Algorithm Debt: Challenges and Future Paths

Emmanuel Iko-Ojo Simon
Melina Vidoni
emmanuel.simon@anu.edu.au
melina.vidoni@anu.edu.au
Australian National University
Australia

Fatemeh H. Fard
fatemeh.fard@ubc.ca
University of British Columbia
Canada

ABSTRACT

Technical Debt (TD) is the implied cost of additional rework caused by choosing easier solutions in favour of shorter release time. It impacts software maintainability and evolvability, manifesting as different types (e.g., Code, Test, Architecture). Algorithm Debt (AD) is a new TD type recently identified as sub-optimal implementations of algorithm logic in scientific and Artificial Intelligence (AI) software. Given its newness, AD and its impact on AI-driven software remains a research gap. This poster aims to motivate reflective discussion on AD in AI software, by summarising findings, discussing its possible impact, and outlining future areas of work.

KEYWORDS

Technical Debt, Algorithm Debt, Artificial Intelligence, Software Engineering, Code Debt, Scientific Software

ACM Reference Format:
Emmanuel Iko-Ojo Simon, Melina Vidoni, and Fatemeh H. Fard. 2023. Algorithm Debt: Challenges and Future Paths. In *Proceedings of International Conference on AI Engineering – Software Engineering for AI (CAIN'23).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Technical Debt (TD) are the costs incurred in software development due to tasks postponed during its development process [14]. TD-ridden software is reportedly more prone to reduced maintainability and evolvability [13, 14], code-breaking defects, vulnerabilities, irreproducible results, and overall low-quality [5]. Most of the prior works studying TD centred around object-oriented programming languages (OOP) and their particularities [18], with limited research approaching scientific and Artificial Intelligence (AI) software.

Recent research uncovered **Algorithm Debt (AD)**–"sub-optimal implementations of algorithm logic" [10, 18], often found in performance-critical and algorithm-intensive projects such as AI frameworks and scientific software used to support research. AD's presence is concerning given the increasingly ubiquitous presence of AI in daily software [5, 10], and the lack of AD-specific research. In particular, those studies uncovering traces of AD did not investigate it per se, but found evidence of its existence while tackling different research questions [5, 10, 18]. Likewise, AD's definition [10] is derived from other TD types, and does not account for specific nuances that may exist.

Based on research of other TD types, low-quality code, inconsistent and unreproducible results, and faulty code that produces non-robust outputs are the cause of known problems for scientists [15, 18, 19]. With the widespread of AI and the reliance on AI-based systems, and the renewed interest on AI-explainability [12], the quality of these systems has become critical for their acceptance [7]–and given that nowadays AI penetrates all areas of science, industry and our private lives [6], thorough research on Algorithm Debt, its impact, causes, and management strategies should be carried out.

Motivational Examples. Albeit Algorithm Debt is a nascent concept, its negative impact in developing and using AI-driven software have consistently affected the non-academic public.

Economic Impact: *Knight Capital*[1] was a firm that specialised in executing trades for retail brokers. In 2012, its software development team was pressured to complete the changes to the AI-driven algorithms, which led them to take considerable shortcuts to speed the process. Due to this faulty implementation, the company unintentionally *purchased stocks worth $7 billion, only during their first hour of trading, leading to further $440M in cash losses.*

Live Impact:AI-software is also being used in safety critical systems, transportation, and military operations [6], where the effects of TD-ridden algorithm logic could have life-threatening consequences [1, 4]. For example, in 2018, Arizona recorded a pedestrian fatality, *where a person was tragically struck by an Uber Test Vehicle (a self-driving car) while crossing a four-lane road*[2]. The alleged cause was the AI's failure to "classify an object as a pedestrian unless that object was near a crosswalk".

Transitive Impact: Since AI software is mostly developed in package-based environment [17], e hypothesise that if there is a flaw in the algorithm logic of a framework, every application using that framework will transitively suffer from Algorithm Debt. Thus, research leveraging the "smelly" application may be *negatively impacted by validity threats, and potentially incorrect results.*

2 CURRENT CHALLENGES

AD was found on tangential works uncovering traces of it. We present examples, outline limitations, and define a path ahead.

