Technical Debt In Machine Learning System – A Model Driven Perspective

Soumen Sarkar | September 9, 2022 at 5:23 pm

This article is part 2 of the two part series on Technical Debt in Machine Learning Systems development. The link to part 1 is listed below:

A Model For Technical Debt In Machine Learning Systems

Part 1 article recap, here is what was covered

Introduced a simple yet powerful Model of Technical Debt for Machine Learning Systems. The model is simple to remember, easier to extend, and provides a reliable means for reliable and maintainable Machine Learning Systems. This, in a nutshell, is the value proposition of this post.

Introduced four dimensions of the Model, namely, **Time**, **Input**, **System** and **Output**.

Clarified that system development incompetence or just plain professional laziness or expediency-driven shortcuts do not count towards Technical Debt. In other words, Technical Debt is consciously commissioned as opposed to accumulating via acts of omission. Neglect Debt is not considered to be Technical Debt.

Prescribed the best practice of tracking consciously admitted Technical Debt in a registry similar to Architectural Decision Records. The practice of the Tech Debt Registry is expected to serve as an automatic alignment mechanism to achieve budget allocation towards tech debt remediation per quarter or similar cadenced development discipline.

Finally, the alternative terminology of Technical Delta was introduced. The linguistic trick of describing Delta compared to Debt often eases discussion with Finance executives since Debt is a formal term in Finance. Technical delta is the difference between your systems' internals today and your systems' ideal internals for that same functionality. The Delta point of view is broader than the Debt definition in that both Neglect Debt and Technical Debt contribute to the deviation Delta (deviation between Ideal System and Present System).

Any discussion on Model should almost always refer to the following famous quote attributed to statistician George E. P. Box:

All models are wrong, but some are useful

The aphorism acknowledges that models of our knowledge always fall short of the complexities of reality but can still be useful nonetheless. With this model background, let us delve into this article focusing on specific technical debt in Machine Learning System development.

Technical Debt In Machine Learning Systems

The Technical Debt anomalies may be restated for a Machine Learning System as follows.

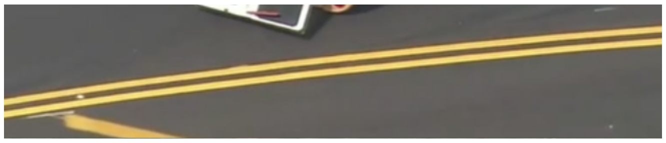
As **Time** passes …

1. The prediction reliability of the ML **System** (i.e. **Output**) degrades.
2. It becomes harder to train the ML **System** for newer **Input**.

3. It becomes harder to comprehend the ML **System** to maintain efficiently.

Machine Learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed. Machine learning algorithms use historical data as input to predict new output values. This reminds us that the lack of explicit programming is both a blessing and a curse. It is a curse since it does not hazve the rigidity and predictability of an explicitly programmed system! Thus the System may behave in surprisingly unexpected ways for minor differences in Input, thus precipitating a crisis of trust in ML System reliability.

Take a picture of a school bus. Flip it so it lays on its side, as it might be found in the case of an accident in the real world. A 2018 study found that state-of-the-art ML Systems that normally correctly identify the right-side-up school bus failed to do so on average 97 percent of the time when it was rotated. One possible way to make AIs more robust against such failures is to expose them to as many confounding “adversarial” examples as possible.



Time

The Input and the System are all changing with Time. Why are they changing? The Input is changing since society itself is changing. Abundant and cheap computation has driven the abundance of data we are collecting and the increased capability of machine learning methods.

In this context, if the ML System does not keep track of the nature of Input which affects the nature of Output, this lack of tracking becomes a Technical Debt (since it causes delivery friction).

Mitigation: Monitoring & Testing

The key question is: what to monitor? Testable invariants are not always obvious, given that many ML systems are intended to adapt over time.

Monitor for Prediction Bias. Biases can result from the data or algorithm used to train your model. For instance, if an ML model is trained primarily on data from middle-aged individuals, it may be less accurate when making predictions involving younger and older people. In a system that is working as intended, it should usually be the case that the predicted labels’ distribution is equal to the observed labels’ distribution.

Bias can be introduced or exacerbated in deployed ML models when the training data differs from the data that the model sees during deployment (that is, the live data). These changes in the live data distribution might be temporary (for example, due to some short-lived, real-world events) or permanent. In either case, it might be important to detect these changes. For example, the outputs of a model for predicting home prices can become biased if the mortgage rates used to train the model to differ from current, real-world mortgage rates.

Technical Debt Management

Document the current bias monitoring and plans in the technical debt registry. Describe whether alerts are implemented and whether remediation is automatic or manual.

Input / Output

Remember the earlier example of a school bus flipped on the side? Most ML systems could not handle such simple disorientation. One possible way to make ML more robust against such failures is to expose them to as many confounding “adversarial” examples as possible and document the data. This is a good example of Input Technical Debt for machine learning systems.

