

Characterizing and Detecting Mismatch in ML-Enabled Systems

Problem

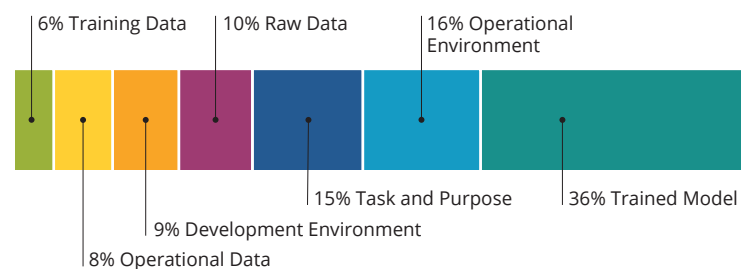
Development, deployment, and operation of ML systems involves three perspectives, often with three completely separate workflows and people: data scientists build the model; software engineers integrate the model into a larger system; and then operations staff deploy, operate, and monitor the system.

Because these perspectives operate separately and often speak different languages, there are opportunities for mismatch between the assumptions made by each perspective with respect to the elements of the ML-enabled system, and the actual guarantees provided by each element.

Solution

Develop descriptors for elements of ML-enabled systems by eliciting examples of mismatch from practitioners; formalizing definitions of each mismatch in terms of data needed to support detection; and identifying potential for using this data for automation of mismatch detection.

Phase 1: Practitioner interviews to elicit examples of mismatch and their consequences



Resulting Mismatch Categories from Practitioner Interviews

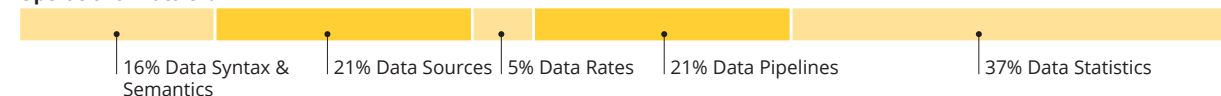
Descriptors for ML system elements make stakeholder assumptions explicit and prevent mismatch.

Phase 1 Findings

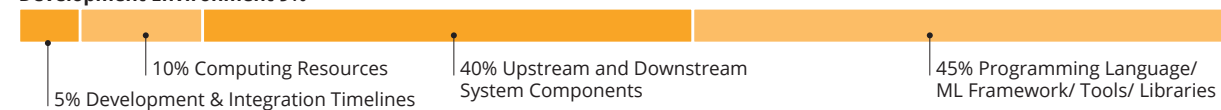
Training Data 6%



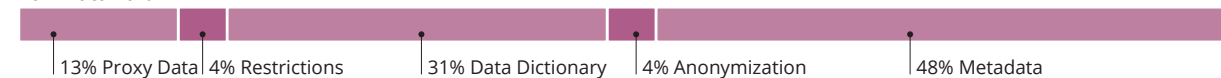
Operational Data 8%



Development Environment 9%



Raw Data 10%



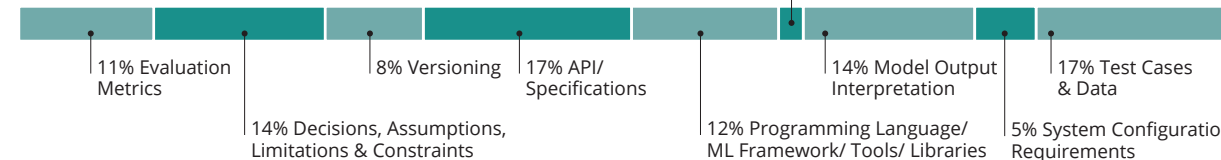
Task and Purpose 15%



Operational Environment 16%



Trained Model 36%



Training Data mismatches are mostly due to lack of clarity on data preparation pipelines (37%) and lack of data statistics (21%).

Operational Data mismatches are mostly due to lack of data statistics (37%) and lack of clarity on data pipelines (21%).

Development Environment mismatches are mostly due to differences in programming languages ... (45%) and lack of knowledge of upstream and downstream components (40%).

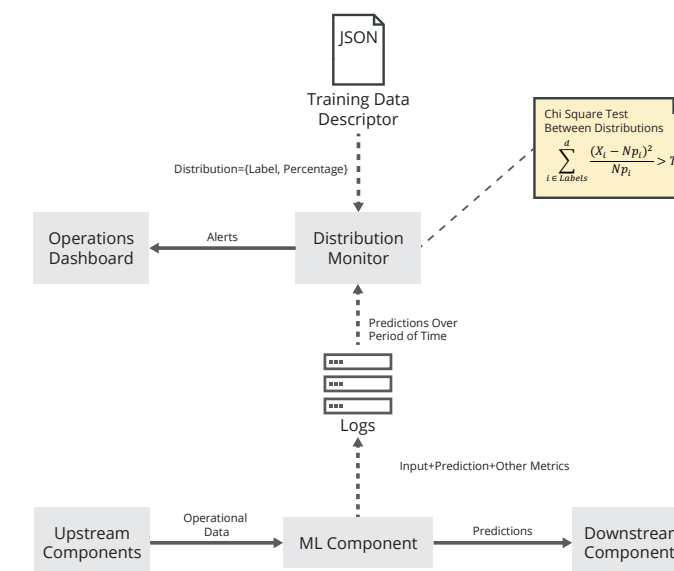
Raw Data mismatches are mostly associated with lack of metadata (48%) and lack of a "data dictionary" (31%).

Task and Purpose mismatches are mostly associated with unknown business goals (29%) or success criteria (26%).

Operational Environment mismatches are mostly associated with unavailable runtime metrics and data (54%) and unawareness of computing resources available for model serving (32%).

Trained Model mismatches are mostly associated with lack of test cases and test data (17%) and lack of model specifications and APIs (17%).

Looking Ahead: Automated Mismatch Detection



Descriptors Being Used for Automated Drift Detection

Copyright 2020 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

DM20-0865