Self-Admitted Technical Debt (SATD). Instances of AD were found while searching SATD in Deep Learning (DL) projects (e.g., `Is there a faster way to do pooling in the channelfirst`

[1]Case Study: Knight Capital
[2]Five Case Studies of AI Failures

CAIN'23, May 15-16th, 20th, Melbourne, Australia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

`case?`) [10], and R packages (e.g., `It seems this over-estimates the truth`) [18], and reportedly represent 5% of SATD in scientific software. Likewise, AD has only been detected on source code comments [16], even though SATD can occur in any human-written document that reflects developer interaction, such as commits, issues, pull requests, and code reviews.

Challenges. Despite traces of AD being uncovered, there is presently no commonly accepted definition; Liu et al. [10] proposed one specific to DL frameworks, and Vidoni [18] extended it to scientific software, but its nuances have not been addressed. Automated AD identification is also challenging [16], given both its scarcity and the lack of specific word-patterns; although they exist for other TD types [11, 18], they are not available for AD.

TD Smells and AD Detection. TD is often defined by its *smells*–specific symptoms in the source code related to a TD type [11]. Prior research searched for TD smells (and specifically, code smells) within AI software; e.g., Sculley et al. [15] found that Design Debt erode abstraction boundaries in Machine Learning (ML) systems, while others identified ML-specific code smells [9, 19]. There have also been advances regarding data-related smells [8].

Challenges. These prior works demonstrated that scientific and AI software has specific TD smells. However, there is no research regarding AD-specific smells, which hinders the creation of metrics to automatically and statically detect AD (e.g., through Automated Static Analysis Tools such as SonarQube[3]). This also constrains what type of investigations can be undertaken–if AD cannot be identified, it cannot be studied, managed, and/ or remedied.

AD and Explainable AI. Explainability clarifies AI-made decisions to the user (e.g., policy makers), to better comprehend the model's decision-making process. Overall, explainability is essential to support a model's accountability, and to ensure that users fully understand how the recommendations were produced [12].

Challenges. We argue that explainability may not be fully achieved if AD exists. Prior works [14, 19] show that TD limits what can be done and explainability is required for the understanding and uncovering of AD. Additionally, based on research in other TD types, AD may increase the complexity of a code, therefore it may hinder explainability. We also argue that TD has been shown to decrease readability and understandability of code from a developer's perspective [2], complicating the transparency of a model's implementation. We can also foresee that the more a system grows [3], TD tends to naturally accrue; so it will hinder the explanations if the system continues to grow and that is not clear enough.

3 FUTURE WORKS

Proposal #1. Defining AD will enable its assessment on multiple natural language sources (e.g., commits, issues, code reviews, pull requests); namely, all sources considered for SATD. This would allow investigating how AD may vary across disciplines, enabling its identification. Intuitively, code created for bioinformatics has considerable differences from social sciences, and self-admitted AD may vary (e.g., be written differently). SATD will allow discerning what nuances are commonly admitted by AI developers.

Proposal #2. We must identify AD's unique smells, including possible metrics for its detection, to allow determining how long AD survives, how it is fixed and managed. Without understanding AD's smells, it is not feasible to thoroughly analyse its causes and repercussions on the performance and behaviour of AI-driven software, nor in the broader scientific software. Also, these smells and metrics could then be used to craft detection tools that would be utilised for a study on the detection of AD within a code to assist in its management and with its mitigation in real-time.

Proposal #3. Investigating the impact of AD in AI's inconsistencies generated is critical–it would allow studying how developers' code-related decisions could transitively affect high-ranking policymaker that depend on AI-driven software. This is essential, because if AD remains under-researched, any issues arising from it will remain unaddressed. Given AI's current pervasiveness, the continued ethical discussions about its impact (e.g., AI-driven art), and the general public's predisposition, research into AD could also lead to creating quality standards for AI-driven software.

[3]Static analysis with Sonarqube- https://www.sonarsource.com/products/sonarqube

REFERENCES

[1] Jyotika Athavale, Andrea Baldovin, Ralf Graefe, Michael Paulitsch, and Rafael Rosales. 2020. AI and reliability trends in safety-critical autonomous systems on ground and air. In *Int. Conf. on Dependable Systems and Networks*. IEEE, 74–77.

