An Exploratory Study Into Polarized Augment Calibration for Membership Inference in Code LLMs

Roham Koohestani

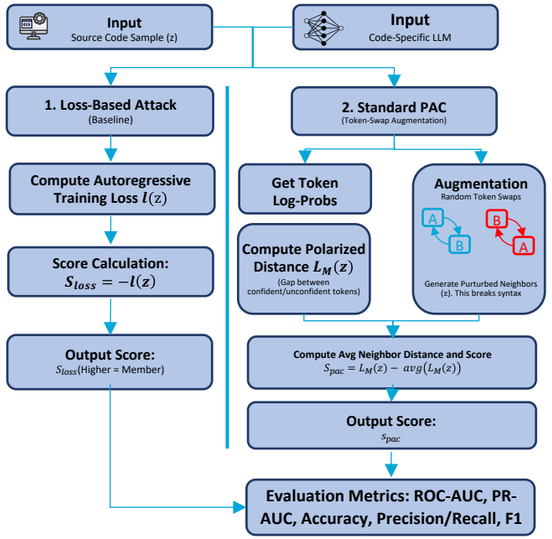
Motivation

- LLMs trained on **large-scale data** raise concerns regarding **privacy**, **copyright**, and **verbatim memorization**.
- Detecting whether copyrighted or license-restricted data **appears in training sets** requires technical auditing tools.
- Membership Inference Attacks (**MIAs**) offer a **scalable** alternative to expensive Training Data Extraction.
- Existing white-box **MIAs lack comparative evaluation**, especially on **code LLMs**.

Research Questions

- **RQ1:** Compare attack performance of Loss vs PAC on code LLMs.
- **RQ2:** Measure robustness across data characteristics (size, AST complexity, alphanumeric ratio).
- **RQ3:** Introduce and evaluate a code-specific adaptation of PAC using AST-aware augmentation.

Approaches



Dataset & Experimental Setup

Dataset

- Java Subset of **The Heap**
- Three Groups: **Members**, **near-members**, **non-members**
- **Stratified** Sampling across **code size**, **AST complexity**, **alphanumeric ratio** and **membership type**

Models Evaluated

- **StarCoder2-3b**
- **Mellum-4B-base**
- **SmolLm3-3B**

Evaluation Metrics

ROC-AUC, PR-AUC, Standard Classification Metrics

Comparing Attacks

Attack	Model	Scenario	ROC-AUC	PR-AUC	Acc.	Prec.	Rec.	F1
loss	Mellum	MA vs. NM	0.628	0.764	0.565	0.762	0.497	0.602
		ME vs. NM	0.629	0.628	0.598	0.627	0.480	0.544
		MN vs. NM	0.626	0.617	0.601	0.611	0.499	0.549
	SmolLM	MA vs. NM	0.573	0.701	0.595	0.705	0.666	0.685
		ME vs. NM	0.572	0.539	0.561	0.557	0.585	0.571
		MN vs. NM	0.575	0.541	0.560	0.540	0.668	0.597
	StarCoder2	MA vs. NM	0.625	0.750	0.580	0.750	0.546	0.632
		ME vs. NM	0.622	0.600	0.596	0.601	0.562	0.581
		MN vs. NM	0.629	0.605	0.600	0.604	0.519	0.558
pac	Mellum	MA vs. NM	0.696	**0.827**	0.613	0.807	0.545	0.651
		ME vs. NM	**0.712**	0.742	0.662	0.730	0.511	0.601
		MN vs. NM	0.679	0.691	0.634	0.660	0.516	0.579
	SmolLM	MA vs. NM	0.578	**0.735**	0.517	0.738	0.418	0.533
		ME vs. NM	0.579	0.591	0.566	0.588	0.431	0.498
		MN vs. NM	**0.578**	0.579	0.568	0.581	0.414	0.483
	StarCoder2	MA vs. NM	**0.662**	**0.798**	0.585	0.791	0.505	0.617
		ME vs. NM	**0.662**	0.680	0.624	0.654	0.522	0.581
		MN vs. NM	0.661	0.664	0.625	0.649	0.504	0.567

