Technical Debt in Machine Learning Systems

Muhammed Emin Kılıçaslan

Briefly Technical Debt

- As a term, technical debt is a metaphor invented by Ward Cunningham in 1992
- Although it is not measurable or possesses strict limits by definition, technical debt used to generalize different costly errors, neglections and overseeing.
- Titular paper, although I preferred to remove the word “hidden” for the evident debts are as major topic of the paper as hidden counterparts, examines the dissections of technical debts and, in places, points out their recommendations to avoid or reduce technical debts.

Reasons and Complex Models

- Some formats and designs in the development of traditional software engineering, such as encapsulation, is inevitably inherited to machine learning. Since the nature of ML systems is reliance of data, beclouding the access to data increases technical debt.
- This reliance makes it difficult to draw abstraction boundries in systems by enforcing intended behaviors on systems

Entanglement

- ML systems currently in use are thoroughly entangled, one simple error, alteration or even an improvement in a single can cause devastating ramifications to other parts of the system. It is labeled as CACE (Change Anything Change Everything) principle.
- One solution to reduce this debt is to isolate models. However in such cases improving one isolated model can cause the whole system performance to decrease.
- Another solution is live reviewing these changes and catgorizing them according to dimensions and slices.

Correction Cascades

- When there is a model for a certain problem does exist, but another model for a slightly different problem needed, in practice another version of the model with minute modifications can be used, but when this occurs multitude times, what we gonna en up with is a chain of models relying on the correctness of each others.
- Any unreported change can cause problems to next models relying on the output.

Undeclared Consumers

- Without an access control, consumers can use an alter the system outputs according to their intentions. This can cause unexpected and unstable results and increase the visibility debt.
- One solution against this is designing the system specifically to prevent undeclared usage of the data.

Data Dependencies

- In ML, data dependencies are noted to playing a more major role than code dependencies.
- **Unstable Data Dependencies:** In applications of ML, quality and quantity received from signals depending on real world can change rapidly. Mostly system also must be reactive and fit to input as fast. This usually can be because the separation of the ownership of model and the signals. Solution to this debt is to using versioned copy of a signal.

Underutilized Data Dependencies: Underutilized data is the data that provides little modeling benefit, inefficient compared to their procession cost. Underutilized data can vary in different types such as legacy features, those eventually grew redundant, bundled features, those are evaluated ensembled but some can have little to no addition to model, features those makes a little addition to model against its high cost and correlated features those one or many of them remains casual and can be removed without a major loss.

Static Analysis of Data Dependencies: Automatic annotation and dependency tree checks can be done to refrain from any possible unsafe migration and deletion.

Feedback Loops

- ML systems can affect themselves overtime if they are not updated in frequent duration. Direct feedback loops are the result of a model influencing the input data of its own. Bandit algorithms should be preferred in those cases over supervised algorithms which is currently in practice. Another variation is hidden feedback loops, which are much harder to detect due to coexistence of two separate models those outputs affect each other.

Glue Code

- Glue code is the name of the code those used to get into and out the data supplied from general purposed packages.
- Glue code is costly in this term because of being solely dependent on external packages and their peculiarities.
- Due to these supporting modifications, a mature system likely to end up with at least 95% glue code.
- One solution to this is turning black box codes into common APIs to use as intended.

Pipeline Jungles

- If not enough care given, an ML code likely to end up with a jungle of pipelines relying to one another.
- Those chained pipelines increase the research cost of the system.
- To solve these as well as glue code problem and lower the friction, development team must contain engineers and researchers working together.

Dead Experimental Codepaths

- When uncontrolled experimentational codes written implementing the system they will accumulate eventually.
- Those codepaths will branch the root system and cause unnecessary complexity as well as research cost.
- It is likely that only a small part of these branches will be used in practice, leaving the rest abandoned.

Abstraction Debt

- Map-reduce abstraction is practically popular abstraction in use although it is declared that it performs poorly on iterative ML systems.
- The parameter-server abstraction is recommended as a more robust alternative to map-reduce abstraction.

Common Smells

- Smell is an underlying designing error of a component or a system
- Plain-old-data type smell is problem of components being unaware of the use of data and features. In a robust system, functions and types of data explicitly defined to related components.
- Multiple-language smell is the result of using more than one language in a system which will limit the passing the ownership to other people.
- Prototype smell occurs due to excessive use of prototype, making the model useless to fit real world problems.

Configuration Debt

- Configuration debt accumulates mainly due to configuration is being treated as an afterthought.
- Configuration should be prioritized to consider while the creation of a system.
- System should be designed to make it harder to make mistakes while ease the observation, investigation and fixing of the data.

Dealing with Changes in External World

- Fixed thresholds are mostly declared manually by designers of the system.
- Eventual or rapid changes in world can cause these thresholds to age and become unfit. Reviewing and manually changing these thresholds are far from being practical.
- To solve this, systems should be designed with an internal reassigning component to check the thresholds regularly.

- Monitoring and acting a real time response is a vital part of the ML systems.
- Prediction bias although being unlikely to comprehensive, its results can be quite informative and help the programmers to have a vague idea of the changes.
- Action limits can have an alert system to inform the programmers for manual intervention in such changes.
- Up-stream producers will regularly check the source input signals the system being fed for ensuring quality level.

Other Areas of ML-Related Debt

Data Testing Debt: Basic sanity checks can be done to assuring the usefulness of data.

Reproducibility Debt: Due to ever-changing nature of the ML systems working on data extracted from external world, reproducibility of the results are limited.

Process Management Debt: Mostly mature systems uses dozens to hundreds of models to get the job done. Updating, configuring, assigning the resources and overall managing of that amount of models will have its costs.

Cultural Debt: In long term health of the productive teams, a team culture should be built on encouraging the reduction of complexity, improving reproducibility and stability.