# Toward Requirements Specification for Machine-Learned Components

Mona Rahimi
Dept. of Computer Science
University of Toronto
Toronto, Canada
mrahimi@cs.toronto.edu

Jin L.C. Guo
School of Computer Science
McGill University
Montreal, Canada
jguo@cs.mcgill.ca

Sahar Kokaly
Dept. of Computer Science
University of Toronto
Toronto, Canada
skokaly@cs.toronto.edu

Marsha Chechik
Dept. of Computer Science
University of Toronto
Toronto, Canada
chechik@cs.toronto.edu

***Abstract*—In current practice, the behavior of *Machine-Learned Components* (MLCs) is not sufficiently specified by the predefined requirements. Instead, they "learn" existing patterns from the available training data, and make predictions for unseen data when deployed. On the surface, their ability to extract patterns and to behave accordingly is specifically useful for hard-to-specify concepts in certain safety critical domains (e.g., the definition of a pedestrian in a pedestrian detection component in a vehicle). However, the lack of requirements specifications on their behaviors makes further software engineering tasks challenging for such components. This is especially concerning for tasks such as safety assessment and assurance.**

**In this position paper, we call for more attention from the requirements engineering community on supporting the specification of requirements for MLCs in safety critical domains. Towards that end, we propose an approach to improve the process of requirements specification in which an MLC is developed and operates by explicitly specifying domain-related concepts. Our approach extracts a universally accepted benchmark for hard-to-specify concepts (e.g., "pedestrian") and can be used to identify gaps in the associated dataset and the constructed machine-learned model.**

## I. Introduction

In the automotive industry, a variety of Machine Learning (ML) techniques are used to build software components in Advanced Driving Assistance Systems (ADAS). Unlike the development for software components whose behaviors are deterministic, in current practice, the behavior of *machine-learned components* (MLC) is not fully specified according to a set of predefined requirements. Instead, such components learn their specifications from a collected set of training-validation-testing data and generalize their learning to unseen data inputs during operation. Such capacities make them a desirable and preferred approach for advanced functionalities (e.g., pedestrian detection), especially when the lack of specifications makes it hard to direct programmers to perform a deterministic implementation.

One major challenge in ADAS is the development of a reliable pedestrian detection system due to the varying and hard-to-predict appearance of pedestrians (e.g., different clothes, size and shapes) [10]. Thus, it is extremely difficult to specify unambiguous requirements for a pedestrian detector component. One potential mitigation is to determine a set of domain-specific features and assumptions for a potential pedestrian that are as complete as possible. For example, positive answers to questions such as *"Is a person on a bicycle, walking with a bicycle, carrying a bicycle, on a scooter or roller skates, considered a pedestrian?"* ensure that the pedestrian detector will be able to detect and classify people carrying a bike or wearing roller skates as pedestrians.

**Position and contributions:** The challenges posed by developing MLCs demonstrate a compelling need for the requirements engineering community to develop techniques for analyzing and engineering requirement specifications for MLCs. These specifications will later improve the process of verification and validation of MLCs and assist in understanding the underspecified concepts in both the dataset and machine learning models.

As the first step of requirements specification for MLCs, our focus in this paper is on identifying domain-specific concepts during various stages of MLC development and deployment. We propose a high level view of an approach which creates a web-based benchmark for domain specifications and uses it to identify gaps in the dataset and the model specifications. Throughout the paper, we use a pedestrian detector component from the automotive domain as an illustrative example.

The remainder of the paper is structured as follows. Section II summarizes the work related to requirements specification for MLCs, and discusses the prominent challenges in the process of development and application of MLCs. Section III describes the major components in our proposed solution to facilitate the specification of requirements for MLCs. Section IV presents several applications of our proposed approach in the safety critical domain. Finally, Section V shares insights gained through our investigation and concludes.

## II. Background

In this section, we summarize the efforts made by different communities to specify MLCs and highlight the compelling need for requirement engineers and researchers to contribute to this landscape.

