Towards Safe, Secure, and Usable LLMs4Code

Al-Kaswan, Ali

DOI
10.1145/3639478.3639803

Publication date
2024

Document Version
Final published version

Published in
Proceedings - 2024 ACM/IEEE 46th International Conference on Software Engineering

Towards Safe, Secure, and Usable LLMs4Code

Ali Al-Kaswan
a.al-kaswan@tudelft.nl
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Large Language Models (LLMs) are gaining popularity in the field of Natural Language Processing (NLP) due to their remarkable accuracy in various NLP tasks. LLMs designed for coding are trained on massive datasets, which enables them to learn the structure and syntax of programming languages. These datasets are scraped from the web and LLMs memorise information in these datasets. LLMs for code are also growing, making them more challenging to execute and making users increasingly reliant on external infrastructure. We aim to explore the challenges faced by LLMs for code and propose techniques to measure and prevent memorisation. Additionally, we suggest methods to compress models and run them locally on consumer hardware.

CCS CONCEPTS

• **Security and privacy**; • **Software and its engineering**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

large language models, privacy, memorisation, data leakage, compression

ACM Reference Format:
Ali Al-Kaswan. 2024. Towards Safe, Secure, and Usable LLMs4Code. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3639478.3639803

1 INTRODUCTION

In recent years, Large Language Models (LLMs) have become increasingly popular in Natural Language Processing (NLP) due to their impressive accuracy in a wide range of NLP tasks [1]. As the number of parameters in these models increases from millions to billions, their accuracy and capabilities also improve [2]. LLMs designed for coding (LLMs4Code) are trained on large datasets and can learn the structure and syntax of programming languages. As a result, they are highly proficient in tasks such as generating [3], summarising [4], and completing code [5].

The appeal of scaling up LLMs is the discovery of emergent capabilities [6]. Emergent capabilities cannot be anticipated by extrapolating scaling laws and only become visible at a certain critical model size threshold [6]. This encourages the training of ever-larger models, as abilities such as chain-of-thought prompting [7] and instruction tuning [8] can only be achieved in models with more than 100B parameters [6]. However, this increase in parameter counts makes it increasingly difficult to deploy and run LLMs. Many state-of-the-art open source LLMs4Code such as CodeLlama [9] and WizardCoder [10] cannot be executed on consumer GPUs with less than 32GB of VRAM[1].

This excludes many from being able to use current state-of-the-art LLMs. Those who cannot afford the hardware to deploy the models must rely on external services to run the models, such as GitHub Copilot. From a privacy and security perspective, this is not always desirable. Firstly, the source code might contain all types of information about the developer, which is then sent to an external party. Secondly, some organisations do not allow their proprietary source code to leave their premises.

It has also been observed that large language models trained on natural language can memorise and regurgitate training data [11–22]. This issue has not been fully explored for code. Moreover, the issue of memorisation in the source code is different from that of natural language.

The open source code used in LLM training for code is often licenced under non-permissive copy-left licences, such as GPL or the CC-BY-SA licence employed by StackOverflow [11]. Reusing code covered by these licences without making the source code available under the same licence is considered a violation of copyright law. In some jurisdictions, this leaves users of tools such as CoPilot at legal risk [11, 15, 23]. Sharing code without proper licences is also ethically questionable [11, 15, 16].

Memorised data can also include confidential information [24–26], which can include credentials, API keys, emails, and other sensitive data [11, 27]. This means that memorisation could put the private information contained in the training data at risk. Recently, attacks which exploit memorisation have been able to extract (or reconstruct) training data from LLMs [19, 22, 24, 28]. The US National Institute of Standards and Technology (NIST) considers data reconstruction attacks to be the most serious type of privacy attack against machine learning models [29]. OWASP classifies Sensitive Information Disclosure (LLM06) as the sixth most critical vulnerability in LLM applications. [2]

We propose an approach to measure the rate at which memorisation occurs in LLMs4Code. We then measure the rate at which memorisation occurs for PII and copyrighted code. These findings will then be used to inform dataset construction and model training techniques to prevent memorisation. In parallel, we will also investigate approaches to compress LLMs4Code and the impact of compression on memorisation.

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04.
https://doi.org/10.1145/3639478.3639803

[1] Can you run it? LLM version: https://huggingface.co/spaces/Vokturz/can-it-run-llm
[2] OWASP Top 10 for Large Language Model Applications: https://owasp.org/www-project-top-10-for-large-language-model-applications/

2 BACKGROUND AND RELATED WORK

2.1 Memorisation

Memorisation in language models is the capacity to remember and recall details of the data it has been trained on. This happens when the model is too specific and does not generalise well to new or unseen data [20, 30]. As a result, the model can accurately reproduce phrases, sentences, or even entire documents from the training data. Apart from the privacy issues discussed in section 1, memorisation also leads to an overestimation of performance. For example, CodeX has been observed to be able to solve HackerRank problems without receiving the full task description [18].

Memorisation can lead to high accuracy, but it does not necessarily mean that the model will generalise well to new or unseen data. This can lead to poor performance in real-world applications. Furthermore, memorisation can reduce the model's ability to adjust its output to particular use cases. For instance, when slightly altering HackerRank problems, CodeX [31] has difficulty producing the correct solution, instead repeating the answer for the original problem [18, 32].

