Is Attention All You Need? Toward a Conceptual Model for Social Awareness in Large Language Models

Gianmario Voria
gvoria@unisa.it
University of Salerno
Salerno, Italy

Gemma Catolino
gcatolino@unisa.it
University of Salerno
Salerno, Italy

Fabio Palomba
fpalomba@unisa.it
University of Salerno
Salerno, Italy

ABSTRACT

Large Language Models (LLMs) are revolutionizing the landscape of Artificial Intelligence (AI) due to recent technological breakthroughs. Their remarkable success in aiding various Software Engineering (SE) tasks through AI-powered tools and assistants has led to the integration of LLMs as active contributors within development teams, ushering in novel modes of communication and collaboration. However, great power comes with great responsibility: ensuring that these models meet fundamental ethical principles such as fairness is still an open challenge. In this light, our *vision paper* analyzes the existing body of knowledge to propose a conceptual model designed to frame ethical, social, and cultural considerations that researchers and practitioners should consider when defining, employing, and validating LLM-based approaches for software engineering tasks.

KEYWORDS

Social Awareness; Software Engineering for Artificial Intelligence; Large Language Models.

ACM Reference Format:
Gianmario Voria, Gemma Catolino, and Fabio Palomba. 2024. Is Attention All You Need? Toward a Conceptual Model for Social Awareness in Large Language Models. In *AI Foundation Models and Software Engineering (FORGE '24), April 14, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3650105.3652294

1 INTRODUCTION

Large Language Models (LLMs), advanced models designed to comprehend and produce language similarly to human speech based on the provided input, profoundly influence individuals' daily routines, including professionals such as software engineers. These models, recognized for their substantial size and intricacy, frequently encompass millions or billions of acquired details derived from extensive text data throughout the training process. [1–3].

In the **Software Engineering** (SE) field, the most evidence of this relies on the birth of GitHub's CoPilot in 2021[1], where AI finds practical applications in supporting SE tasks. It revolutionized the approach of software engineers to their work. [4]. Following this development, the research community began exploring the integration of LLMs into applications designed to enhance the daily tasks of developers, such as code generation [5, 6]. These experiments highlighted the possibility of introducing AI-powered members into the collaborative landscape [7, 8]. While recognizing their impressive potential, researchers and practitioners in SE approached the integration with a degree of skepticism [9]. This reserved stance reflected a careful consideration of the possible implications of incorporating LLMs into established practices within the field.

Indeed, *all that glitters is not gold.* LLMs are usually trained on uncured and unprocessed sources of existing data. As a reflection of our history, such data inherits both good and bad aspects and knowledge of humans. Thus, potential social biases in language models could derive from the training data gathered from human societies. Many incidents occurred—e.g., Facebook vision model that put the "primate" label to black men, Amazon assigning lower sales ranking to books containing gay themes, and racism and sexism during authentication processes [10]—and they all stress the urge to develop *ethical* AI.

Recent works explored causes and possible solutions to biases and unjust decisions of machine learning models. From a technical point of view, researchers proposed algorithmic solutions to achieve fairness [11]. In decision-making, fairness is *the absence of prejudice or favoritism toward an individual or group based on their inherent or acquired characteristics* [12–14]. Moreover, standards and processes have been defined for building ethical solutions in traditional systems, known as Value-based Engineering [15].

On the one hand, the data sources carry harmful information that mirrors biases, causing ML-based applications to internalize stereotypical behaviors [13]. Conversely, the imbalanced labeling of various demographic and protected groups introduces distributional disparities. This imbalance can result in unjust predictions when a model, trained under the homogeneity assumption, is applied to real-world inputs [16]. Furthermore, being developed by humans, even worse biases could be introduced by developers.

On the other hand, threats of introducing LLMs in the daily life of a software engineer have consistently been underestimated. Creating software is inherently a social activity, and how developers communicate impacts the project's success. Good or bad communication and collaboration patterns among team members have been widely investigated in the software engineering research area. Tamburri et al. defined community smells, i.e., a set of sub-optimal socio-technical characteristics and patterns in software development teams [17]. Unfortunately, the introduction of LLMs in developers' social activities has not yet been investigated.

