Ontological and Machine Learning Approaches for Managing Driving Context in Intelligent Transportation

Manolo Dulva Hina[1], Clement Thierry[1], Assia Soukane[1] and Amar Ramdane-Cherif[2]

[1]*ECE Paris School of Engineering, 37 quai de Grenelle, 75015 Paris, France*

[2]*LISV Laboratory, Université de Versailles St-Quentin-en-Yvelines, 10-12 avenue de l'Europe, 78140 Velizy, France*

Keywords: Internet of Things, Ontology, Machine Learning, Driving Context, Smart Vehicle, Cognitive Informatics.

Abstract: In this paper, a novel approach of managing driving context information in smart transportation is presented. The driving context refers to the ensemble of parameters that make up the contexts of the environment, the vehicle and the driver. To manage this rich information, knowledge representation using ontology is used and through it, such information becomes a source of knowledge. When this context information (i.e. basically a template or model) is instantiated with actual instances of objects, we can describe any kind of driving situation. Furthermore, through ontological knowledge management, we can find the answers related to various queries of the given driving situation. A smart vehicle is equipped with machine learning functionalities that are capable of classifying any driving situation, and accord assistance to the driver or the vehicle or both to avoid accident, when necessary. This work is a contribution to the ongoing research in safe driving, and a specific application of using data from the internet of things.

1 INTRODUCTION

The World Health Organization's statistics on global traffic accident (WHO 2015) are gruesome:

- Every year, about 1.24 million people die each year in road traffic accidents;
- Road traffic injury is the leading cause of death on young people, aged 15–29 years;
- Half of those dying on the world's roads are "vulnerable road users", namely the pedestrians, cyclists and motorcyclists;
- In 2008, in the USA, old people accounted for 15% of all traffic fatalities and 18% of all pedestrian fatalities; and
- If no remedy is employed, road traffic accidents are predicted to result in the deaths of around 1.9 million people annually by 2020.

Here lies the importance of researches on intelligent transportation intended to reduce traffic accident, such as ours. An intelligent transportation (An, Lee et al. 2011, Naja 2013) denotes advanced application embodying intelligence to provide innovative services related to modes of transport and traffic management, enabling various users to be better informed and makes safer, more coordinated, and smarter use of transport networks.

Recently, the Internet of things (IoT) (Ashton 2009, Atzori, Iera et al. 2010), that is, the idea of connecting all things in the world to the Internet, have been integrated into vehicles in order that an intelligent transport would be able to use them as parameters to tools for safe driving, green driving and comfortable driving. Our vision of an innovative vehicle is shown in Figure 1. Inside this vehicle is an intelligent architecture with three main components: (1) Embedded System, (2) Intelligent System, and (3) Network and Real-time System.

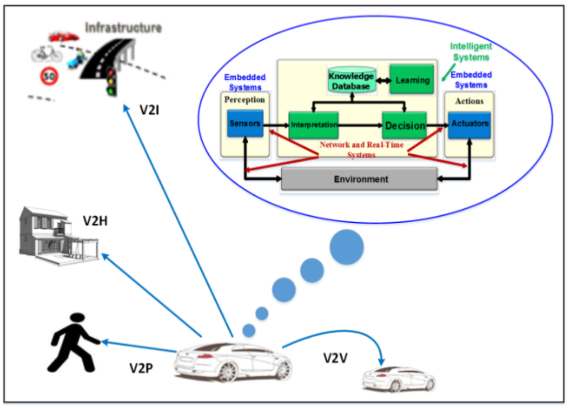

Figure 1: Smart services for an intelligent vehicle.

Hina M., Thierry C., Soukane A. and Ramdane-Cherif A.
Ontological and Machine Learning Approaches for Managing Driving Context in Intelligent Transportation.
DOI: 10.5220/0006580803020309
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD 2017)*, pages 302-309
ISBN: 978-989-758-272-1

The cloud computing infrastructure (Oracle 2009, Yousif 2017) which has become the de-facto hosting platform for all smart applications also hosts the system.

In a smart vehicle, such vehicle is capable of the following: V2V (vehicle-to-vehicle communication), V2P (vehicle-to-person communication), V2I (vehicle-to-infrastructure communication) and V2X (where X represents practically anything such as home, in which case V2X becomes V2H, i.e. the IoT at home being able to communicate to the driver some information, such as when it is running low on some items on the refrigerator, for example).