2.2 Data Extraction Attacks

Data extraction attacks are a type of attack in which an adversary extracts a data point from the training data of a model. Attacks can be divided into two types for LLMs, namely guided and unguided attacks [28].

In an unguided attack, the adversary does not know the sample to be extracted from the model. The adversary simply attempts to extract any training point, contained anywhere in the training corpus [14, 24, 25, 33]. Targeted attacks are more security-critical as they allow the targeting of specific information, such as the extraction of emails [15, 25, 28, 34, 35].

2.3 Model Compression

Model Compression for LLMs can roughly be divided into three techniques. Namely, knowledge distillation, pruning, and quantisation [36, 37].

Knowledge distillation transfers the knowledge of the large teacher model to a smaller and simpler student model [36]. Pruning reduces the size of the model by removing unneeded parameters [36, 38, 39]. Quantisation is a relatively simple technique that reduces the precision of the model by reducing floating point numbers to integers or smaller representations [36, 40, 41].

A number of methods, including XTC [37], have been developed to combine multiple techniques to achieve a higher compression rate. While these hybrid approaches have been used to compress models from the natural language domain, their application to software models has yet to be fully explored.

3 APPROACH

First, we explore the different risks and implications posed by LLMs4Code. In our position paper, we map the existing privacy problems in LLMs to the source code domain. We also identify other code-specific issues, namely licencing and security [11].

3.1 Memorisation

To measure memorisation, we create a set of potentially extractable samples for a given model using a targeted data extraction attack. The process of finding memorised data is relatively simple [28], by changing the number of input tokens, we can change the difficulty of the sample, which in turn allows us to compare the rate between different models and prompting techniques [3]. This work has already been completed and was accepted into the main ICSE track.

Using this framework for measurement, we can extend the evaluation to also look at specific types of data. Using techniques like those described by Niu et al. [42], we can identify code that contains PII and use it as input to our evaluation. We can similarly extend our evaluation to include copyrighted code as well.

Based on these findings, we can identify patterns that elicit memorisation in LLMs4Code and can put the user at risk. These patterns can then be used to design datasets and training regimes that reduce the memorisation rate.

3.2 Compression

Finally, for the Model Compression, we plan to adapt different techniques for compression from the natural language domain to code. We measure the parameter count, disk size, size in VRAM, inference time and accuracy for each given model and compression technique. We further investigate the impact of compressing the LLMs on the rate of memorisation, and the relation between overparametrisation and memorisation.

4 EXPECTED CONTRIBUTIONS

As Large Language Models for Code (LLMs4Code) continue to gain widespread adoption, our research aims to enhance their usability and instil trust among users. By developing robust techniques for measuring memorisation in LLMs4Code, we empower users with the knowledge to make well-informed decisions regarding the models they choose to employ.

Our research contributes to the evolution of LLMs4Code by addressing concerns related to memorisation, thereby reducing the likelihood and associated risks of unintended memorisation in model outputs. This proactive approach ensures that users can have confidence in the reliability and generalisation capabilities of the models they rely on, fostering a more secure and dependable ecosystem for utilising LLMs4Code.

Moreover, our work on compressing LLMs is a significant step towards democratizing access to these powerful tools by substantially reducing the hardware requirements traditionally associated with their deployment. Our efforts will make LLMs4Code more accessible to a broader audience, paving the way for wider adoption and greater participation, which would enable more individuals to benefit from the use of LLMs4Code.

REFERENCES

[1] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, "Codexglue: A machine learning benchmark dataset for code understanding and generation," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[3] Language Models Training Data Extraction Challenge: https://github.com/google-research/lm-extraction-benchmark

[2] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," in *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 2022, pp. 1–10.

[3] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih, L. Zettlemoyer, and M. Lewis, "Incoder: A generative model for code infilling and synthesis," *preprint arXiv:2204.05999*, 2022.

[4] A. Al-Kaswan, T. Ahmed, M. Izadi, A. A. Sawant, P. Devanbu, and A. van Deursen, "Extending source code pre-trained language models to summarise decompiled binaries," in *Proceedings of the 30th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2023.

[5] M. Izadi, R. Gismondi, and G. Gousios, "Codefill: Multi-token code completion by jointly learning from structure and naming sequences," in *Proceedings of the 44th International Conference on Software Engineering (ICSE)*. ACM, 2022, p. 401–412.

[6] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[8] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[9] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve, "Code llama: Open foundation models for code," 2023.

[10] Z. Luo, C. Xu, P. Zhao, Q. Sun, X. Geng, W. Hu, C. Tao, J. Ma, Q. Lin, and D. Jiang, "Wizardcoder: Empowering code large language models with evol-instruct," 2023.

[11] A. Al-Kaswan and M. Izadi, "The (ab)use of open source code to train large language models," in *Proceedings of the 2nd International Workshop on Natural Language-based Software Engineering (NLBSE)*, 2023.

[12] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1897–1914.

[13] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying memorization across neural language models," *preprint arXiv:2202.07646*, 2022.

