MONITORING OPAQUE LEARNING SYSTEMS

Leilani H. Gilpin
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
lgilpin@mit.edu

ABSTRACT

Machine learning models and their underlying debugging methods are specific and not applicable to other data sets and models without much effort. The goal of this work is to improve machine learning robustness with an adaptable monitoring framework for identifying and explaining anomalous output that can be easily customized for different domains with a common vocabulary and rule language. The input to the monitoring system is the output of an opaque learning system, which is parsed into a common language. The monitor then uses a reasoner to precisely find the important concepts leading to contradictions between expected behavior and anomalous behavior, and tracks that anomalous behavior back to a constraint or rule. The output of the system is a human-readable explanation succinctly describing the core reasons and support for an intended behavior.

1 INTRODUCTION

When machine learning algorithms make mistakes, we would like to know why the error occurred. Explanatory systems that provide the core reasons and premises for an intended behavior can answer this question. But many of these systems require access and knowledge about the underlying learning mechanism, which may be implausible (Marcus, 2018). Further, the learning algorithms could be opaque or proprietary, so we want to create a way to *monitor* their output, with limited knowledge of underlying system. In addition, we want these algorithms to be able to support their decision or output with an explanation or reason. The ability to provide coherent explanations of complex behavior is important in the design and debugging of such systems, and it is essential to give us all confidence in the competence and integrity of machine learning algorithms and aids.

With the need for malleable, self-explaining systems, this work examines a new self-monitoring framework that can impose constraints of reasonableness for the output of machine learning systems in multiple domains. With the input of a domain-specific knowledge base and rules, it uses a reasoner, data parser with a ontology, and explainer to judge and explain whether the input is reasonable or not. This paper demonstrates how this technology can be used to detect and explain anomalous output after the fact, and also motivate its use as a real-time monitoring system. We also introduce the use of explanations as a type of feedback to *learn* the rules and reasons, and make better judgments in future iterations.

2 RELATED WORK

Monitoring for reasonability is an open topic in computer science. Collins & Michalski (1989) present a formal ontology for reasonability, but it lacks a structural implementation. Cohen & Levesque (1988) present a logical theory of reasonable actions by representing "language as action." Their method defines a set of reasonable action expressions from logic. Although formal approaches are provably correct, they do not lend themselves well to an implementation. Many have tried to make ontologies and generalizations of these kinds of judgments, but they remain specific to the machine specifications (Abellan-Nebot & Subirón, 2010) or software specs.

Combining knowledge bases and web standards has been implemented as ConceptRDF; a conversion of ConceptNet to RDF/XML format (Najmi et al., 2016), although the rules are not applied

to working system. ConceptNet and rules have been combined for emotion recognition (Shaheen et al., 2014), but this work is a combination of rules and commonsense for detecting and explaining reasonableness.

Reasoning systems have been developed to keep track of consistencies. This work is focused on making a generic monitor using semantic web technologies with RDF to represent logs and AMORD In RDF (AIR) (Khandelwal et al., 2010) as a rule language that supports AMORD-like constructs Kleer et al. (1978). AMORD is a rule-based problem solver that keeps track of dependencies, similar to truth maintenance systems (De Kleer, 1986).

AIR describes properties that can be used to define policies and a reasoner to provide reasons and explanations. RDF [1] is an acronym for the Resource Description Framework, which is a World Wide Web Consortium (W3C) standard. W3C is the main international standards organization for the World Wide Web. While RDF is used mainly for managing and representing data in web frameworks and resources, it is also used in a variety of knowledge engineering tasks due to its triple-store format. This format is natural for representing language (as subject-predicate-object) and for representing premises.

3 IMPLEMENTATION

The monitoring system can provide accurate judgments of reasonableness and convincing explanations of reasonableness by applying the system to two use cases: first descriptions of perception (which could be generated from a machine learning scene understanding systems), and secondly vehicle plans (from an autonomous vehicle planning system, which could be proprietary).

The monitoring system receives a natural language description as input. The monitor takes this input, parses it into a log file, which is compared with rules to produce the reasons and premises leading to an explanation. The output of the monitoring system is a judgment and an explanation of whether the input is reasonable or not. The monitoring system should satisfy the following two properties.

1. The monitor can detect clearly unreasonable input.
2. The explanations generated from the monitoring system should be "useful." Although current efforts are focused on ways to use these explanations for feedback to make better decisions, for the purposes of this work, I ask whether a human *user* finds the explanations to be convincing.

For this work, reasonableness is defined as abiding to a set of commonsense rules. In the case of vehicle planning, these are the rules of the road. For the perception example, these are commonsense rules for the actor and object of the input description. These commonsense decompositions are detailed in previous work (Gilpin et al., 2018), in which the rules and representations are specific to the use case of an opaque "scene understander" annotating images with text descriptions.