To define the behavior of MLCs as a whole with respect to how they address the target applications, **component-level specification** is often used. For example, the requirement for the automated pedestrian collision avoidance system might specify that "*the position of the pedestrian should be detected*

*within an accuracy of 0.5m*". However, decomposing such high level specifications to lower-level verifiable ones is difficult, if not impossible [16]. This often leads to insufficiencies in machine learned functions [6]. One specific challenge is the lack of specifications for the *domain-specific concepts*. In the previous example, the concept "pedestrian" is not defined concretely in terms of what it refers to, and in which environment. Therefore, it is unclear what is the implication of the high-level specification to the downstream MLCs development tasks, such as data collection and model selection.

One key ingredient of MLCs is the dataset used to train the models. Therefore, dataset management is critical to the overall quality of systems with MLCs. For **dataset specifications**, in particular, Marc D. Kohli et. al. suggested best practices in the medical domain. The authors proposed that the specifications should be clearly defined for medical image data, covering all aspects of image management, including image metadata, pixel data, post processing techniques, image cataloging, etc. [13]. On the other hand, as the major enablers in MLCs that perform learning and prediction, the machine learning models need to be implemented correctly according to the **model specifications**. Based on the particular learning algorithms, model specifications normally define how the theoretical properties should hold during the implementation [15].

The MLC development process continues with training the machine learning models using the input data. This process often requires multiple steps to ensure an optimal performance. To enable a consistent and assured training result, an **MLC development process specification** is also essential. To this end, R. Salay and K. Czarnecki [14] proposed that the training procedure be clearly specified to meet the functional safety standard. For example, an analysis on the adequacy of regularization (one specific technique in machine learning) shall be carried out to mitigate the flaw of over-fitting. Among the process requirements for other phases of the software development, this work delineates a relatively complete set of instructions to realize the safety assurance of software with MLCs for safety critical domains.

For verification purposes, it is pivotal to construct the **traceable path** to demonstrate the compliance of source code with the design specification and coding guidelines [14]. The traceable path can support building the safety case to demonstrate that the identified hazards are sufficiently mitigated. F. Falcini et. al. also called for building the infrastructure to support traceability in automotive software when integrating a "V" model for data development with the standard "V" model for software development, calling the result a "W" model [8]. Recent work by A. Banks and R. Ashmore has demonstrated the potential of traceability by implementing and maintaining high level software requirements through building confidence in training data [3]. Such confidence includes nine items including data sufficiency, self-consistency and absence of bias. However, the specific steps to achieve this confidence in the training data are yet to be defined. For example, for the automated pedestrian collision avoidance system, what are the specific criteria of *sufficiency* for managing the dataset to recognize the concept of a "pedestrian"? As we described earlier, the difficulty starts with our very limited understanding of how this concept should be defined even at the high level, and how it should be represented in the training data.

To summarize, while the effort for improving the MLC specifications has been emerging for different aspects during the MLCs development and deployment, one of the major barriers is the impotence to systematically reason and specify the semantics of domain-specific concept, and to trace them through various stages of the MLCs life cycle. Accordingly, our work sets off to tackle the ambiguity of domain concept semantics in requirements for MLCs, and to support the identification of gaps in the high level specification, data instances, and machine learning models.

## III. Proposed Solution

In this section, we describe our proposed solution that enhances the process of requirements specification for a MLC. Figure 1 gives an overview of its four major phases.

### A. Benchmarking the Domain

Our approach first creates a benchmark ontology for the domain concepts. This phase contains the following steps of knowledge acquisition and conceptualization [4]:

*1) Scoping the Domain:* To specify a domain, a generic set of characteristics, which matter in the perception of the concept, need to be identified. For example, to specify the concept of *pedestrian*, the first step is to discover what people perceive to be important features for a pedestrian. These features later serve as key terms in the requirements specifications for a pedestrian detector component. To provide preliminary results, we perform a search query ("pedestrian") using the existing web search engines (e.g., Google, Yahoo, etc.) to retrieve a set of relevant documents. For this purpose, we utilize Watson [11], a tool previously developed to augment trace queries. Relevant terms are selected according to the cosine similarity between a query and web-based documents as well as the relative frequency of the term within the domain-specific documents. Figure 2 represents an initial set of relevant terms such as *clothing*, *speed* and *scooter* which are returned for the search query *"pedestrian"*. The term list can be manually refined and extended until a desired terminating condition, such as a number of iterations, is met.