2 RELATED WORKS

The Internet of Things (IoT) has been considered the technology that will integrate classical networks and networked objects (Miorandi, Sicari et al. 2012). Through IoT, it is expected that things can be identified automatically, can communicate with each other, and out of this, we can make better decisions for ourselves. One important question in this regard is how do we convert the data generated or captured by IoT into knowledge that we may be able to use for our convenience? An important work pertinent to this question is that of (Tsai, Lai et al. 2014). They proposed knowledge discovery in databases (KDD) and data mining technologies to find the information hidden in the data of IoT. Recent researches do focus on developing effective data mining technologies for the IoT. The results described in (Bin, Yuan et al. 2010) show that data mining algorithms can be used to make IoT more intelligent, thus providing smarter services. Our Embedded System's task is related to this area. In the area of intelligent transportation, the connected and autonomous vehicles are a technological revolution, combining radical changes in the design of road vehicles and in the understanding of their interactions with the networked infrastructure. The core science and technology required to support cyber-physical vehicles (Brioschi, Hina et al. 2016) are essential for future economic competitiveness. This is where this work lies. We intend to contribute to its advancement.

3 THE DRIVING MODEL

Ontology (Dice 2017) is the structure of concepts and relations representing the meaning of a given domain. In other words, ontology is a description of things that exist and how they relate to each other (Obrst 2003). Ontologies are partial and formal specifications of a conceptualization. Ontologies are formal because they are expressed as formalism with formal semantics (Zhao, Ichse et al. 2015). Ontology is used as knowledge representation in this work. For visualization purposes, we use a Protégé plug-in, VOWL (visual notation for OWL ontologies) (VOWL) in order to describe the ontology components. OWL ontology consists of individuals, properties, and classes.

3.1 The Driving Context

The driving context is the representation of a traffic/driving situation. Ontology is used for modelling, putting in place a common conceptual language between the driving situation and the assistance system. Briefly, the driving context is the result of the fusion of three main contexts: the vehicle context, the driver context and the environment context. It is depicted in Figure 2.

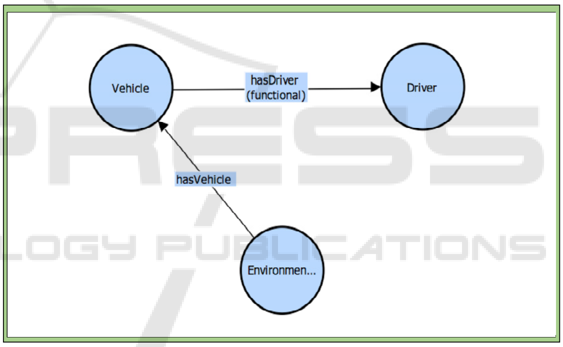

Figure 2: Ontological representation of the driving context.

As shown, the "*Environment*" is the class used to describe the external environment where human-vehicle interactions take place. The "*Vehicle*" is the class that represents the vehicle in consideration, interacting with its driver. The "*Driver*" is the class that describes the driver of a simulated vehicle. "*Environment*" and "*Vehicle*" are related through *hasVehicle* object property while the "*Vehicle*" and the "*Driver*" are related to each other through *hasDriver* object property. This ontology is capable of modeling all possible types of driving situations. Through ontology, we can find answers to a query related to a driving situation.

3.2 Modeling the Context of the Driver

The ontological representation of the context of the driver is shown in Figure 3. The *Driver* class is related

