Firm Foundations for Membership Inference Attacks Against Large Language Models

Jeffrey Wang [*1] **Jason Wang** [*1] **Marvin Li** [*1] **Seth Neel** [2]

Abstract

Membership inference attacks (MIAs) are a canonical way to assess a machine learning model's privacy properties. While many approaches have been proposed for conducting MIAs on language models, the extant literature has suffered numerous difficulties in constructing clean evaluations to test new techniques. In particular, subtle distribution shifts between member and non-member sets can completely change performance; recent work has underscored this by showing that "blind" methods with no access to the underlying model can perform far better than published methods on the same benchmarks. In this paper, we propose the first pipeline for principled evaluation of membership inference attacks against LLMs. Our approach leverages the insight that training data before and after a fixed point during training are drawn from the same distribution with minimal contamination; therefore, all open-source models with intermediate checkpoints and public training data are membership inference testbeds. We apply our framework to a half-dozen published attacks on the Pythia and OLMo family of models, from 70M to 7B parameters. To facilitate further privacy research, we open-source a modular library for designing and implementing attacks in this setting: `https://github.com/safr-ai-lab/pandora_llm`.

1. Introduction

Large language models (LLMs) have become indispensable workhorses for knowledge-intensive tasks, from summarizing medical records and drafting clinical notes to screening legal contracts and flagging anomalous financial transactions. As these models are increasingly deployed in high-stakes arenas such as healthcare and finance, privacy has become a first-order requirement for responsible use. As such, recent security research has revisited a threat model long studied for classifiers and regressors: that of membership inference attacks (MIAs) (Liu et al., 2021; Shokri et al., 2017). In an MIA, an adversary with some level of access to the model attempts to decide whether a particular example was used in a model's training data ("member") or is merely drawn i.i.d. from the same distribution ("non-member"). Highly accurate MIAs against LLMs would be useful not only to demonstrate privacy leakage, but could also be adopted as practical probes for related phenomena, including quantifying memorization during training (Zhou et al., 2023) and empirically evaluating the success of machine unlearning techniques (Kurmanji et al., 2023; Pawelczyk et al., 2023; Hayes et al., 2024).

While there has been a recent surge of research on MIAs against LLMs (Duan et al., 2024; Li et al., 2023; Mattern et al., 2023), it has been notoriously difficult to implement a correct MIA evaluation benchmark for pre-trained LLMs (Das et al., 2025; Maini & Suri, 2024). In particular, (Das et al., 2025) showed that many evaluations for MIAs commonly used in the literature are beaten by simple supervised learning methods trained on i.i.d splits of member/non-member data that have no access to the underlying model. This means the accuracy of these MIAs primarily emerges from the distributional differences between member and non-member examples rather than the privacy properties of underlying LLM. For instance, (Shi et al., 2023) proposed using *temporal differences* (from training data cutoffs of models) to construct member and non-member sets with the WikiMIA benchmark, where Wikipedia articles written before Jan 1st, 2017 were treated as the member data and articles after Jan 1st, 2023 were treated as non-member data. While reasonable at first glance, as such cutoffs enforce the member/non-member distinction, these sets are separable for another reason: their contents are distinct due to changing writing patterns over time. Indeed, (Das et al., 2025) trained a simple bag-of-words classifier that could distinguish between member and non-member data without the underlying model with a True Positive Rate (TPR) of 94.7% at a False Positive Rate (FPR) at 5%, far

[*]Equal contribution [1]Harvard College, Cambridge, MA, USA [2]Harvard Business School, Cambridge, MA, USA. Correspondence to: Jeffrey Wang <jgwang@college.harvard.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

outperforming the best attack proposed on the benchmark.

In this paper, we begin with an overview of the challenges researchers have faced in creating theoretically clean evaluations for membership inference. Then, we instantiate a cleaner dataset split for two common open LLM families, sanity checking them with "blind baselines" to verify the splits do not suffer from those same challenges. Finally, we benchmark a slate of common methods on these clean dataset splits, finding more limited performance.

2. Related Work

MIAs refer to a class of methods that can determine if a given data point z was included in the training dataset of a particular model θ. They were initially motivated by privacy considerations; if an adversary can determine if z was used to train θ with accuracy higher than the base rate, then information theoretically θ *must* encode some information about z that the attack is able to leverage. The setup to evaluate such MIAs is fairly straightforward: given an initial dataset D, some subset of the data is set aside prior to training (or sampled from the distribution after training) as a validation set D_{nonmem}, and some dataset D_{mem} is used for training to produce a new model θ. Then the following process is repeated a set number of times:

1. Flip a fair coin. If heads, sample with replacement $z \sim D_{\text{mem}}$; otherwise, sample $z \sim D_{\text{nonmem}}$.
2. Given z and some access to θ, the MIA produces a score p for that data point.

Given a membership inference score, for any threshold τ, there is a corresponding membership inference attack that predicts $z \in D_{\text{mem}}$ if and only if $p < \tau$. Each τ corresponds to a point on the Receiver Operating Characteristic (ROC) curve for the binary classification. As is common in the literature, to evaluate MIAs in this paper, we report the area under the ROC curve (AUC) as well as the TPR at low FPR of each method. The latter metric is widely used in the literature because any attack which can extract with high confidence even a small fraction of the training data poses a serious privacy risk (Carlini et al., 2022).