[1]GitHub Copilot is an AI-powered code completion tool developed by GitHub in collaboration with OpenAI. It's designed to assist developers by providing context-aware code suggestions as they write code. Available at: https://github.com/features/copilot

FORGE '24, April 14, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0609-7/24/04
https://doi.org/10.1145/3650105.3652294

Based on all the above considerations and concerns, the *goal* of our vision is to highlight the necessity of considering **Social Awareness** in the future of Large Language Models in the Software Engineering scenario. Our conceptual exercise aims to integrate technical and socio-technical factors into a unified framework. Researchers and practitioners can leverage this framework to enhance the design of future Large Language Model (LLM)-based approaches for Software Engineering (SE) tasks, exploring their interactions with users in real-world environments.

2 BACKGROUND AND MOTIVATION

We define Large Language Models (LLMs) as a machine learning model designed to understand and generate human-like language. They fall under the Foundation Models (FMs) umbrella, i.e., a machine learning (ML) model pre-trained to perform various tasks. While LLM and FM are often used interchangeably, it is essential to point out that a Foundation Model is a broader concept. The former focuses on natural language tasks, so it is primarily used in software engineering scenarios. The latter, instead, can support various modalities, e.g., videos.

Although Generative Pre-trained Transformer (GPT) by OpenAI [18] is now the most popular LLM, the first step toward the definition of such models was made by Google in 2018. In particular, they introduced BERT (Bidirectional Encoder Representations from Transformers), a model pre-trained on vast datasets [3].

Following BERT, research on LLMs has overtaken the AI community, leading to the development of new solutions spread in many different contexts. The forefront of this trend is the Generative Pre-trained Transformer (GPT) by OpenAI, with its most recent update in GPT-4 [18], often used by SE researchers and practitioners in their daily tasks.

For example, Hou et al. [8] conducted a systematic literature review, uncovering 55 SE tasks that professionals automate using Language Model Models (LLMs). Furthermore, they explored various studies offering insights into using LLMs for software development, highlighting 21 tasks, e.g., code search.

Belzner et al. [7] envisioning the paradigm"LLM-assisted software engineering". This approach advocates for collaborative partnerships between LLMs and developers throughout all stages of development. In this model, LLMs act as development experts, and developers serve as domain experts, jointly contributing to requirements, software design, and code evaluation [7].

While previous studies emphasized the support provided by LLMs in daily tasks, researchers at the Center from Research on Foundation Models (CRFM)[2] identified opportunities and risks associated with applying these models in various fields. They particularly stressed the societal aspects, urging for responsible development and deployment. Despite the inherent complexity, the authors asserted that achieving fairness and ethics is feasible, encouraging researchers to strive for these goals [19].

To tackle such concerns, researchers have shifted their focus to specific unethical behaviors exhibited by LLMs. Wan et al. [20] proposed an automated framework to measure social bias in AI-based conversational assistants. Also, recent research has brought attention to tackling the problem of gender bias in LLMs. On the one side, Kotek et al. [21] tested LLMs to evaluate the fairness of their responses related to gender-related aspects. Their findings revealed biased assumptions concerning men and women, influenced by subjective opinions rather than factual evidence. Additionally, the study delved into the justifications provided by the models for their decisions, highlighting the importance of rigorous testing before their release. On the other side, Truede et al. [22] found notable gender bias in software development tasks like issue assignment, highlighting the need for improved training in LLMs.

The study's results above underscored the necessity for guidelines in using or developing LLMs. Spiekermann et al. [15] formulated rules and considerations for ethically designed solutions. They introduced the concept of Value-based Engineering for ethics by design, emphasizing ethical and moral values in software system design. This concept aligns with the IEEE P7000 standard[3], defining Value-based Engineering to address ethical issues across the software life cycle. However, this work primarily focuses on traditional systems, leaving a huge gap that needs addressing for AI and LLM-based applications. Concerning AI-based systems, Lu et al. [23] proposed a roadmap for Software Engineering for Responsible AI. The authors focus on enhancing responsible AI application development through multi-level governance establishment, integration of process-oriented best practices, and implementation of responsible-AI-by-design principles into system-level architectural styles, patterns, and techniques. Moreover, they presented several challenges that need to be addressed for ML systems, and they serve as a starting point for our reflections in the context of more complex ML-based systems.