[14] N. Carlini, M. Jagielski, C. Zhang, N. Papernot, A. Terzis, and F. Tramer, "The privacy onion effect: Memorization is relative," *Advances in Neural Information Processing Systems*, vol. 35, pp. 13 263–13 276, 2022.

[15] P. Henderson, X. Li, D. Jurafsky, T. Hashimoto, M. A. Lemley, and P. Liang, "Foundation models and fair use," *arXiv preprint arXiv:2303.15715*, 2023.

[16] Z. Sun, X. Du, F. Song, M. Ni, and L. Li, "Coprotector: Protect open-source code against unauthorized training usage with data poisoning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 652–660.

[17] K. Tirumala, A. Markosyan, L. Zettlemoyer, and A. Aghajanyan, "Memorization without overfitting: Analyzing the training dynamics of large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 38 274–38 290, 2022.

[18] A. Karmakar, J. A. Prenner, M. D'Ambros, and R. Robbes, "Codex hacks hackerrank: Memorization issues and a framework for code synthesis evaluation," *arXiv preprint arXiv:2212.02684*, 2022.

[19] S. Biderman, U. S. Prashanth, L. Sutawika, H. Schoelkopf, Q. Anthony, S. Purohit, and E. Raf, "Emergent and predictable memorization in large language models," *arXiv preprint arXiv:2304.11158*, 2023.

[20] V. Feldman, "Does learning require memorization? a short tale about a long tail," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 954–959.

[21] N. Lukas, A. Salem, R. Sim, S. Tople, L. Wutschitz, and S. Zanella-Béguelin, "Analyzing leakage of personally identifiable information in language models," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2023, pp. 346–363.

[22] S. Ishihara, "Training data extraction from pre-trained language models: A survey," 2023.

[23] M. Z. Choksi and D. Goedicke, "Whose text is it anyway? exploring bigcode, intellectual property, and ethics," *ArXiv*, vol. abs/2304.02839, 2023.

[24] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

[25] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," *arXiv preprint arXiv:2301.13188*, 2023.

[26] D. Ippolito, F. Tramèr, M. Nasr, C. Zhang, M. Jagielski, K. Lee, C. A. Choquette-Choo, and N. Carlini, "Preventing verbatim memorization in language models gives a false sense of privacy," *arXiv preprint arXiv:2210.17546*, 2022.

[27] S. K. Basak, L. Neil, B. Reaves, and L. Williams, "Secretbench: A dataset of software secrets," *arXiv preprint arXiv:2303.06729*, 2023.

[28] A. Al-Kaswan, M. Izadi, and A. van Deursen, "Targeted attack on gpt-neo for the satml language model data extraction challenge," *ArXiv*, vol. abs/2302.07735, 2023.

[29] A. Oprea and A. Vassilev, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," National Institute of Standards and Technology, Tech. Rep., 2023.

[30] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 1964–1974. [Online]. Available: https://proceedings.mlr.press/v139/choquette-choo21a.html

[31] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," 2021.

[32] J. Tan, D. LeJeune, B. Mason, H. Javadi, and R. G. Baraniuk, "A blessing of dimensionality in membership inference through regularization," in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, F. Ruiz, J. Dy, and J.-W. van de Meent, Eds., vol. 206. PMLR, 25–27 Apr 2023, pp. 10 968–10 993. [Online]. Available: https://proceedings.mlr.press/v206/tan23b.html

[33] M. G. Oh, L. H. Park, J. Kim, J. Park, and T. Kwon, "Membership inference attacks with token-level deduplication on korean language models," *IEEE Access*, vol. 11, pp. 10 207–10 217, 2023.

[34] J. Huang, H. Shao, and K. C.-C. Chang, "Are large pre-trained language models leaking your personal information?" in *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2038–2047. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.148

[35] F. Mireshghallah, K. Goyal, A. Uniyal, T. Berg-Kirkpatrick, and R. Shokri, "Quantifying privacy risks of masked language models using membership inference attacks," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 8332–8347. [Online]. Available: https://aclanthology.org/2022.emnlp-main.570

[36] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, "A survey on model compression for large language models," *arXiv preprint arXiv:2308.07633*, 2023.

[37] X. Wu, Z. Yao, M. Zhang, C. Li, and Y. He, "Xtc: Extreme compression for pre-trained transformers made simple and efficient," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3217–3231, 2022.

[38] M. Xia, Z. Zhong, and D. Chen, "Structured Pruning Learns Compact and Accurate Models," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1513–1528. [Online]. Available: https://aclanthology.org/2022.acl-long.107

[39] Z. Wang, J. Wohlwend, and T. Lei, "Structured Pruning of Large Language Models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6151–6162, arXiv:1910.04732 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1910.04732

[40] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers," Oct. 2022. [Online]. Available: https://openreview.net/forum?id=f-fVCElZ-G1

[41] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8815–8821, Apr. 2020, number: 05. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6409

[42] L. Niu, S. Mirza, Z. Maradni, and C. Pöpper, "CodexLeaks: Privacy leaks from code generation language models in GitHub copilot," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 2133–2150. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/niu