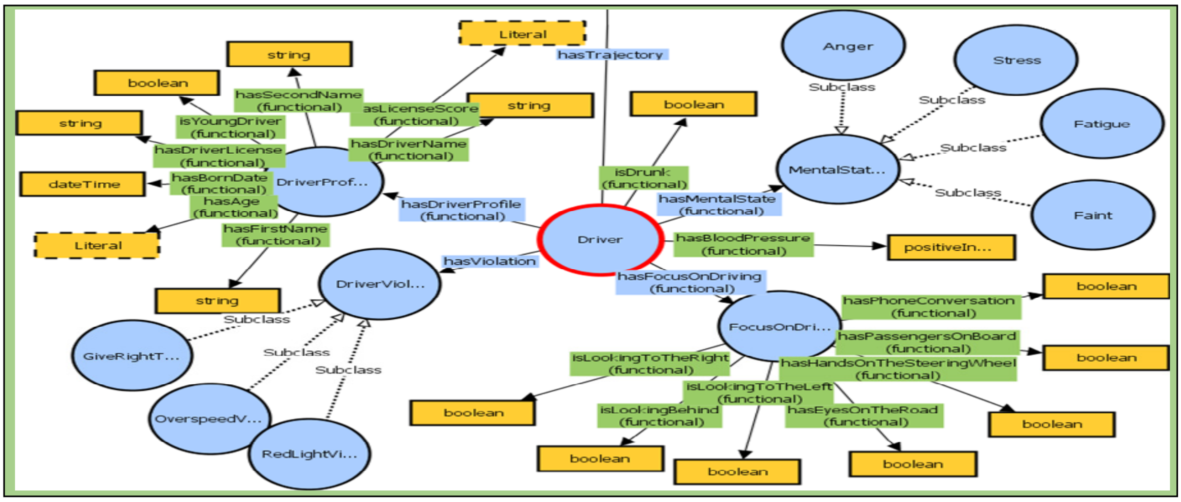

Figure 3: The ontological representation of the driver's context.

to many other classes that describe the driver's context in the ontology, such as:

- *DriverProfile:* it is related to the "*Driver*" through "*hasDriverProfile*" functional property.
- *MentalState*: describes the mental state that can negatively influence the behavior of the driver while driving a vehicle. It has three subclasses: '*Fatigue*', '*Stress*', and '*Faint*'. Each one has one of these values: '*High*', '*Average*' or '*Low*'.
- *FocusOnDriving*: has data properties '*hasEyesOnTheRoad*', '*isLookingToTheRight*', '*isLookingToTheLeft*', *hasPhoneConversation*, '*hasHandsOnSteeringWheel*', etc.
- *DriverViolation*: The violations can be of type '*GiveRightToPass*', '*RedLightViolation*' and '*OverSpeedViolation*.

3.3 Modeling the Context of the Vehicle

The class "*Vehicle*" is a subclass of *MovingObject* template. As shown in Figure 4, the Vehicle class has three subclasses: '*Car*', '*Truck/Bus*' or '*MotorBike*'. A vehicle has some data properties, such as '*hasPlateNumber*' and '*hasInsurance*'. The Vehicle class is linked to other classes, such as:

- *Cockpit*: contains the status of all elements that are found in a vehicle's cockpit. For example, '*hasWindowsOpen*' is a data property that has a Boolean value;
- *Components*: contains all components that guarantee a good driving experience. Among these subclasses are "*DirectionIndicator*" (values are '*NoIndicator*' '*RightIndicator*', '*LeftIndicator*', and '*DoubleIndicators*'), "*TyresPression*", "*LubricantTemperature*", "*EngineLubricantLevel*" ('*Low', 'Half*' or '*Full*'), and "*FuelQuantity*". It also has some Boolean properties indicating if some components are active or not. Example is '*hasFogLightsOn*';
- *TechnicalData*: it is made up of three subclasses, namely "*FuelType*" ('*Petrol*', '*Diesel*', '*Electricity*' and '*GPL*'), "*EmissionClass*" ('*euro0*', '*euro1*', …, '*euro6*') and "*TractionType*" ('*Front-WheelDrive*', '*Rear-WheelDrive*', '*All-WheelDrive*');
- *VehicleObject*: this refers to the class of other vehicles on the road and is related to our vehicle via property "*hasPossibleCollision*".

3.4 Modeling the Context of the Environment

The Environment (see Figure 5) describes all the entities that are present in the external setting where the human-vehicle interaction takes place. Here, the environment is an abstract class and general concept made up of cities where vehicles, and moving and non-moving objects are present. The classes related to the Environment are given as follows:

- *City:* In this work, an Environment is an area or region where we can find many cities. A city has two data properties, namely '*hasCityName*' and '*hasLimitedTrafficZone*' which is a Boolean value indicating if the city can be accessed only during some intervals of the day;
- *DistrictArea*: contains different districts of a city. The position of the "*Driver*" is stored in the "*PositionArea*", a subclass of "*Physics*" and equivalent to "*DistrictArea*";

