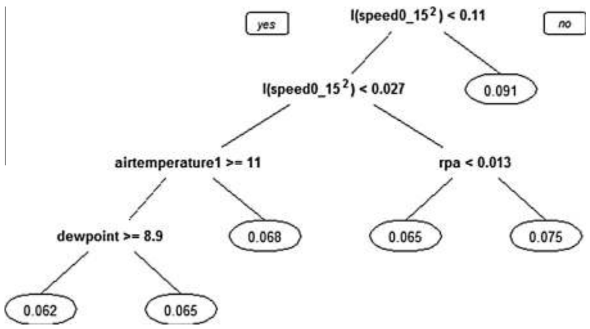
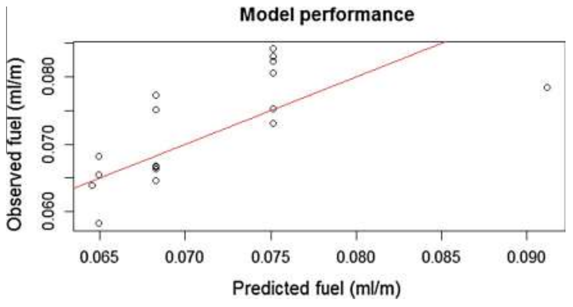
Model	Model characteristics (train set)	Model performance (test set)
A	20-15-1 network with 331 weights (decay = 0.02), RMSE = 0.0096, R-squared = 0.45	 RMSE = 0.0063
B	7-70-1 network with 631 weights (decay = 0.02), RMSE = 0.0089, R-squared = 0.46	 RMSE = 0.0102

Table A4
Examples of relearned models.

Model | Model characteristics (train set) | Model performance (test set)

A





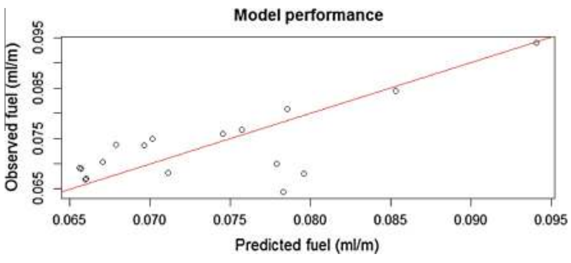
RMSE = 0.0059

B

Fuel ~ Dew point temperature + Friction + rpa + vtype1 + vtype4 + speed0_15^2

Coefficients:	Estimate	t value	Pr(>|t|)
Intercept	1.214e-01	6.516	2.03e-08
Dew point temperature	-2.981e-04	-2.770	0.007551
Friction	-5.621e-02	-2.491	0.015655
rpa	1.532e-01	2.358	0.021840
vtype1	-1.468e-04	-4.160	0.000108
vtype4	-1.332e-03	-2.430	0.018291
speed0_15^2	1.290e-01	11.328	3.20e-16

Residual standard error: 0.005887 on 57 DF
Multiple R-squared: 0.7876,
Adjusted R-squared: 0.7653
F-statistic: 35.23 on 6 and 57 DF, *p*-value: <2.2e−16



RMSE = 0.0055

References

Abdullah, R., Hussain, A., Warwick, K., Zayed, A., 2008. Autonomous intelligent cruise control using a novel multiple-controller framework incorporating fuzzy-logic-based switching and tuning. Neurocomputing 71 (13–15), 2727–2741.
Araujo, R., Igreja, A., de Castro, R., Araujo, R.E., 2012. Driving coach: A smartphone application to evaluate driving efficient patterns. In: Intelligent Vehicles Symposium (IV), 2012 IEEE, pp. 1005–1010.
Bikakis, A., Antoniou, G., 2010. Rule-based Contextual Reasoning in Ambient Intelligence. In: Proc. of the International Symposium on Semantic Web Rules (RuleML 2010), LNCS, vol. 6403, Springer, 2010, pp. 74–88.
Boriah, S., Chandola, V., Kumar, V., 2008. Similarity measures for categorical data: a comparative evaluation. In: Proc. of the 2008 SIAM International Conference on Data Mining, pp. 243–254.
Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C., 2005. On map-matching vehicle tracking data. In: Proc. of the 31st International Conference on Very Large Data Bases (VLDB '05), pp. 853–864.
Brundell-Freij, K., Ericsson, E., 2005. Influence of street characteristics, driver category and car performance on urban driving patterns. Transport. Res. Part D: Transport Environ. 10 (3), 213–229.
Chen, A., 2005. Context-aware collaborative filtering system: predicting the user's preference in the ubiquitous computing environment. Location Context-Awareness Lect. Notes Comput. Sci. 3479, 244–253.
Cox, M., Raja, A., 2008. Metareasoning: a manifesto. In: Proc. Metareasoning: Thinking about Thinking Workshop Held within 23 AAAI Conf. on Artificial Intelligence.
Cox, M., Raja, A., 2011. Metareasoning: Thinking about Thinking. The MIT Press.
Dey, A.K., 2001. Understanding and using context. Personal Ubiquitous Comput. 5 (1), 4–7.
Eren, H., Makinist, S., Akin, E., Yilmaz, A., 2012. Estimating driving behavior by a smartphone. In: Intelligent Vehicles Symposium (IV), 2012 IEEE, pp. 234–239.
Ericsson, E., 2001. Independent driving pattern factors and their influence on fuel-use and exhaust emission factors. Transport. Res. Part D: Transport Environ. 6 (5), 325–345.
Ericsson, E., Larsson, H., Brundell-Freij, K., 2006. Optimizing route choice for lowest fuel consumption – potential effects of a new driver support tool. Transport. Res. Part C: Emerg. Technol. 14 (6), 369–383.
European Environment Agency (EEA), Air quality in Europe – 2011 report, 2011. <http://www.eea.europa.eu/publications/air-quality-in-europe-2011/at_download/file> (accessed 14.07.14).
Eurostat, European Comission, "Panorama of transport," 2009. <http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-DA-09-001/EN/KS-DA-09-001-EN.PDF> (accessed 14.07.14).
Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Comput. Surv. 46 (4). Article 44.