*2) Visualizing the Domain:* In order to visualize the web-based extracted knowledge on domain concepts, we manually created a domain ontology using relevant terms from the previous step. A partial snapshot of a constructed ontology for *pedestrian* is illustrated in Figure 2, where we manually classified web-extracted terms such as *scooter*, *bike*, *wheelchair*, as a ***device***; *side of the road*, *crosswalk*, and *footpath* as a ***theme***; and *clothing*, *safety*, *speed*, as an ***attribute*** for a *pedestrian*. In order to fill the gap between knowledge extraction and data modeling, the domain ontology should be reviewed by domain experts and stakeholders for completeness and correctness. For example, a review would verify if a *pedestrian on bike* should be classified and treated the same as a pedestrian or differently.

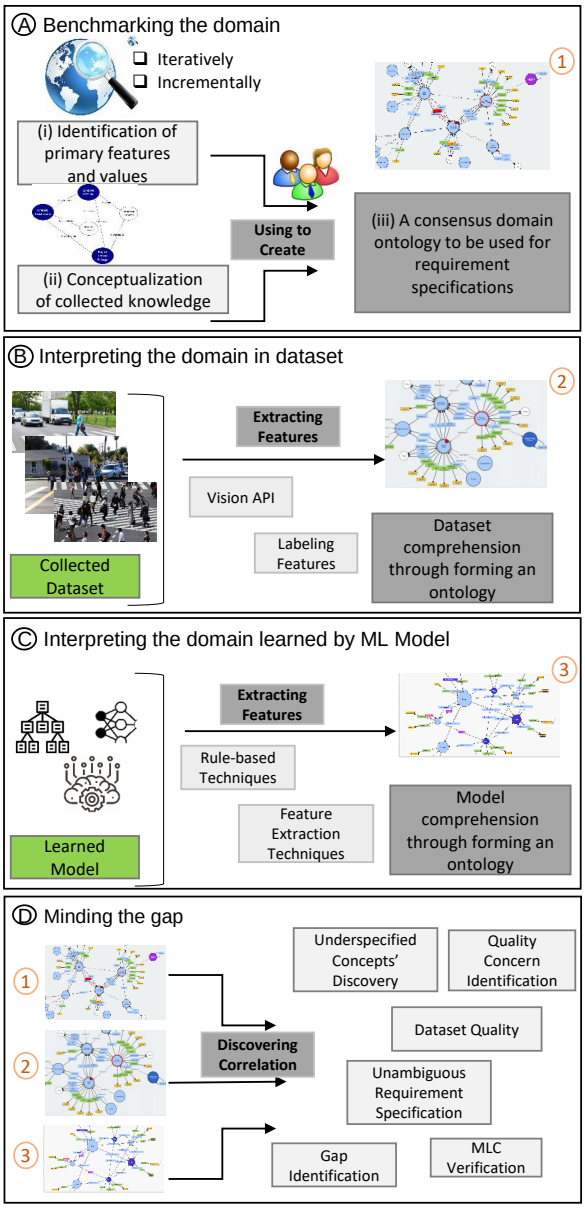


Fig. 1. High-level overview of proposed approach.

### B. Interpreting the Domain in Dataset

In the second phase, we identify the domain specifications through analyzing the collected dataset, with the goal of comprehending the concept demonstrated by the data. Analysis on the inconsistencies between what the dataset represents and the specifications produced in the previous step can reveal the potential weaknesses in both the dataset and the produced ontology. As preliminary results, we extracted features from the Caltech dataset, a large scale pedestrian dataset for training and evaluating pedestrian detection algorithms [7]. We used the Google vision API, which provides an image analysis service based on machine learning [1]. We retrieved a set of labeled features from the Caltech pedestrian images such as *street*, *road*, *stroller*, etc. By looking at the partial ontology illustrated in Figure 2, we observe a potential correlation between an image feature *stroller* with the ontology element *device*, and a similarly between *street* and *road* with *theme*. Incorrect and missing correlations for an existing ontology element can potentially represent weaknesses in the dataset.