- *Road*: a road has many data properties, such as '*hasMinSpeedLimit*', '*hasMaxSpeedLimit*', '*hasNumberOfLanes*', '*hasContinuousLine*' and '*hasLength*'. A road is made up of three subclasses: '*Urban*', '*ExtraUrban*', and '*Highway*'. Every subclass of a road has its minimum and maximum speed limit;
- *RoadProperty*: it stores all the properties that belong to a particular road. This includes "*Visibility*" ('*Low*', '*Average*' or '*High*'), "*Weather*" ('*Fog*', '*Sun*', '*Rain*' and '*Snow*'), "*AccidentHistory*" ('*Unusual*' or '*Frequent*'), "*TrafficCongestionHistory*" ('*Low*', '*Average*' or '*Intense*') and "*CurrentTrafficCongestion*" ('*Low*', '*Average*' or '*Intense*');
- *Lane*: represents different lanes of a road ;
- *Position*: contains the exact position of the referenced object. It has two data value properties: '*hasLatitude*' and '*hasLongitude*;
- *Time*: it has data value properties, such as '*hasDate*' and '*hasTime*'

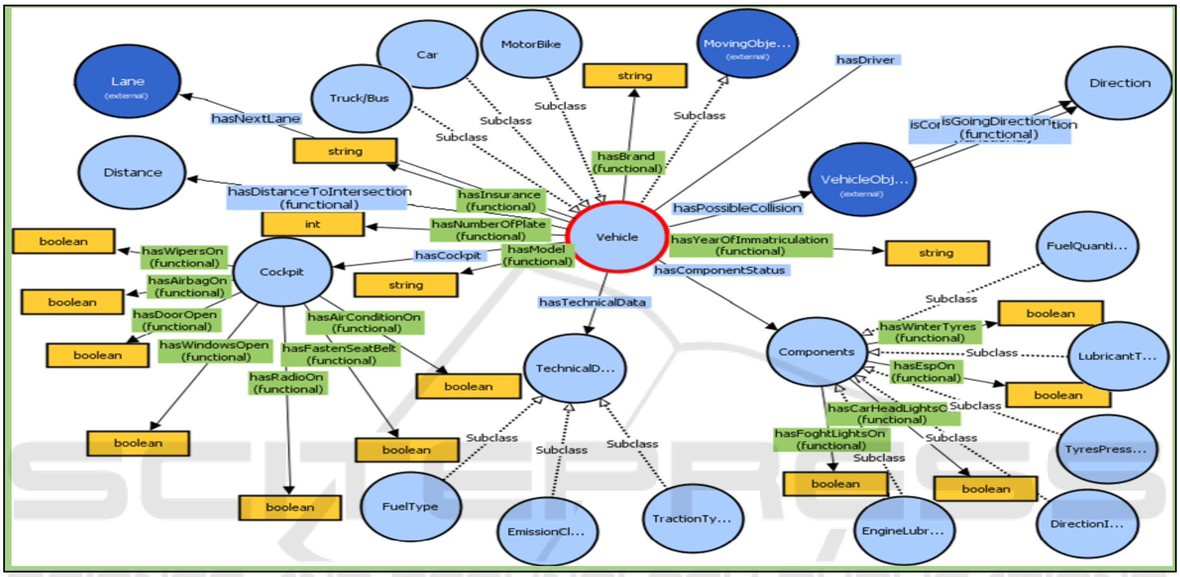

Figure 4: The representation of the context of the vehicle.