Motivation. Considering the background information outlined above, our vision's *motivation* leverages the gaps and issues identified. Due to their socio-technical nature, LLMs are transforming SE, with AI-based assistants becoming essential contributors to development communities. Unfortunately, being trained on historical data, they inherit the bad and the good parts of our history, highlighting the urge to work toward defining ethical and moral solutions when using and developing them. As part of a *social* environment, we must consider their *awareness* of other individuals when designing and using such applications.

3 SOCIAL AWARENESS: A VISION

LLMs are socio-technical by nature: their applications, such as ChatGPT or CoPilot, aim to collaborate with users and support them in their tasks, being active actors in their activities and becoming community members. Communities' issues could easily lead to the failure of a project [24]. As LLM-based assistants play an increasingly functional role in the daily activities of software engineers, there is a need to regulate their social and ethical behaviors, thereby introducing the concept of *social awareness* in LLMs.

> **Social Awareness in Large Language Models**
>
> *The understanding and consideration of ethical, social, and cultural factors in the development and use of Large Language Models by those who develop and use them.*

[2] https://crfm.stanford.edu/

[3] https://standards.ieee.org/ieee/7000/6781/

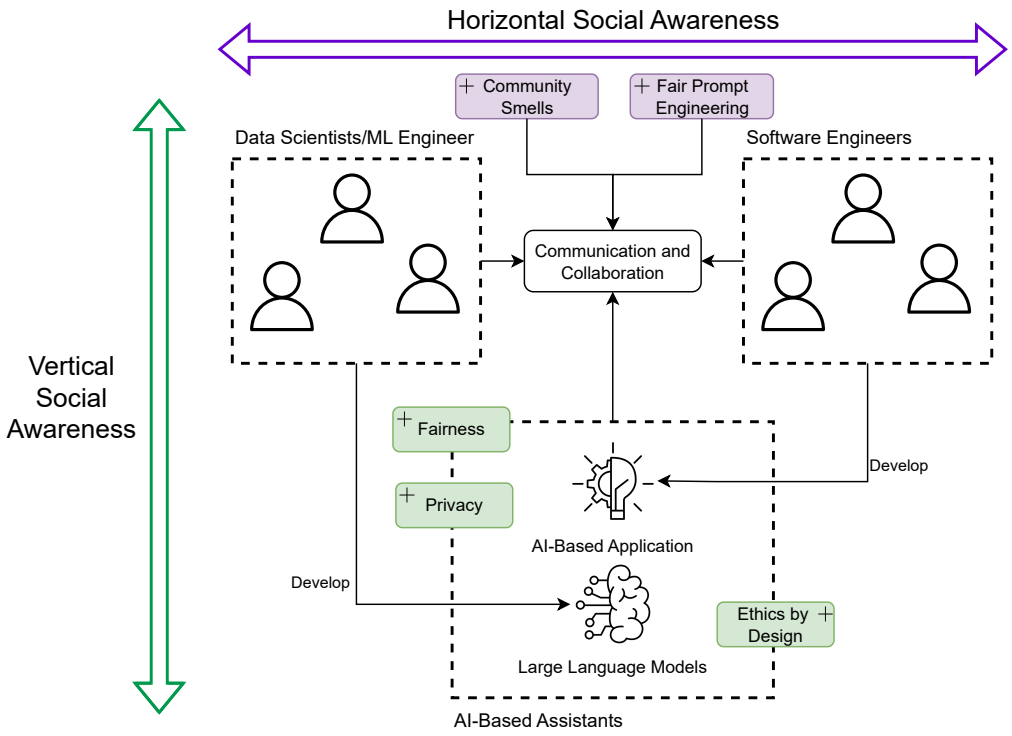

Figure 1: Overview of our Vision of Social Awareness in Large Language Models

The following sections will further detail our *vision* in its fundamental parts, which are also summarized in Figure 1. Building on the concept introduced by Belzner et al. [7], we illustrate collaborative partnerships between development teams and LLM-based assistants as integral components of the environment. In light of this scenario, we aim to highlight specific concerns and challenges.