3.1 LOG GENERATION AND ONTOLOGY

In order for the monitor to be generic, it's required that the log, or data, is constructed in the RDF, which is a World Wide Web Consortium (W3C) standard[2]. As mentioned in Section 2 RDF allows for data descriptions and relationships between data in terms of triples. The RDF log contains the system state in terms of symbolic triple relations. An example RDF log is in Figure 2 in the Appendix. It contains the subject, predicate and object of the input description, and relevant descriptions aggregated from the commonsense database. This aggregation utilizes RDFS (the RDF Schema), a semantic extension of RDF, allowing for additional mechanisms for describing groups of related resources and the relationships between these resources.

For perception, I generate the RDF log for a description by parsing for relevant concepts. From the input of a natural language description, I use a regex parser in python to extract the noun phrase, verb phrase, context information (prepositional phrases) to identify the actor, object, and action of

[1] `https://www.w3.org/RDF/`

[2] `https://www.w3.org/TR/rdf-schema/`

the description. The development of an ontology is incorporated with the process of developing the log data. For this perception description use case, I develop a set of anchor points to extract commonsense knowledge from ConceptNet, and primitive actions represented as a conceptual dependency primitives. This conceptual dependency primitive will be used to construct rules, with the actor, object, and context information as input. An example of a parsed description represented as an RDF log is shown in the Appendix in Figure 2.

For vehicle planning, the process is extended with the same parsing process, the representation is extended to include vehicle primitive actions like yield, move (with speed and direction), stop and turn. Context is also extended to cover external factors that are specific to vehicle planning like stop lights, pedestrians, and weather.

3.2 Rule Input

It is required that the rules are written in AIR (AMORD In RDF)[3]. AIR is a Semantic Web-based rule language that can generate and track explanations for its inferences and actions and conforms to Linked Data principles. Another advantage of AIR is that it supports Linked Rules, which can be combined and reused in a manner similar to Linked Data. Since the AIR explanations themselves are Semantic Web data, they can be used for further reasoning. A benefit to using these Semantic web standards and data is that you can find related data, rules and concepts easily.

For the perception problem, the rules are from Schanks' conceptual primitives (Schank, 1972). An example rule for the primitive "MOVE" is that the actor must be animate, or the actor must be propelled by something. Other rules require more commonsense knowledge—for "INGEST" the action of consuming food and drink must be through the mouth or stomach of the actor.

For the vehicle planning problem, rules are derived from the Cambridge driving handbook. These rules can be easily changed to express the rules of the road for other states and areas. For example, the "right on red" turning rule is explicitly banned in most intersections in the greater Boston area, although legal in the state of Massachusetts. Some basic driving rules are shown in the Appendix in Figure 3.

3.3 Reasoning and Explanations

AIR nicely captures the reasons and descriptions necessary to output explanations. There is a bit of post-processing, since the output is written in RDF. Using python and RDFLIB[4], the output RDF file is parsed for the justifications and rule descriptions, which are combined together into a human-readable explanation. For example, if the pedestrian rule is violated, then the resulting description is "Pedestrian detected." This is combined with the other rules fired (like the speed rule-do not make a sudden stop at high speeds) to create the explanation– "Since there is a pedestrian in the road, move is unreasonable."

4 Evaluation

The monitoring system is evaluated in two ways–by validating the judgment of reasonableness, and invoking a user study to evaluate how well our system can generate explanations. The output of the monitor is binary judgment, indicating whether the proposed input is reasonable or not, and a human readable explanation that explaining that judgment. Examples of these explanations can be found the appendix. With this evaluation, this work aims to answer the following two questions:

1. How accurate is the system at detecting unreasonability?
2. Are user's convinced that the statements provide a convincing explanation for reasonable or unreasonable input?

[3] `http://dig.csail.mit.edu/2009/AIR/`
[4] `https://github.com/RDFLib/rdflib`

4.1 VALIDATION

Since I have not been able to implement our model on a deployed system with real data, I developed our own test sets based on uses cases from interviews with potential customers. The perception description test set is from 100 descriptions that I previously developed for a specific system. The descriptions are equally split between unreasonable and reasonable, with different verbs, subjects, objects, and contexts.

For the vehicle action test cases, I developed 24 examples. These examples were generated from four lights (red, yellow, green, no light), three motions (move forward, turn, stop), and a binary choice for obstructions (pedestrian or no pedestrian). For validation purposes, I checked that our monitor can determine whether a perception description or a vehicle action is reasonable or not. Each description of a vehicle action or perception description is labeled with a 1 or 0 as reasonable (1) or unreasonable (0).