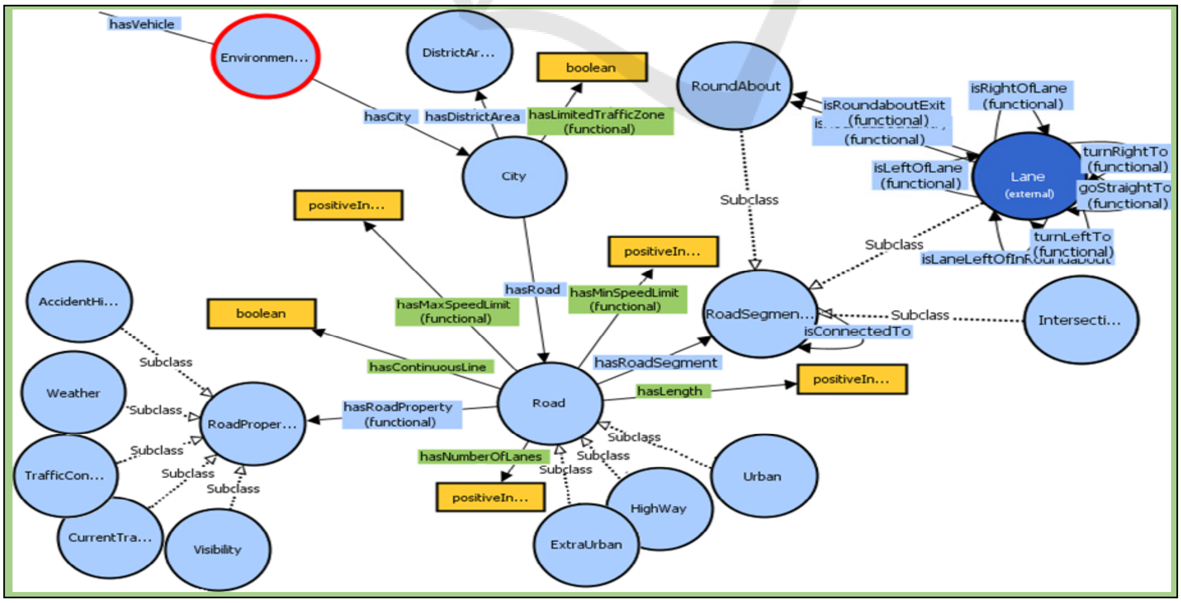

Figure 5: The ontological representation of the context of the environment.

3.5 Multimodal Fusion and Fission

To determine the driving event, the context parameters need to be fusioned. Multimodal fusion is the process of combining two or more signals or parameters obtained from two or more sensors (or other sources such as IoT) using two or more modalities. Once fusioned, the driving event is interpreted at the higher-level of abstraction.

Machine learning training sets (i.e. supervised learning) are created so that the driving event in consideration can be easily predicted. The fusioned information is searched/matched on the knowledge database to determine if a pattern to the new situation is found. If so, then such situation has a meaning to the system and a corresponding action needs to be implemented.

Consider, for example, the fusion of the following data. It represents a typical driving situation. The notation of the rules in the fusion of context parameters below is that of SWRL (Semantic Web Rule Language) (Subercaze Julien , W3C). Note that the symbol '?' below represents an instance or individuals of classes in our ontology:

- Let vehicle X be an individual of class "*Vehicle*" == *Vehicle (?X)*
- Let Y be the individual of class '*Road*' == *Road(?Y)*
- Let vehicle X be on Road Y == *isOnTheRoad(?X, ?Y)*
- Let Z an individual of class '*Lane*' on Road Y == *hasLane(?Y, ?Z)*
- Let vehicle X be currently located on lane Y == *isOnLane(?X, ?Z)*
- Let W be an individual of class Object be located on lane Z == *hasObject(?Z, ?W)*
- Let the distance of object W (after calculation) from vehicle X be near == *hasDistanceFromVehicle(?, nearDistance)*

If we fusion the parameters and using our SWRL rules, we will end up concluding that object W is an obstacle relative to vehicle X. That is, logically:

$$Vehicle(?X) \wedge Road\ (?Y) \wedge isOnTheRoad(?X, ?Y) \wedge hasLane(?Y, ?Z) \wedge isOnTheLane(?X, ?Z) \wedge hasObject(?Z, ?W) \wedge hasDistanceFromVehicle(?W, nearDistance) \rightarrow Obstacle\ (?W)$$

Multimodal fission is the necessary action to the given driving situation. Given that we have an obstacle, the fission yields an action corresponding to avoiding the obstacle. The action itself is composed of sub-actions. For example, there are various steps to avoid an obstacle.

3.6 Assistance for the Driver and the Vehicle

The actions for different driving situations are implemented as ontology. The ontological class "*Action*" is the overall class that will be instantiated depending on the driving situation at hand. The class "*Action*" has two sub-classes, *DriverAction* and *VehicleAction* directed towards the driver and vehicle, respectively. In the *VehicleAction*, we see subclasses: *Call_Emergency*, *Turn_Light_On* and *Adjust_Airconditioner*. For the class *DriverAction*, the subclasses are as follows:

- *ChangeLane*: is an action to change lane to avoid an obstacle located on the same lane and that there is a free lane where the vehicle can go.
- *RemainInTheSameLane*: invoked when there is an obstacle but there is no free lane available where the vehicle can go.
- *Brake*: invoked when braking is necessary. This class has the following sub-classes, describing the situation when to brake: (1) *BrakeForPedestrian* – here, the pedestrian is an obstacle and the vehicle needs to brake to allow the pedestrian to cross the street; (2) *BrakeForObstacle* – this is invoked when the obstacle is on the same lane that is not moving or when the obstacle is a another vehicle whose speed is lower than the speed of the vehicle; (3) *BrakeForRedLight* – this is invoked when the system detects that there is a '*TrafficLight*' object in the scenario and its value is '*Red*', or '*Yellow*' and the distance of the vehicle and *TrafficLight* object is near (the colour will shift from yellow to red in a short while)
- *SlowDown*: this is invoked when the driver is overspeeding
- *HaveABreak*: invoked when the driver is tired, stressed or going to have malaise.
- *BadWeather*: informs the driver that weather has changed due to the presence of rain, snow or fog on the road.
- *AdvanceCarefully*: informs the driver to continue driving slowly.
- *ReleaseAccelerator*: this is related to slowing down as it is related to demanding driver to release foot from accelerator to reduce gas consumption (i.e. applicable when driving from a road segment with higher speed limit to a road segment of 0lower speed limit). This is part of our work related to the green driving.

3.7 Machine Learning

Machine Learning algorithms (Mitchell 1997) use data to discover pattern and can be used to predict an output from a formatted input after training the algorithm on a sufficiently big set of training data (Tchankue, Weeson et al. 2013). To begin with, a driving trip from point A to point B is a repetition of basic driving events: *go straight*, *turn left*, *turn right*, *stop*, *slow down*, *go roundabout*, and *avoid obstacle* (where obstacle may be a moving object – a vehicle, a pedestrian, a bicycle – or a non-moving object). In this work, we make use of supervised learning to classify common driving situations (e.g. stop, turn to the left, turn to the right, over speeding, etc.) and unsupervised learning to classify driving situations that are the combinations of two or more common driving situations (e.g. driver to turn to the left and pedestrian is crossing the street). When a driving event is detected, an assistance action may be invoked. In such a case, we make use of optimization and reinforcement learning. The intent is to do assistance for the driving situation and integrate these aspects into the action: *safe*, *green* and *comfortable driving*.

4 RESULTS AND DISCUSSION

4.1 Methodology

We created a driving simulator using Unity 3D software (Engine 2016) to mimic driving scenarios in the laboratory. As one drives, the data from the driving event are sent to the ontology template as a JSON data. These data instantiate various classes and objects in the ontology. The result is an actual driving event.

The ontology becomes an actual description of the driving event. The event needs to be classified so it is fed as input to the machine learning classifier. The process yields an identification of a driving event. For now, the driving events are basic ones: stop, turn left, turn right, and normal driving. These are representative samples of all other driving events. Why only a limited driving events? The logic is simple: *if this works fine with the representative samples, then it will work fine in all other driving events*. Using driving rules in SWRL, we can identify if there is an infraction committed or if there is a need for driving assistance intervention. If such is the case, a driving assistance message will be sent to the driver. If there is a need to intervene for the vehicle (e.g. turn on the fog light), a signal will be sent to the vehicle. This signal will be used to implement an action for the vehicle. The driver then continues driving as he wishes.

4.2 Simulation, Data Analysis and Processing

In the simulation experiments, the aim is to detect the driving event. To do so, parameters that describe the context of the vehicle are needed. We collected all data that are near our vehicle (parameters for the context of the environment). We also pre-defined the context of our driver (i.e. we are the driver). These objects need to be identified for the purpose of classification using machine-learning algorithms. We record road objects (e.g. pedestrian, traffic signs, other vehicles) detected by the car. The class "*RoadObject*" is a template for this purpose. A RoadObject has the following properties: (i) type (e.g. stop, pedestrian, vehicle, etc.), (ii) speed, (iii) acceleration (iv) distance (i.e. distance of this object from us), (v) position, (vi) lane and (vi) weight. The RoadObjects are stored in the JSON format.

To collect data, multiple test drives were conducted for each common driving event. As it is to be used as training set for machine learning algorithms, we collected 1699 driving state; the repartition of each driving event is shown in Table 1 This is a representative sample of the basic and preliminary driving events conducted in the lab.

Table 1: Representative sample driving events.

Driving event	Number of event	Percentage of event
Normal	1138	66.98%
Stop	257	15.12%
Turn left	162	9.53%
Turn right	142	8.35%

Machine learning algorithms require formatted data (generally numeric) and a fixed number of columns. For instance, we used one hot encoding for the categorical data, like the position on lane. The last step in data processing is tagging the data. We used the simple tool in Python, the IDE Jupyter Notebook.