We build on top of existing literature [15, 19, 21, 22] in different contexts to define two distinct **catalogs of practices and challenges** to apply as software engineers when designing or using LLM-based applications, namely *Vertical* and *Horizontal* Social Awareness. The former should be seen from the developers' perspective, who should take technical and moral steps toward designing ethical solutions. The latter should be seen from the perspective of the same developers but in their use and application of such systems, considering ethical and social concerns in their activities.

3.1 Vertical Social Awareness

LLMs simulate human language interactions, attaining performance through training on large datasets and fine-tuning for specialized tasks. Advances in deep learning and AI techniques like transformers, attention mechanisms [25], transfer learning [26], and continuous learning [27] make these processes possible. Unfortunately, these tasks are often labor-intensive, prone to mistakes, and, of all the errors, the least palpable yet most troublesome relate to *ethics*, *fairness*, and *privacy*. Ensuring that systems align with ethical principles, such as those outlined in the European Artificial Intelligence Act[4], and avoiding discrimination against vulnerable groups poses a complex challenge for developers.

In light of all the challenges outlined, we propose a set of considerations that software engineers should analyze when developing LLM-based applications.

Fairness. Although explored and studied in artificial intelligence and software engineering, achieving fairness is still an open challenge [12]. However, when considering big and complex systems like LLMs, the challenge is even more complicated, as shown by works that analyzed particular bias issues, i.e., gender [21, 22]. Given their potential impact on critical areas, e.g., e-health, loan assignment, or hiring decisions, tasks performed by LLM-based assistants in software engineering demand heightened attention to avoid the potential harm of unfair solutions. LLMs are typically trained on general data and fine-tuned on specific tasks using specific datasets. Consequently, fairness solutions in this context must consider various characteristics intrinsic to the field of application.

Privacy. As LLMs extensively train on diverse data sources, they can recall much of the information used during training or fine-tuning. However, this data is not necessarily intended for reproduction as it often contains private information. Therefore, it is crucial to consider privacy-preserving solutions and techniques [28], including approaches like Differential Privacy Decoding proposed by Mujmudar et al. [29]. Established privacy techniques from

[4]The AI Act is a proposed European law on artificial intelligence (AI). The law categorizes AI applications into three risk levels. Available at: https://artificialintelligenceact.eu/

the machine learning field, such as cryptography [30], should be tested and evaluated. We advocate for multidisciplinary approaches, combining software engineering and security considerations, to effectively address privacy issues in LLM-based applications.

Ethics by Design. Considering the Value-based Engineering for Ethics By Design framework proposed for the software engineering area [15], together with the IEEE P7000 standard, we suggest that researchers should work toward the definition of specific practices for the development of LLMs and LLM-based systems. The framework proposed involves two main phases: *Ethical Exploration* and *Ethically Aligned Design*. The former encompasses processes like value elicitation, prioritization, and identification of ethical value quality. The latter focuses on integrating these values into both technical and organizational aspects of system design, employing iterative and risk assessment-based approaches to address values at risk in system architecture and design. Defining specific rules for LLM-based applications is crucial due to their unique sociotechnical nature and the concerns surrounding their training data. These applications, which serve as constant user assistants and interact closely with developers, require careful consideration of various ethical concerns and values throughout the design process.

3.2 Horizontal Social Awareness

LLM-assisted software engineering envisions collaborations between software engineers and LLM-based assistants throughout all stages of software development [7]. AI becomes an integral part of the working environment in this new paradigm, supporting developers in their daily tasks, much like a colleague. However, playing such a crucial role comes with responsibilities. Beyond mastering the art of building fair LLMs, there is a need to learn how to use them, considering ethical, social, and cultural factors. Building upon the reflections in this paper, we propose a set of aspects that should be further explored concerning the interaction with LLMs.

