Critique of Hidden Technical Debt in Machine Learning Systems

YIMING TANG, The Graduate Center, CUNY, USA

Additional Key Words and Phrases: machine learning, technical debt, software engineering

ACM Reference Format:
Yiming Tang. 2020. Critique of Hidden Technical Debt in Machine Learning Systems. 1, 1 (April 2020), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

CRITIQUE

Although Machine Learning (ML) is successful in speeding up the process of building large prediction systems, there is still much technical debt which brings the huge cost to machine learning maintenance. The challenge is that this technical debt is hard to discover because it is on the system level. The paper investigates the system-level interactions and interfaces to find the factors leading to technical debt.

Unlike traditional software engineering development, the development of ML systems does not create a strong abstraction boundary to help software maintenance in the future. Developers should isolate models to ease the pain that if developers modify anything, it will change everything in the system. Developers also need to reduce system dependency and control undeclared consumers for the system.

Data dependencies are difficult to detect. Due to unstable data dependencies, such as unstable input signals, it is hard for developers to improve the system. Version control may solve this issue, but it could bring other costs. ML systems may have underutilized dependencies that should be removed. A system model can influence itself in the future. Besides, two systems can influence each other. The improvement in one system may influence its future version or other systems, which brings the maintenance costs.

The paper proposed 5 ML system anti-patterns which should be avoided. Using glue-code may bring extra test costs to ML systems due to a lack of domain-specific code for the system. Pipeline jungles may be difficult for developers to manage. Massive and accumulated dead experimental codepaths increase the testing and maintenance cost for developers. ML system design is lacking abstraction support. There is even no standard interface to describe data as a relational database. Besides, code smells can bring potential problems.

For a large and mature ML system, the number of source lines for configurations is huge. Each line may lead to an underlying mistake. Configurations may be difficult to modify for this system. The paper mentions 6 rules to help build an ML system with good configurations. ML system always interacts with the external world but the external world is always unstable. New input data may change the decision thresholds for the system. Unit testing may not be sufficient for an ML system. Testing invariants may not be obvious for an ML system.

Author's address: Yiming Tang, The Graduate Center, CUNY, USA, ytang3@gradcenter.cuny.edu.

Manuscript submitted to ACM

Except for the technical debt mentioned above, some other ML-related technical debt may exist. The paper also proposes some solutions. For example, teams should create their own culture to enhance long-term ML system development due to the gap between ML research and software engineering.

This paper investigates ML systems from a software engineering perspective, which is innovative, as we all know that there is a hard line between ML research and software engineering. This paper mentions some risk factors and solutions for ML specific technical debts. In my opinion, these factors should have different influences on the ML system development. Can we know which one is more important than others? In addition, this paper was published in 2015. In that year or after that year, more and more Machine Learning frameworks were released, such as Tensorflow. Do these frameworks help developers eliminate some technical debt mentioned in this paper?