4.3 Decision Tree and K-Nearest Neighbor

Decision tree learning uses a decision tree as a predictive model. A decision tree is a flowchart-like structure in which each internal node represent a "test" on an attribute, each branch representing the outcome of the test while each leaf representing a class label for classification tree. A tree can be created by splitting the training set into subset based on an attribute value test and repeating the process until each leaf of the tree contains a single class label or we reach the desired maximum depth. There are multiple criterion that can be used to divide a node into two branch, such as the information gain which consist of finding the split that would give the biggest information gain, based on the entropy from the information theory (Witten, Frank et al. 2011).

The k-nearest neighbor algorithm, on the other hand, is a simple algorithm which consists of selecting for an instance of data the k-nearest other instances and assigning to the first instance the most frequent label in the k instance selected. The value of k is user defined. The distance can be computed in different ways, such as the Euclidian distance for continuous variables like ours. The importance of each neighbor can be weighted; often the weight used is inversely proportional to the distance to give more importance to closer neighbor.

4.4 K-Fold Cross Validation and Machine Learning Classification Results

Validation measures how a predictive model will perform after generalization on an independent data set. It is useful to know the viability of a model and to avoid overfitting problem, i.e. fitting the model too closely to the training set, after that, it gets very good results on it bugs will perform poorly on other data set. The cross validation is a technique used to validate a predictive model. It consist of splitting the original data set into a validation set and a training set, training our model on the training set and validating it on the validation set. We repeat the operation multiple times with different splits and take the average of the validation results to reduce variability. The k-fold cross validation consists of splitting the original data into k equal size sub-samples. We then retain a sub-sample as the validation set and use the *k-1* other sub-samples to train the model. The process is repeated *k* times, with each of the k sub-sample used one time as the validation data. Cross validation is commonly used when the number of data is small and the usual splitting into a training set and a validation set would not be convenient. To validate a classification problem like ours, we use our model to predict a number of samples, and we then compare the number of sample correctly tagged to the number of sample incorrectly tagged, giving us the percentage of correctly tagged data. Furthermore, confusion matrix can be used to know where a classification algorithm behaves poorly.

After processing and tagging our data, experiments with machine learning algorithms were made using scikit-learn library. We experimented with decision tree and k-nearest-neighbor algorithms. Results are validated using cross-validation with 10 folds, meaning that we have divided the dataset into 10 equal parts, and each time we trained it on 9 parts and tested it on the tenth. The results are an average precision of *95%* for decision tree and a precision of *93%* for the k-nearest neighbour algorithm. The results indicate good results, although the number of sample is quite low. We intend to test further with other parameters for each algorithm to get better results; we will also collect more data in different situations.

4.5 Human-vehicle Interaction Interface

This work is in evolution and available data is based on laboratory experiments results. The HCI interface shows messages intended for the driver. This confirms that our machine learning classification system is correct and that the SWRL driving rules are correctly applicable to the detected driving events. Our driving assistance system classifies messages as one of the following: (1) *Notification* – a message to inform the driver, and (2) *Alert* – this is a type of message that attempts to get the driver's attention. A notification or alert is sent according to the category of driving situation:

- *Behaviour* – this refers to the driver's conduct of driving. Example: over speeding;
- *Danger* – a potential risk to the driver or people on the road exists; the driver's behavior has nothing to do with this. Example: Poor visibility due to fog; and
- *Ability* – this concerns about the person's ability to drive. Example, the driver is falling asleep.

Given that two or more messages cannot be sent to the driver at the same time, a priority scheme is implemented, as follows:

- Alert has a higher priority than Notification;
- Ability has the highest priority, followed by Danger and lastly by Behaviour.

5 CONCLUSION

In this paper, we have demonstrated our work on intelligent transportation. This paper shows the current status of our work on the ontological and machine learning approach for managing driving context. In the paper, we present the components of a driving context using ontology, starting from the context of the driver all the way to the context of the environment. The driving context template is generic such that all kinds of driving situations on the road can be represented. We designed our own driving scenario simulator and modeling various events but sampling on the basic ones: turn left, turn right, stop, etc. By simulation, we are able to instantiate objects using real values. We use machine learning to classify driving events. As the results show, event classification using decision tree yields 95% detection rate accuracy. More machine learning tests and collection of sample training data are on the agenda. Deep reinforcement learning (Phan, Dou et al. 2015, Phan, Dou et al. 2017) will be invoked once we are to perform the driving assistance actions for some driving situations. Future works include designing and implementing a cognitive user interface component.