Fair Prompt Engineering. Prompt engineering generally refers to the practice of carefully crafting or designing prompts to obtain desired responses from language models or other artificial intelligence systems. Indeed, correctly prompting LLMs is critical to gathering valuable resources, making this aspect central when considering the social activities in software development teams that include LLM-based assistants. When implemented, which is the case for the most famous LLMs, continuous learning [27] can use these prompts to keep training and improve the model's performances, getting better and better as the assistant gets asked for help. To avoid future discrimination or biases from models, we should carefully evaluate what we ask and how we do it. For instance, providing sensitive data could end up creating a privacy issue. Moreover, properly communicating with LLMs—fair prompts— will also lead to obtaining more socially aware answers. Thus, we suggest researchers and practitioners start understanding the function of prompt engineering in the context of ethical and social concerns.

Reevaluating Community Smells. As mentioned at the paragraph's outset, treating LMMs as integral team members prompts the exploration of their interactions when communicating or collaborating with human team members, including potential associated issues. Being trained on open-source data, issues like toxicity [31] could be inherited, possibly leading to unhealthy working environments. In this context, the concept of Community Smells [32]—suboptimal communication and collaboration patterns in software development communities, known to contribute to social debt—may warrant reevaluation. Literature on community smells [33] identifies patterns related to team member interactions, expectations, and perceptions of others' work, making them pertinent for reconsidering AI-human collaborations. For example, the *Cognitive Distance* smell, which refers to the distance that peers perceive on the cultural, physical, social, or technical level, is mainly caused by the experience diversity [17]. However, *how do developers perceive such a difference with LLMs?* Not trusting team members' solutions may cause the *Dissensus* smell, i.e., the inability to achieve consensus on solutions and being unable to proceed with the project [34]. *Would this happen with LLM assistants as team members?* The immediacy of solutions provided by assistants could result in a *Disengagement* smell [17], i.e., loss of engagement and curiosity by team members. Given such considerations, we suggest researchers start working toward understanding social patterns in teams that include LLMs.

4 CONCLUSION

This paper reports our vision of the future of Large Language Models in Software Engineering, specifically focusing on social and ethical aspects. We introduced the concept of *Social Awareness* for LLMs, i.e., the understanding and consideration of ethical, social, and cultural factors in the development and use of LLMs by those who develop and use them. Furthermore, we proposed two critical aspects: (1) vertical and (2) horizontal social awareness. The former suggests that developers and researchers pay attention to ethical and privacy concerns in the design and development of LLMs, highlighting the necessity of processes and rules to grant ethics by design. The latter focuses on the social and ethical consequences of introducing LLM-based assistants in collaborative scenarios like software engineering.

With this vision, we hope practitioners and researchers reevaluate the following three key aspects. Firstly, the definition and implementation of LLM-based approaches should consider social awareness in their design. Secondly, communication and collaboration patterns should be adapted to a new scenario involving LLMs as active software engineering community members. Finally, from the researchers' perspective in Software Engineering for Artificial Intelligence, evaluating such approaches needs reconsideration, focusing on analyzing the validity considering social awareness.

5 ACKNOWLEDGMENT

This work has been partially supported by the European Union - NextGenerationEU through the Italian Ministry of University and Research, Project PRIN 2022 PNRR "FRINGE: context-aware FaiRness engineerING in complex software systEms" (grant n. P2022553SL, CUP: D53D23017340001). The opinions presented in this article solely belong to the author(s) and do not necessarily reflect those of the European Union or The European Research Executive Agency. The European Union and the granting authority cannot be held accountable for these views.

REFERENCES

[1] T. B. B. et al., "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, "Github copilot ai pair programmer: Asset or liability?" *Journal of Systems and Software*, vol. 203, p. 111734, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223001292

[5] M. C. et al., "Evaluating large language models trained on code," 2021.

[6] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "Codebert: A pre-trained model for programming and natural languages," 2020.

[7] L. Belzner, T. Gabor, and M. Wirsing, *Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study*, 12 2023, pp. 355–374.

[8] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," 2023.

[9] M. Welsh, "The end of programming," *Commun. ACM*, vol. 66, no. 1, p. 34–35, dec 2022. [Online]. Available: https://doi.org/10.1145/3570220

[10] M. Wei and Z. Zhou, "Ai ethics issues in real world: Evidence from ai incident database," 2022.