### C. Interpreting the Domain Learned by ML Model

Models, depending on the machine learning techniques employed, can be described in diverse ways [18]. While interpretability (i.e., comprehensibility) of machine learning models remains an open challenge, some models are considered more interpretable (e.g., those created by rule-based techniques) than others (e.g., built by neural network techniques). Clearly, the comprehensibility of a model depends, strongly and subjectively, on the actual *contents*. Identifying the most relevant and important attributes in a model helps understand its content. For example, attributes which are identified in higher levels of a decision tree, attribute-value conditions in classification rules, and features with larger weights in neural networks can further help extract a *concept* from a model [9]. To this end, part of the literature focuses on feature extraction techniques as a means to select salient attributes of a model [19], [17]. Others focus on extracting more comprehensible models (e.g., rule sets or decision trees) from black box models, produced by methods such as neural networks [2], [5]. Another work provides a useful classification of existing approaches for interpreting black box models [12]. We believe that such techniques can be used to extract the most important and relevant features from a model and further help specifying the domain concept which the model has learned. The concrete assessment of their effectiveness is left for future work.

### D. Minding the Gap

Despite using different terminologies, and derived from different origins (document vs. image) and techniques (web-based extraction vs. feature extraction), the identified domain concepts in each step can be correlated to identify the gaps among the constructed ontology, dataset representations, and machine learning models. For example, the feature *stroller* presented in the dataset is an entity under the ontology element *device*. To discover relations between the extracted terms, we can use either semantic-based methods (e.g., probabilistic topic models) or structure-based methods (e.g., word correlation). Correlating features reveals the gap between the public specification of the domain and features described by the dataset and model. For example, not finding any feature related to an existing *device*, such as a *wheelchair*, in the domain ontology, can be representative of such a potential gap.

## IV. Applications

The conceptualization and visualization of domain concepts provide support for a number of software engineering activities for safety-critical systems. We describe some of them below.

**Requirement Specifications:** Our approach facilitates the specification of unambiguous requirements for MLCs by creating a web-based benchmark for hard-to-specify domain concepts. For example, using our approach, one requirement

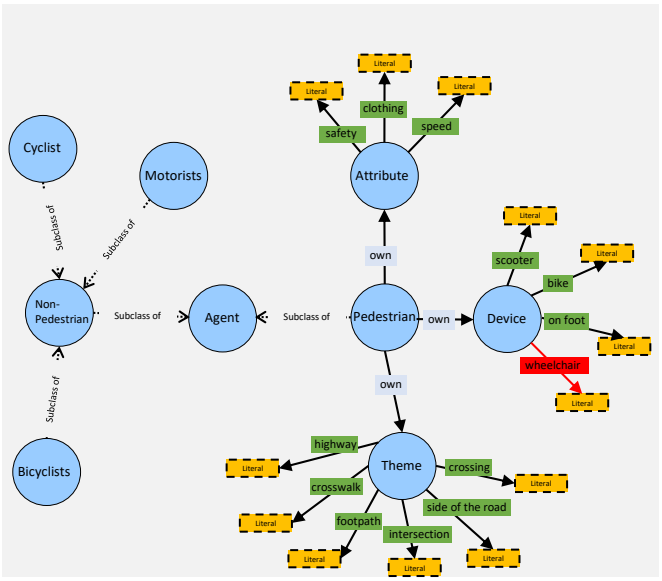


Fig. 2. A partial snapshot of an ontology for concept *pedestrian* constructed by automatically extracting web-based relevant terms.

for a pedestrian detector component should be specified as: "The pedestrian detector component shall be able to detect pedestrians *on foot*, on a *scooter* and on a *wheelchair*" (these were identified in Section III-A2 as an accompanying ***device*** for a *pedestrian*).

**Verification:** Using our approach for creating a set of requirements specifications, we can perform a systematic verification of an MLC against this set. For example, we can verify whether the component is able to correctly classify a pedestrian on a *wheelchair* or a *scooter*.