REFERENCES

An, S.-h., B.-H. Lee and D.-R. Shin (2011). A Survey of Intelligent Transportation Systems. *3rd Intl Conference on Computational Intelligence, Communication Systems & Networks (CICSyN)*. Bali, Ind. pp. 332-337.

Ashton, K. (2009). "*That 'Internet of Things' Thing*." from http://www.rfidjournal.com/article/print/4986.

Atzori, L., A. Iera and G. Morabito (2010). "The internet of things: A survey." *Computer Networks* 54(15): 2787–2805.

Bin, S., L. Yuan and W. Xiaoyi (2010). Research on data mining models for the internet of things. *Intl. Conf. on Image Analysis and Signal Processing*. pp. 127–132.

Brioschi, G., Hina, M. D., Soukane, A., Ramdane-Cherif, A. and Colombetti, M. (2016). Techniques for Cognition of Driving Context for Safe Driving Application. *ICCI*CC 2016, 15th IEEE Intl Conf. on Cognitive Informatics and Cognitive Computing*, Stanford, CA.

Dice. (2017). "Ontologies." from https://www.dice.com/skills/Ontologies.html.

Game Engine, G. (2016). "Unity 3D." from https://unity3d.com/.

Gubbi, J., R. Buyya, S. Marusic and M. Palaniswami (2013). "Internet of Things: A vision, architectural elements and future directions." *Elsevier Future Generation Computer Systems* 29(1): 1645-1660.

Miorandi, D., S. Sicari, F. D. Pellegrini and I. Chlamtac (2012). "Internet of things: Vision, applications and research challenges." *Ad Hoc Networks* 10(7): 1497–1516.

Mitchell, T. (1997). *Machine Learning*, McGraw Hill.

Naja, R. (2013). A Survey of Communications for Intelligent Transportation Systems. *Wireless Vehicular Networks for Car Collision Avoidance*, Springer NY.

Obrst, L. (2003). Ontologies for Semantically Interoperable Systems. *12th International Conference on Information and Knowledge Management*, New Orleans, LA, USA, ACM Press, New York, USA.

Oracle. (2009). "*Architectural Strategies for Cloud Computing*." from https://www.vanharen.net/Player/eKnowledge/architectural_strategies_for_cloud_computing.pdf.

World Health Organization WHO, (2015). "*10 Facts on Global Road Safety*." from http://www.who.int/features/factfiles/roadsafety/en/.

Phan, N., D. Dou, H. Wang, D. Kil and B. Piniewski (2015). Ontology-based Deep Learning for Human Behavior Prediction in Health Social Networks. *2015 ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. Atlanta, GA, USA.

Phan, N., D. Dou, H. Wang, D. Kil and B. Piniewski (2017). "Ontology-based deep learning for human behavior prediction with explanations in health social networks." *Information Sciences* 384: 298-313.

Subercaze Julien, P. M. Semantic Agent Model for SWRL Rule-based Agents. *International Conference on Agents and Artificial Intelligence (ICAART 2010*), Valencia, Spain, INSTICC Press.

Tchankue, P., J. Weeson and D. Vogts (2013). Using machine learning to predict the driving context whist driving. *SAICSIT '13 South African Institute for Computer Scientists and Information Technologists Conference*, East London, South Africa

Tsai, C.-W., C.-F. Lai, M.-C. Chiang and L. T. Yang (2014). "Data Mining for Internet of Things: A Survey." *IEEE Communications Surveys & Tutorials* 16(1).

VOWL. "Visual Notation for OWL Ontologies." from http://vowl.visualdataweb.org/v2/.

W3C. "*SWRL: A Semantic Web Rule Language Combining OWL and RuleML*." from https://www.w3.org/Submission/SWRL/.

Witten, I., E. Frank and M. Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington, MA, Morgan Kaufmann.

Yousif, M. (2017). "The State of the Cloud." *IEEE Cloud Computing* 4(1): 4 - 5.

Zhao, L., R. Ichse, S. Mita and Y. Sasaki (2015). Ontologies for Driver Assistance Systems. *35th Semantic Web and Ontology Workshop*: 1 - 6.