[11] M. C. W. Alina Köchling, "Discriminated by an algorithm: a systematic review of discrimination and fairness by algorithmic decision-making in the context of hr recruitment and hr development," 2020.

[12] D. Pessach and E. Shmueli, "A review on fairness in machine learning," *ACM Comput. Surv.*, vol. 55, no. 3, feb 2022. [Online]. Available: https://doi.org/10.1145/3494672

[13] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surv.*, vol. 54, no. 6, jul 2021. [Online]. Available: https://doi.org/10.1145/3457607

[14] C. Starke, J. Baleis, B. Keller, and F. Marcinkowski, "Fairness perceptions of algorithmic decision-making: A systematic review of the empirical literature," 2021.

[15] S. Spiekermann and T. Winkler, "Value-based engineering for ethics by design," *CoRR*, vol. abs/2004.13676, 2020. [Online]. Available: https://arxiv.org/abs/2004.13676

[16] D. Shah, H. A. Schwartz, and D. Hovy, "Predictive biases in natural language processing models: A conceptual framework and overview," *CoRR*, vol. abs/1912.11078, 2019. [Online]. Available: http://arxiv.org/abs/1912.11078

[17] D. A. Tamburri, R. Kazman, and H. Fahimi, "The architect's role in community shepherding," *IEEE Software*, vol. 33, no. 6, pp. 70–79, 2016.

[18] O. et al., "Gpt-4 technical report," 2023.

[19] B. R. et al., "On the opportunities and risks of foundation models," *ArXiv*, 2021. [Online]. Available: https://crfm.stanford.edu/assets/report.pdf

[20] Y. Wan, W. Wang, P. He, J. Gu, H. Bai, and M. Lyu, "Biasasker: Measuring the bias in conversational ai system," 2023.

[21] H. Kotek, R. Dockum, and D. Sun, "Gender bias and stereotypes in large language models," in *Proceedings of The ACM Collective Intelligence Conference*, ser. CI '23. ACM, Nov. 2023. [Online]. Available: http://dx.doi.org/10.1145/3582269.3615599

[22] C. Treude and H. Hata, "She elicits requirements and he tests: Software engineering gender bias in large language models," 2023.

[23] Q. Lu, L. Zhu, X. Xu, J. Whittle, and Z. Xing, "Towards a roadmap on software engineering for responsible ai," 2022.

[24] A. Hannemann, H.-J. Happel, M. Jarke, R. Klamma, S. Lohmann, W. Maalej, and V. Wulf, "Social aspects in software engineering," *Software Engineering 2009-Workshopband*, 2009.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[26] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[27] C. Käding, E. Rodner, A. Freytag, and J. Denzler, "Fine-tuning deep neural networks in continuous learning scenarios," in *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III 13*. Springer, 2017, pp. 588–605.

[28] C. Peris, C. Dupuy, J. Majmudar, R. Parikh, S. Smaili, R. Zemel, and R. Gupta, "Privacy in the time of language models," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1291–1292. [Online]. Available: https://doi.org/10.1145/3539597.3575792

[29] J. Majmudar, C. Dupuy, C. Peris, S. Smaili, R. Gupta, and R. Zemel, "Differentially private decoding in large language models," 2022.

[30] C. Rechberger and R. Walch, "Privacy-preserving machine learning using cryptography," in *Security and Artificial Intelligence: A Crossdisciplinary Approach*. Springer, 2022, pp. 109–129.

[31] C. Miller, S. Cohen, D. Klug, B. Vasilescu, and C. Kästner, ""did you miss my comment or what?" understanding toxicity in open source discussions," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 710–722.

[32] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "What is social debt in software engineering?" in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 93–96.

[33] E. Caballero-Espinosa, J. C. Carver, and K. Stowers, "Community smells—the sources of social debt: A systematic literature review," *Information and Software Technology*, vol. 153, p. 107078, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922001872

[34] F. Palomba, D. A. Tamburri, F. A. Fontana, R. Oliveto, A. Zaidman, and A. Serebrenik, "Beyond technical aspects: How do community smells influence the intensity of code smells?" *IEEE transactions on software engineering*, vol. 47, no. 1, pp. 108–129, 2018.