[2] Gabriele Bavota and Barbara Russo. 2016. A large-scale empirical study on self-admitted technical debt. In *Int. Conf. on Mining Software Repositories*. 315–326.

[3] Terese Besker, Antonio Martini, Rumesh Edirisooriya Lokuge, Kelly Blincoe, and Jan Bosch. 2018. Embracing technical debt, from a startup company perspective. In *Int. Conf. on Software Maintenance and Evolution*. IEEE, 415–425.

[4] Kuo Chen, Jin Zhang, Narasimha M Beeraka, Mikhail Y Sinelnikov, Xinliang Zhang, Yu Cao, and Pengwei Lu. 2022. Robot-Assisted Minimally Invasive Breast Surgery: Recent Evidence with Comparative Clinical Outcomes. *Journal of Clinical Medicine* 11, 7 (2022), 1827.

[5] Zadia Codabux, Melina Vidoni, and Fatemeh H Fard. 2021. Technical debt in the peer-review documentation of r packages: A rOpenSci case study. In *Mining Software Repositories*. IEEE, 195–206.

[6] Frank Emmert-Streib. 2021. From the Digital Data Revolution toward a Digital Society: Pervasiveness of Artificial Intelligence. *Machine Learning and Knowledge Extraction* 3, 1 (2021), 284–298. https://doi.org/10.3390/make3010014

[7] Michael Felderer and Rudolf Ramler. 2021. Quality Assurance for AI-Based Systems: Overview and Challenges (Introduction to Interactive Session). In *Int. Conf. on Software Quality*. Springer, 33–42.

[8] Harald Foidl, Michael Felderer, and Rudolf Ramler. 2022. Data Smells: Categories, Causes and Consequences, and Detection of Suspicious Data in AI-Based Systems. In *Int. Conf. on AI Engineering: Software Engineering for AI* (USA). ACM, 229–239.

[9] Jiri Gesi, Siqi Liu, Jiawei Li, Iftekhar Ahmed, Nachiappan Nagappan, David Lo, Eduardo Santana de Almeida, Pavneet Singh Kochhar, and Lingfeng Bao. 2022. Code Smells in Machine Learning Systems. *arXiv preprint 2203.00803* (2022).

[10] Jiakun Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo, and Shanping Li. 2020. Is using deep learning frameworks free? characterizing technical debt in deep learning frameworks. In *ICSE: Software Engineering in Society*. ACM/IEEE, 1–10.

[11] Everton da S Maldonado and Emad Shihab. 2015. Detecting and quantifying different types of self-admitted technical debt. In *Int. Workshop on Managing Technical Debt*. IEEE, 9–15.

[12] Harshkumar Mehta and Kalpdrum Passi. 2022. Social Media Hate Speech Detection Using Explainable Artificial Intelligence. *Algorithms* 15, 8 (2022), 291.

[13] Aniket Potdar and Emad Shihab. 2014. An exploratory study on self-admitted technical debt. In *Int. Conf. on Software Maintenance and Evolution*. IEEE, 91–100.

[14] Nicolli Rios, Manoel Gomes de MendonCa Neto, and Rodrigo Oliveira SpInola. 2018. A tertiary study on technical debt. *Inf. and Soft. Tech.* 102 (2018), 117–145.

[15] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in Neural Inf. Proc. Systems* 28 (2015).

[16] Rishab Sharma, Ramin Shahbazi, Fatemeh H Fard, Zadia Codabux, and Melina Vidoni. 2022. SATD in R: Detection and Causes. *JASE* 29, 2 (2022), 1–41.

[17] Melina Vidoni. 2021. Evaluating unit testing practices in r packages. In *Int. Conf. on Software Engineering*. IEEE, 1523–1534.

[18] Melina Vidoni. 2021. Self-admitted technical debt in R packages: An exploratory study. In *Int. Conf. on Mining Software Repositories*. IEEE, South Korea, 179–189.

[19] Haiyin Zhang, Luís Cruz, and Arie van Deursen. 2022. Code Smells for Machine Learning Applications. In *Int. Conf. on AI Engineering: Software Engineering for AI* (Pittsburgh, Pennsylvania). ACM, USA, 217–228.