On Human Intellect and Machine Failures: Troubleshooting Integrative Machine Learning Systems*

Besmira Nushi Ece Kamar Eric Horvitz Donald Kossmann
Microsoft Research, Redmond, WA, USA
benushi,eckamar,horvitz,donaldk@microsoft.com

ABSTRACT

We study the problem of troubleshooting machine learning systems that rely on analytical pipelines of distinct components. Understanding and fixing errors that arise in such integrative systems is difficult as failures can occur at multiple points in the execution workflow. Moreover, errors can propagate, become amplified or be suppressed, making blame assignment difficult. We propose a human-in-the-loop methodology which leverages human intellect for troubleshooting system failures. The approach simulates potential component fixes through human computation tasks and measures the expected improvements in the holistic behavior of the system. The method provides guidance to designers about how they can best improve the system. We demonstrate the effectiveness of the approach on an automated image captioning system that has been pressed into real-world use.

1 PROBLEM CHARACTERIZATION

Advances in machine learning have enabled the design of integrative systems that perform sophisticated tasks via the execution of analytical pipelines of components. Despite the widespread adoption of such systems, current applications lack the ability to understand, diagnose, and fix their own mistakes which consequently reduces users' trust and limits future improvements. Therefore, the problem of understanding and troubleshooting failures of machine learning systems is of particular interest in the community [1–4, 8, 9]. This work studies *component-based* machine learning systems composed of specialized components that are individually trained for solving specific problems and work altogether for solving a single complex task. Our goal is to assist system designers by (ii) identifying and quantifying the different types of *system failures*, and (2) measuring the impact of potential *component fixes* in the overall system quality for guiding system improvement decisions.

We analyze how the following characteristics of these integrated learning systems make it challenging to assign blame to individual components:

Continuous quality measures. Uncertainty is inherent in machine learning components. When these components work together to solve complex tasks, the measure of quality for individual components and the system as a whole is no longer binary, rather it spans a wide spectrum. Troubleshooting in this quality continuum where all components are only partially correct is non-trivial. Therefore, the evaluation of these systems needs to go beyond accuracy metrics to deeper analysis of system behavior.

Complex component entanglement. In integrative learning systems, components have complex influences on each other as they may be tightly coupled or the boundaries between their responsibilities may not be clear. When the quality of a component depends on the output of previous components, blame cannot be assigned to individual components without decoupling imperfection problems in component inputs.

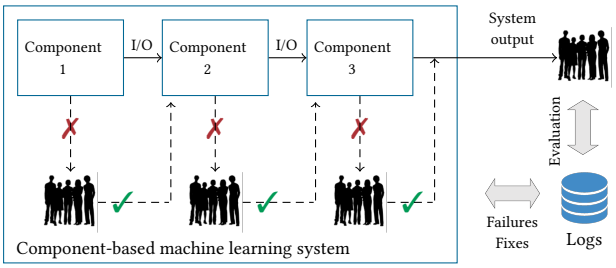

Figure 1: Troubleshooting with humans in the loop.

Non-monotonic error. Oftentimes, improving the outputs of single components does not guarantee holistic system improvement. On the contrary, doing so may lead to quality deterioration. For example, when components are tuned to suppress erroneous behavior of preceding components, applying fixes to the earlier ones may result to unknown failures.

These challenges hinder future system improvements as designers lack an understanding of how different potential fixes on components may improve the overall system output.

2 METHODOLOGY

Overview. We introduce a troubleshooting methodology which relies on crowdworkers to identify and fix mistakes in existing systems. Human intervention is crucial to the approach as human fixes simulate improved component output that cannot be produced otherwise without significant system development efforts. Figure 1 shows the main flow of our approach. First, workers evaluate the system output without any fix to analyze the current system state. To simulate a component fix, the input and output of the component accompanied with the fix description are sent to a crowdsourcing platform as microtasks. Once workers apply the targeted fixes for a component, the fixed output is integrated back into the running system, which thereby generates an improved version of the component. The system is executed as a simulation with the injected fix and the output is evaluated again via crowdworkers. The overall process collects a valuable set of log data on system failures, human fixes, and their impact on the final output. This data can then be analyzed to identify the most effective combination of component improvements to guide development efforts.

The execution of the methodology is guided by the *fix workflow*, which is a combination of various component fixes to be evaluated. Based on the system architecture, the system designer chooses

*This work was previously published in AAAI 2017[7].

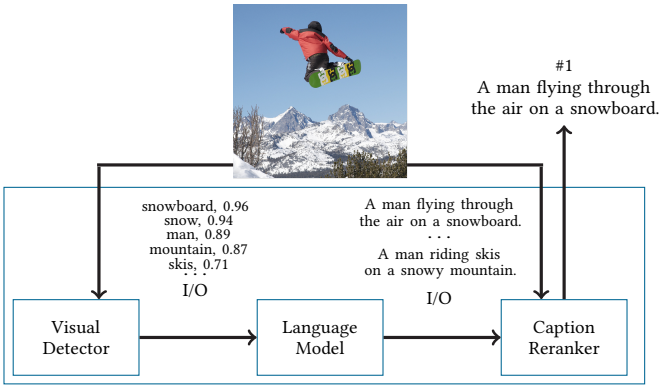

Figure 2: The image captioning system.

which fix workflows to execute and evaluate for the purpose of troubleshooting.