The adaptable monitoring system judges reasonableness with 100% accuracy on the vehicle action test set. Since there are a countable number of rules and combinations, this makes sense. However, when deploying the system in a working vehicle simulation or platform, I will need to create more sophisticated and complex rules, which may cause the monitor to perform less accurately.

4.2 USER STUDY

In order to evaluate whether our explanations are convincing, I recruited 100 users to evaluate our explanations. Users were recruited from Amazon Mechanical Turk and instructed to rate each explanation on a five point Likert scale from "not convincing" to "very convincing."

Participants were presented with 40 explanations, evenly split between reasonable and unreasonable judgments. There were 20 vehicle planning explanations, and 20 perception description examples. I presented only 40 explanations to avoid exhaustion. Participants were instructed rate how convincing the explanations were, on a scale from 1 to 5. The average score over all explanations was 3.94, indicating that most users were moderately convinced that the explanations. The survey also included an optional question for users to explain why they choose their indicated rating for each explanation. A table of a sample of explanations for the four types of explanations tested: reasonable vehicle plans, unreasonable vehicle plans, reasonable perception descriptions, and unreasonable perception descriptions is in the Appendix in Table 1.

Users were convinced of the monitor's explanations, since all explanations were rated at an average above 3.5. A distribution of ratings can be found in Figure 1. In general, users were slightly more convinced by two factors. Firstly, reasonable were rated higher than unreasonable statements. This could be due to positive bias (Kareev, 1995), demonstrating that people are generally more favorable to positive examples. Secondly, perception description explanations (both reasonable and unreasonable) were rated higher than vehicle planning explanations. This could be attributed to users being less familiar with vehicle rules. Both differences were not statistically significant.

5 DISCUSSION AND FUTURE WORK

This work presents a framework and a for a generic monitoring system that can *judge* and *explain* the reasonableness of a machine learning algorithms' output, given a set of rules. The log data is represented in RDF and the rules are written in AIR as to make the system a framework that is easy to augment and adapt to other applications.

Although this system works fairly well in practice, it does have some obstacles to overcome to become a deployable system. I developed a parser to represent our data in RDF, but this may have to be done manually in different applications. Our use case of a perceptual "scene understander," sometimes referred to as an opaque, deep neural network, is over-simplified for demonstration purposes. In future work, I may want to apply this system to an actual image captioning system, although their description of "reasonableness" is slightly different. In that context, reasonableness is a caption that accurately describes the intended photo. That kind of monitoring is much harder to enforce than the reasonableness rules that I have defined in this paper. This system is currently being integrated

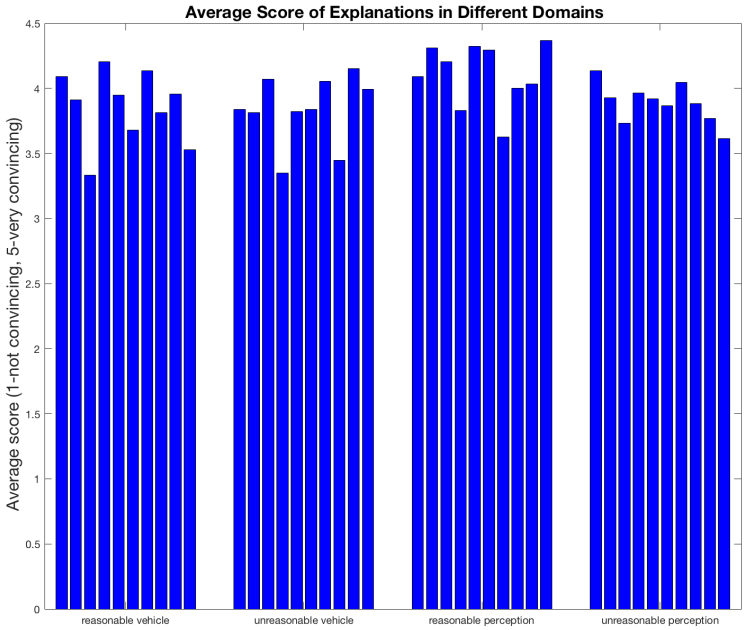

Figure 1: Average rating of 40 explanations over 100 user trials. There were 4 sets of explanations (from left to right): reasonable vehicle plans, unreasonable vehicle plans, reasonable perception descriptions, and unreasonable perception descriptions

into the RACECAR platform[5] as a preliminary application. Our end goal is to work with vehicle engineers to alter our monitoring system for vehicle actions that can detect and explain false positives and false negatives with sensor information. Our hope is that by implementing such a system with a safety driver, they will be able to interact with our system and provide real-time feedback and evaluation of the system. I am also currently adding capability to *learn* rules automatically. These rules, or the output of such rules would need to be verified or labeled by a user in a working system.