Properly Documenting Data – Avoiding Input Technical Debt

Data is central to the development and evaluation of machine learning models. Many responsible ML harms can be traced back to the characteristics of datasets. For example, a lack of appropriate representation of different groups of people can lead to models that exhibit performance disparities. Spurious correlations and other unanticipated anomalies in training datasets can result in models that fail to generalize. Subjectivity in dataset labels and inaccurate notions of ground truth can result in models with misleading outputs. Documenting datasets helps promote more deliberate reflection and transparency about how these datasets might affect machine learning models. For dataset creators, documenting your data can help you think through underlying assumptions, potential risks, and use implications. It can also help dataset consumers—those who will use a dataset to develop or evaluate their models—make informed decisions about whether specific datasets meet their needs and what limitations they need to consider. For these reasons, good data documentation practices are an essential component of responsible AI.

In 2018, Microsoft introduced datasheets for datasets, a tool for documenting the datasets used for training and evaluating machine learning models. Datasheets contain questions about dataset motivation, composition, collection, pre-processing, labeling, intended uses, distribution, and maintenance. Crucially, and unlike other tools for meta-data extraction, datasheets are not automated but are intended to capture information known only to the dataset creators and often lost or forgotten over time.

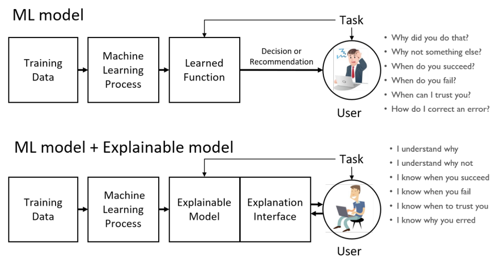
Often Output from an ML system is Input to another ML system. Hence the considerations for avoiding Input Technical Debt equally apply to Output. The machine learning community has no standardized process for documenting datasets, leading to severe consequences in high-stakes domains. In the electronics industry, every component, no matter how simple or complex, is accompanied by a datasheet that describes its operating characteristics, test results, recommended uses, and other information. By analogy, Microsoft proposes that every dataset be accompanied by a datasheet that documents its motivation, composition, collection process, recommended uses, etc. **Datasheets for datasets** will facilitate better communication between dataset creators and dataset consumers and encourage the machine learning community to prioritize transparency and accountability.

System

The biggest System Technical Debt with Machine Learning models is **Explainability** (also known as the **Black Box Problem**).

With gaining popularity and its successful application in many domains, Machine Learning (ML) also faced increased skepticism and criticism. In particular, people question whether their decisions are well-grounded and can be relied on. As it is hard to comprehensively understand their inner workings after being trained, many ML systems — especially deep neural networks — are essentially considered black boxes. This makes it hard to understand and explain the behavior of a model. However, explanations are essential to trust that the predictions made by models are correct. This is particularly important when ML systems are deployed in decision support systems in sensitive areas impacting job opportunities or even prison sentences. Explanations also help to correctly predict a model's behavior, which is necessary to avoid silly mistakes and identify possible biases. Furthermore, they help to gain a well-grounded understanding of a model, which is essential for further improvement and to address its shortcomings.

Explainable ML attempts to find explanations for models that are too complex to be understood by humans. The applications range from individual (local) explanations for specific outcomes of black-box models (e.g., Why was my loan denied? Why is the prediction of the image classifier wrong?) to global analysis that quantifies the impact of different features (e.g., What is the biggest risk factor for a particular type of cancer?).



Reference

I wanted to write the reference section for the articles (parts 1 & 2) differently compared to just listing the sources. I wanted to tell the story of how this article was made. In other words, the reference section narrates the Architecture Of Thought behind the article reference section narrates the Architecture Of Thought behind the article.

I got the idea of making the Reference section as a Thought Architecture from

The Minto Pyramid Principle: A compelling process for producing everyday business documents

The Minto Pyramid Principle® is the compelling process for producing everyday business documents – to-the-point memos, clear reports, successful proposals, or dynamic presentations.

There are few models for the Model aspect of Technical Debt, but they mostly deal with the cost aspect. This is often welcome since unmanaged Technical Debt poses an Opportunity Cost for maintaining the feature velocity. However, I wanted to take a Systems Thinking point of view, and I did not find any preexisting work for this viewpoint. It is important to be clear on what is not Technical Debt. For this the article borrowed from

What Is Not Technical Debt by Heather Abbott

I came to know of Python GIL (technical debt) issue from a recent (May 2022) news article from The Register:

Python is getting faster: Major performance tweaks on horizon

The examples of hidden technical debt in machine learning systems is sourced from the following paper

Hidden Technical Debt in Machine Learning Systems – NIPS papers

The article chooses technical debt wherever the example helped clarify the model for Technical Debt in Machine Learning Systems. The article does not claim to be an exhaustive list of technical debts in machine learning systems. In the following, further references are presented in traditional list form:

- Dynatrace Blog : Understanding Black-Box ML Models with Explainable AI
- IEEE Spectrum : 7 Revealing Ways AIs Fail
- Maha Amami : Leveraging explainability in real-world ML applications
- The Tyranny Of Time – Noema Magazine