Gilman, E., Riekki, J., 2012. Is there meta-level in smart spaces? In: Proc. IEEE Pervasive Computing and Communications Workshops (PERCOM Workshops), 19–23 March 2012. pp. 88–93.

Gilman, E., Sánchez, I., Saloranta, T., Riekki, J., 2010. Reasoning for Smart Space Application: Comparing Three Reasoning Engines CLIPS, Jess and Win-prolog. In: Proc. of the IEEE 10th International Conference on Computer and Information Technology (CIT), pp. 1340–1345.

Gonder, J., Earleywine, M., Sparks, W., 2011. Final Report on the Fuel Saving Effectiveness of Various Driver Feedback Approaches. National Renewable Energy Laboratory, US Department of Energy, Office of Energy Efficiency and Renewable Energy (2011).

Guan, T., Frey, C.W. 2012, EXPERT: A Driver Assistance System for Fuel Efficient Driving. In: Proc. of the 3rd International Conference on Machine Control & Guidance, Stuttgart, March 27–29.

Hellström, E., Ivarsson, M., Åslund, J., Nielsen, L., 2009. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. Control Eng. Pract. 17 (2), 245–254.

Johnson, D.A., Trivedi, M.M. 2011. Driving style recognition using a smartphone as a sensor platform. In: Proc. of 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1609–1615.

Khayyam, H., Nahavandi, S., Davis, S., 2012. Adaptive cruise control look-ahead system for energy management of vehicles. Expert Syst. Appl. 39 (3), 3874–3885.

Koskimaki, H., Juutilainen, I., Laurinen, P., Roning, J., 2008. Detection of correct moment to model update. Lect. Notes Electric. Eng.: Inform. Control, Automat. Robot. 24, 87–94.

Kuhn, M., 2008. Building predictive models in R using the caret package. J. Stat. Softw. 28 (5), 1–26.

Mensing, F., Bideaux, E., Trigui, R., Tattegrain, H., 2013. Trajectory optimization for eco-driving taking into account traffic constraints. Transport. Res. Part D: Transport Environ. 18, 55–61.

Meyer, D., Hornik, K., 2009. Generalized and customizable sets in R. J. Stat. Soft. 31 (2), 1–27.

Parlak, A., Islamoglu, Y., Yasar, H., Egrisogut, A., 2006. Application of artificial neural network to predict specific fuel consumption and exhaust temperature for a Diesel engine. Appl. Therm. Eng. 26 (8–9), 824–828.

Paz, A., Peeta, S., 2009. On-line calibration of behavior parameters for behavior-consistent route guidance. Transport. Res. Part B: Methodol. 43 (4), 403–421.

Sivak, M., Schoettle, B., 2012. Eco-driving: strategic, tactical, and operational decisions of the driver that influence vehicle fuel economy. Transport Policy, 96–99.

Steiniger, S., Hunter, A.J.S., 2013. The 2012 free and open source GIS software map – a guide to facilitate research, development, and adoption. Comput. Environ. Urban Syst. 39 (May), 136–150.

Syed, F.U., Filev, D., Tseng, F., Ying, H., 2009. Adaptive real-time advisory system for fuel economy improvement in a hybrid electric vehicle. In: Annual Meeting of the North American Fuzzy Information Processing Society, June, pp. 1–7.

The AAA Foundation for Traffic Safety, 2009. Aggressive Driving: Research update. <http://www.aaafoundation.org/pdf/AggressiveDrivingResearchUpdate2009.pdf> (accessed 14.07.14).

Tsymbal, A., 2004. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland, April 2004.

Vagg, C., Brace, C.J., Hari, D., Akehurst, S., Poxon, J., Ash, L., 2013. Development and field trial of a driver assistance system to encourage eco-driving in light commercial vehicle fleets. IEEE Trans. Intell. Transport. Syst. 14 (2), 796–805.

van der Voort, M., Dougherty, M.S., van Maarseveen, M., 2001. A prototype fuel-efficiency support tool. Transport. Res. Part C: Emerg. Technol. 9 (4), 279–296.

Wu, C., Zhao, G., Ou, B., 2011. A fuel economy optimization system with applications in vehicles with human drivers and autonomous vehicles. Transport. Res. Part D: Transport Environ. 16 (7), 515–524.

Zhou, X., Huang, J., Lv, W., Li, D. 2013. Fuel Consumption Estimates Based on Driving Pattern Recognition, Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom). In: IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp. 496–503.