**Identification of Quality Concerns:** Our approach extracts quality-related terms such as *pedestrian safety* for the Caltech dataset [7]. This extraction emphasizes *safety* as a major quality concern in the domain and forces asking questions such as "to what extent is the identified quality concern addressed in the requirements specification of an MLC?" Answering these leads to verifying the relative completeness of the existing set of requirements specifications and encourages consequent corrections if necessary.

**Dataset Quality:** Our approach can help understand whether a dataset correctly represents what it is expected to with respect to a benchmark. For example, we can answer questions such as "does this dataset identify people on bikes as pedestrians?" and depending on our goal, an analyst may choose to augment the dataset.

**Identification of Underspecified Concepts:** Upon comparing the model or dataset ontology with an extracted benchmark and not finding any feature related to a term (e.g., a *wheelchair*, as shown in red in Figure 2), an analyst can readily see that a concept (e.g., a *pedestrian on a wheelchair*) is underspecified in the dataset or is not properly learned by the model, allowing them to augment the dataset.

## V. Conclusion

In this paper, we have emphasized the importance of requirements specification for MLCs. Through a literature review, we identified where the requirements specifications play a role in the development and operation of an MLC. To facilitate specification of requirements for MLCs, we proposed a potential solution for extracting and visualizing domain specifications. Our web-based approach can be further used to identify gaps between a dataset, a model and what the general public thinks a concept (e.g., "a pedestrian") to be. Our next steps are to fully implement the proposed approach, specifically, the parts related to interpreting the learned domain (Section III-C) and identifying the gap (Section III-D), and validate its effectiveness. We plan to automate the process of ontology creation which so far has been done manually, in order to increase the usability of the approach.

## References


[1] Google Vision API description. https://cloud.google.com/vision/. Accessed: 2019-02-30.

[2] M. G. Augasta and T. Kathirvalavakumar. Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150, 2012.

[3] A. Banks and R. Ashmore. Requirements assurance in machine learning. In *Workshop on Artificial Intelligence Safety*, 2019.

[4] M. R. Bautista-Zambrana. Creating corpus-based ontologies: a proposal for preparatory work. *Procedia-Social and Behavioral Sciences*, 212:159–165, 2015.

[5] O. Boz. Extracting decision trees from trained neural networks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 456–461. ACM, 2002.

[6] S. Burton, L. Gauerhof, and C. Heinzemann. Making the case for safety of machine learning in highly automated driving. In *International Conference on Computer Safety, Reliability, and Security*, pages 5–16. Springer, 2017.

[7] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2011.

[8] F. Falcini, G. Lami, and A. M. Costanza. Deep learning in automotive software. *IEEE Software*, 34(3):56–63, 2017.

[9] A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.

[10] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1239–1258, 2009.

[11] M. Gibiec, A. Czauderna, and J. Cleland-Huang. Towards mining replacement queries for hard-to-retrieve traces. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 245–254, New York, NY, USA, 2010. ACM.

[12] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93, 2018.

[13] M. D. Kohli, R. M. Summers, and J. R. Geis. Medical image data and datasets in the era of machine learning. *Journal of Digital Imaging*, 30(4):392–399, 2017.

[14] R. Salay and K. Czarnecki. Using machine learning safely in automotive software: An assessment and adaption of software process requirements in ISO 26262. *arXiv preprint arXiv:1808.01614*, 2018.

[15] S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, and X. Yue. Formal specification for deep neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 20–34. Springer, 2018.

[16] S. A. Seshia, D. Sadigh, and S. S. Sastry. Towards verified artificial intelligence. *arXiv preprint arXiv:1606.08514*, 2016.

[17] R. Setiono and H. Liu. Neural-network feature selector. *IEEE Transactions on Neural Networks*, 8(3):654–662, 1997.

[18] A. Vellido, J. D. Martín-Guerrero, and P. J. G. Lisboa. Making machine learning models interpretable. In *20th European Symposium on Artificial Neural Networks, ESANN 2012*, volume 12, pages 163–172, 2012.

[19] A. Verikas and M. Bacauskiene. Feature selection with neural networks. *Pattern Recognition Letters*, 23(11):1323–1335, 2002.