Troubleshooting outcomes. Applying human fix workflows helps system designers to observe the effect of component fixes on system performance, overcoming the challenges raised by the problem characteristics.

(1) *Continuous quality measures* — Comparing the system quality before and after various fix workflow executions not only can quantify the current quality of system and component output, but it can also isolate and quantify the effect of individual component fixes. For example, if many components are partially failing and are possibly responsible for a specific error, the designer can test the respective fixes, systematically understand their impact, and decide which are the most promising ones.

(2) *Non-monotonic error* — Non-monotonic error propagation is disclosed when the overall system quality drops after a component fix. When such a behavior is observed, the system designer can conclude that although these fixes may improve the internal component state, they are not advisable to be implemented in the current architecture as they produce negative artifacts.

(3) *Complex component entanglement* — Entanglement detection requires the execution of workflows with different combinations of fixes to measure the individual and the joint effect of component fixes. For example, if two consecutive components are entangled, individual fixes in either one of them may not improve the final output. However, if both components are fixed jointly, this may trigger a significant improvement. The designer could also use this information to detect entanglement and correct the system architecture.

3 CASE STUDY AND RESULTS

We apply our methodology to a state-of-the-art integrative learning system developed to automatically caption images [5]. As shown in Figure 2, the system involves three machine learning components in a pipeline. The *Visual Detector* takes an image as an input and detects a list of words associated with recognition scores. The *Language Model* generates likely word sequences as captions based on the words recognized from the Visual Detector, without having access to the input image. Finally, the *Caption Reranker* reranks the captions generated from the Language Model and selects the best match for the image based on <image,caption> similarity scores. The multimodal nature of this case study allows us to demonstrate the applicability of the approach for components processing different forms of data and carrying out various tasks.

Component	Fix description
Visual Detector	Add / remove objects
Visual Detector	Add / remove activities
Language Model	Remove noncommonsense captions
Language Model	Remove non-fluent captions
Caption Reranker	Rerank Top 10 captions

Table 1: Summary of fixes for the image captioning system.

	No fix	Visual Detector	Language Model	Caption Reranker	All fixes
Accuracy	3.674	4.035	3.712	4.145	**4.451**
Detail	3.563	3.916	3.602	3.966	**4.247**
Language	4.509	4.432	4.632	4.626	**4.660**
Commonsense	0.957	0.942	0.982	0.988	**0.998**
General	3.517	3.831	3.572	3.973	**4.264**
%Satisfactory	57.8%	68.0%	59.3%	73.6%	**86.9%**

Table 2: Summary of results.

For this system we implemented crowdsourcing tasks for (i) evaluating the system output, and (ii) simulating distinct component fixes. Table 1 lists all component fixes specifically designed for this case study. Each fix shows to crowdsourcing workers the respective component input and output for a particular image and requires workers to fix the output based on the task guidelines. For example, for the Visual Detector, crowdsourcing workers can add new items in the list of automatically detected words or they remove non-relevant words.

The evaluation of the captioning system with our methodology uses an Evaluation dataset of 1000 images randomly selected from the MSCOCO validation dataset[6]. Table 2 shows a summary of results including the improvements from each component and the complete fix workflow which sequentially applies all component fixes. A complete analysis using both human and automatic machine translation scores can be found in the full version of the paper[7]. Experiments highlight the benefits of making informed decisions about component fixes as their effectiveness varies greatly (18%, 3% and 27% for the three components respectively). In contrast to initial assumptions of system designers, fixes in the Caption Reranker are the most effective ones due to detected entanglement in the previous two components. This entanglement then causes non-monotonic error behavior for the Visual Detector fixes, which we can observe and report with our methodology. Finally, the complete fix workflow increases the number of satisfactory captions by 50%.

Conclusion. As machine learning models become more mature, integrating them into larger systems creates opportunities for solving highly complex tasks. However, troubleshooting component-based learning systems is a tedious process due to the inherent uncertainty and entangled dependencies of machine learning components. Therefore, the work of system designers goes beyond initial system development and extends to continuously monitoring, evaluating, and improving these integrative systems. The proposed human-in-the-loop troubleshooting methodology supports system designers by providing a rich counterfactual analysis, which is informative for taking future improvement decisions.

REFERENCES

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[2] Sean Andrist, Dan Bohus, Ece Kamar, and Eric Horvitz. 2017. What Went Wrong and Why? Diagnosing Situated Interaction Failures in the Wild. In *International Conference on Social Robotics*. Springer, 293–303.

[3] Dan Bohus, Sean Andrist, and Mihai Jalobeanu. 2017. Rapid Development of Multimodal Interactive Systems: A Demonstration of Platform for Situated Intelligence. (2017).

[4] Shanqing Cai, Eric Breck, Eric Nielsen, Michael Salib, and D Sculley. 2016. TensorFlow Debugger: Debugging Dataflow Graphs for Machine Learning. (2016).

[5] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *CVPR*. 1473–1482.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV 2014*. Springer, 740–755.

[7] Besmira Nushi, Ece Kamar, Eric Horvitz, and Donald Kossmann. 2017. On Human Intellect and Machine Failures: Troubleshooting Integrative Machine Learning Systems.. In *AAAI*. 1017–1025.

[8] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *NIPS*.

[9] Mark Staples, Liming Zhu, and John Grundy. 2016. Continuous validation for data analytics systems. In *Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on*. IEEE, 769–772.