The use cases are simplified for demonstration purposes. The vehicle actions have limited context and perceptual use case is simplified. Also, the details of the types of rules for the perception description use case is outline in prior work Gilpin et al. (2018). In future work, these cases will be expanded to cover complex corner cases that are typically not well-represented in the training data.

6 Conclusion

Machine learning algorithms work fairly well in practice, but when they fail, diagnosing the root-cause is difficult. More so, developing an explanation of how and why they failed is even harder. Self-monitoring constructs, like the one proposed in this paper, is a small step towards developing machines without errors, that are more trustworthy, and that perform reasonably, as we expect them to.

Understanding and detecting inconsistencies in autonomous systems is an interesting and challenging problem, especially for opaque learning systems. In general, these complex algorithms work very well in practice. However, a monitor, like the one proposed in this paper, can provide a final "sanity-check" machine learning algorithms, especially those making mission-critical or safety-critical tasks.

[5] `http://fast.scripts.mit.edu/racecar/`

REFERENCES

Jose Vicente Abellan-Nebot and Fernando Romero Subirón. A review of machining monitoring systems based on artificial intelligence process models. *The International Journal of Advanced Manufacturing Technology*, 47(1-4):237–257, 2010.

Philip R Cohen and Hector J Levesque. Rational interaction as the basis for communication. Technical report, SRI International, 1988.

Allan Collins and Ryszard Michalski. The logic of plausible reasoning: A core theory. *Cognitive science*, 13(1):1–49, 1989.

Johan De Kleer. An assumption-based tms. *Artificial intelligence*, 28(2):127–162, 1986.

Leilani H. Gilpin, Jamie C. Macbeth, and Evelyn Florentine. Monitoring scene understanders with conceptual primitive decomposition and commonsense knowledge. *Advances in Cognitive Systems*, 2018.

Yaakov Kareev. Positive bias in the perception of covariation. *Psychological review*, 102(3):490, 1995.

Ankesh Khandelwal, Jie Bao, Lalana Kagal, Ian Jacobi, Li Ding, and James Hendler. Analyzing the air language: a semantic web (production) rule language. In *International Conference on Web Reasoning and Rule Systems*, pp. 58–72. Springer, 2010.

Johan de Kleer, Jon Doyle, Charles Rich, Guy L Steele Jr, and Gerald Jay Sussman. Amord: A deductive procedure system. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1978.

Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

Erfan Najmi, Zaki Malik, Khayyam Hashmi, and Abdelmounaam Rezgui. Conceptrdf: An rdf presentation of conceptnet knowledge base. In *Information and Communication Systems (ICICS), 2016 7th International Conference on*, pp. 145–150. IEEE, 2016.

Roger C Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4):552–631, 1972.

S. Shaheen, W. El-Hajj, H. Hajj, and S. Elbassuoni. Emotion recognition from text based on automatically generated rules. In *2014 IEEE International Conference on Data Mining Workshop (ICDMW)*, volume 00, pp. 383–392, Dec. 2014. doi: 10.1109/ICDMW.2014.80. URL doi.ieeecomputersociety.org/10.1109/ICDMW.2014.80.

APPENDIX

```
foo:my_actor
    a ont1:Subject ;
    ont1:phrase "a wall" .

foo:my_object
    a ont1:Object ;
    ont1:phrase "the street" .

cd:move
    a ont1:Move ;
    ont1:actor foo:my_actor ;
    ont1:object foo:my_object ;
    ont1:verb "cross" .
```

Figure 2: The RDF log of "a wall crossing the street."

```
:safe_car_policy a air:Policy;
  air:rule :light-rule;
  air:rule :pedestrian-rule;
  air:rule :right-turn-rule;
  air:rule :speed-rule .

:pedestrian-rule a air:Belief-rule;
  air:if {
    foo:some_pedestrian
    ont1:InPathOf foo:my_car.
  };
  air:then [
    air:description ("Pedestrian detected");
    air:assert [air:statement{
        foo:my_car
        air:non-compliant-with
    :safe_car_policy .}]];
  air:else [
    air:description ("No obstructions");
    air:assert [air:statement{
        foo:my_car
        air:compliant-with
        :safe_car_policy .}]] .

:light-rule a air:Belief-rule;
  air:if { :EVENT a :V;
    ont1:Location
    foo:some_traffic_light.
};
  air:then [air:rule :traffic-light-rule].
```

Figure 3: A subset of the safe-driving rules written in AIR.