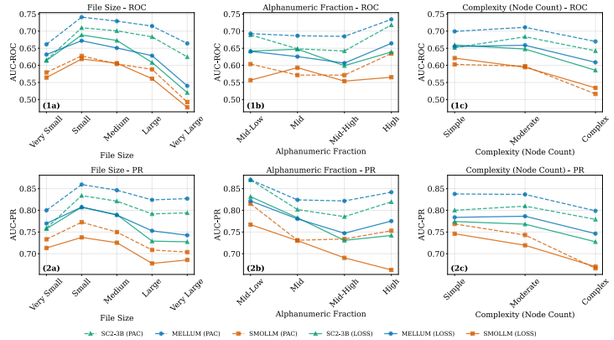
- **Modifications Offer No Protection:** Near-duplicate code is detected just as easily as exact copies.
- **PAC Outperforms Loss:** Calibrated attacks reveal deeper privacy leakage than standard baselines.
- **Code Models Are Leakier:** Specialized models (e.g., StarCoder2, and Mellum) are more vulnerable to attacks than general-purpose LLMs.
- **High Precision:** The PAC attack achieves nearly 0.80 PR-AUC, providing a reliable auditing signal

Finding the Limitation

Complexity Hinders Detection: Attack performance drops as code becomes "Complex" or "Very Large". Likely due to the structural rigidness of code.

The "Sweet Spot": The attacks are most effective on "Small" to "Medium" sized files, peaking before performance degrades on larger files.

Alphanumeric Sensitivity: Low and high alphanumeric density has a positive impact on the predictive power of the attacks. Likely due to high entropy in these types of data (e.g., textual or numeric data carriers consistent with literature)

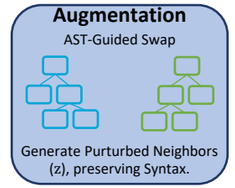


Fixing the Limitation

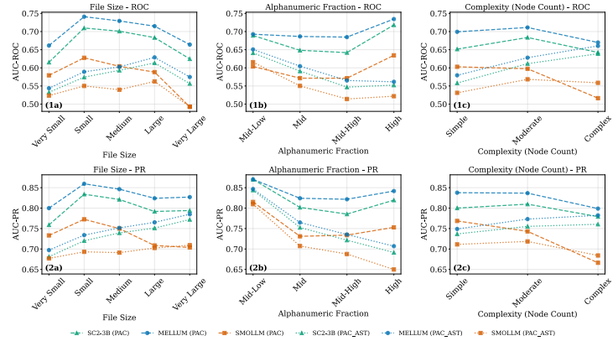
Structure-Aware Calibration Helps at Scale:
AST-guided augmentation improves the stability of PAC on Large and Complex code, where token-swap calibration degrades. This indicates that syntactic validity becomes increasingly important as program structure grows.

Under-Mutation Limits Small-Code Performance:
PAC-AST underperforms on Small and Simple files due to a substantially lower effective perturbation magnitude. As a result, calibration neighbors are insufficiently separated from the original sample in these regimes.

Lexical Information Remains Critical:
PAC-AST degrades on alphanumeric-rich code, in contrast to token-swap PAC. This suggests that purely syntactic perturbations fail to disrupt lexical cues such as identifiers and comments that are predictive for membership.



Main Idea: Swap syntactically equivalent pieces of code to minimize the disruption in the flow of the original piece of code.



Remaining Limitations / Future Work

PAC and PAC-AST differ in effective perturbation magnitude, confounding augmentation strategy with calibration strength. Future work should enforce token-level perturbation budgets or adaptive mutation schemes.

PAC-AST's weakness on small and alphanumeric-rich code motivates hybrid syntactic and lexical neighbors.

References

- Al-Kaswan et. al. Traces of memorization in large language models for code. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pages 1–12, 2024.
- Bakouch et. al. Smollm3: smol, multilingual, long-context reasoner https://huggingface.co/blog/smollm3, 2025
- Carlini et. al., Extracting training data from large language models. In the 30th USENIX security symposium, pages 2633-2650, 2021
- Katzy et. al., the heap: A contamination-free multilingual code dataset for evaluating large language models, 2025
- Lozhkov et. al., Starcoder 2 and the Stack V: The next generation. arXiv preprint arXiv:2402:19173, 2024
- Pavlichenko et. al., Mellum: Production-grade in-ide contextual code completion with multi-file project understanding. arXiv preprint arXiv:2510.05788, 2025
- Shokri et. al., Membership inference attacks against machine learning models. In 2017 IEEE symposium on security and privacy (SP), pages 3-18. IEEE, 2017.
- Ye et. al., Data contamination calibration for black-box llms. In findings of the association for computational linguistics (ACL) 2024, pages 10845-10861, 2024