One key assumption underlies this evaluation scheme: individual samples in the training and validation data are drawn i.i.d. from the same distribution $\mathcal{P}$. Clearly, if θ contains no information about D_{mem} then the maximum attack accuracy is 50%, since D_{mem} and D_{nonmem} have identical marginal distributions. Thus in this case, any additional information about whether $z \sim D_{\text{mem}}$ or $z \sim D_{\text{nonmem}}$ must come from the model θ. On the other hand, if D_{mem} and D_{nonmem} differ in some way that is independent of whether θ is trained on it, then the maximum attack accuracy can be 100% even if the θ contains no information about D_{mem}. Imagine training an LLM on text which always begins with the phrase `The quick, brown fox jumps.` In that case, an MIA which knows nothing about the model θ can still achieve perfect accuracy by simply detecting if a given z begins with this phrase.

Existing Evaluations. Existing evaluations for MIAs fall in three broad categories. Several papers introduce benchmarks using the temporal cutoff approach of WikiMIA (Shi et al., 2023; Liu et al., 2024). Others utilize train/val splits of open models, most commonly using The Pile dataset with the Pythia family of models (Biderman et al., 2023; Gao et al., 2020). The MIMIR benchmark refines this approach, with an additional deduplication step of the non-member sets against training data[1] (Duan et al., 2024).

Train/Validation Overlap. Fuzzy notions of membership and what consitutes a unique data point also pose a bottleneck to rigorous evaluation (Hayes et al., 2025; Liu et al., 2025; Meeus et al., 2025). Even the published validation sets, which are supposed to be selected i.i.d from the same data corpus along with the training data, can be problematic for serving as a membership inference testbed. For instance, (Duan et al., 2024) show that there is substantial overlap between the training and validation sets for The Pile, which is the dataset used to train the Pythia and GPT-NeoX series of models, an oft-used benchmark for MIA papers.

Challenges of Deduplication. The Pythia family of models includes those trained on a deduplicated version of The Pile's training set; however, since only the training data is deduplicated, it has a different distribution than the validation split of The Pile. One solution, used in MIMIR, deduplicates the validation set against the train set—but this approach is not perfectly sound either, as this induces a different marginal distribution on member vs. non-member points. To see this, consider the setting where there are two very rare documents in the corpus. If both documents are split into the training corpus, only one will be included post deduplication; on the other hand, if they are both included in the validation split, they will both make it through preprocessing since they will only be deduplicated against the training set. It is easy to verify that the marginal probability that z is the rare document differs for the training and validation data. Indeed, (Meeus et al., 2025) find that a bag-of-words classifier is able to achieve extremely high AUCs (up to 0.86 on certain splits), implying that deduplication causes distribution shifts that violate the MIA assumptions.

[1] They create several member/non-member splits such that all non-member points with more than a p-proportion overlap in n-grams with any training data point are removed, for various settings of n and p.

It is important to note that these difficulties are in some sense unique to LLMs; for classical supervised machine learning (ML), e.g., training an image classifier, the evaluator will typically have access to the entire dataset and can choose the train/val partition before training the model. Because privacy researchers typically cannot train their own LLMs from end-to-end, the setup to evaluate MIAs for LLMs is not as simple. Thus it is extremely vital that we design experimental evaluations for MIAs that we verify are clean.

3. Our MIA Evaluation Pipeline

In this section, we will describe a clean method to derive member and non-member splits. This method only relies on access to the model checkpoints in the middle of training and the training data order, which is available for many models, like the Pythia family or OLMo (Groeneveld et al., 2024; Kim et al., 2025). We will also describe the procedure to generate member and non-member data using different checkpoints during the training, checking it against blind MIAs as in (Das et al., 2025; Maini & Suri, 2024).

Member and non-member data generation. Recall that the LLM data collection and training procedure typically involves forming documents taken from links scraped from the web, which are then tokenized and packed into training examples of a fixed size. These packed examples are deduplicated against each other and shuffled to form the *training order* of the model (Figure 1). This shuffling of the training order will be the key to our MIA evaluation setup. In our setup, we select a model, e.g., Checkpoint 300 in Figure 1, using the data before the checkpoint as the member data (herein {B2,B4,B1}) and the data after the checkpoint (herein {B3}) as the non-member data. We then evaluate the model at that checkpoint. Because the data before and after are drawn from the same distribution, the member and non-member data have the same marginal distributions.

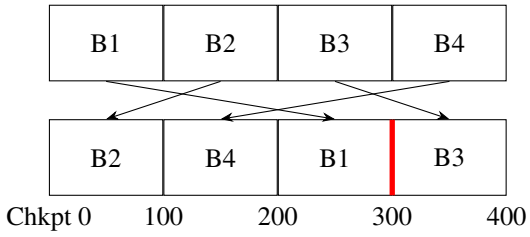

Figure 1. The shuffling of packed samples to construct the training data of Pythia and our checkpoint-based method to evaluate MIAs. In this scenario, we would sample from chunks {B2,B4,B1} for member data and chunk {B3} for the nonmember data to evaluate MIAs for the model at checkpoint 300.

Generating and learning from features. Using member and non-member data generated in the previous steps, we can evaluate different MIAs. We release our codebase as a modular Python library, with documentation and tests, that allows one to implement and benchmark new MIAs in this setting. We also implement (and in Section 4, benchmark) many MIAs from extant literature, including: simple loss thresholding (Yeom et al., 2018), Min-K (Shi et al., 2023), Min-K%++ (Zhang et al., 2025), Zlib entropy (Carlini et al., 2020), ReCaLL (Xie et al., 2024), and MoPe (Li et al., 2023).

Blind MIAs. Finally, our pipeline implements many "blind" MIAs as a baseline check that member and non-member data don't have distributional differences easily checkable with a simple supervised learning method. For a given set of splits, we train classifiers on statistical representations of the text, incuding Bag of Words, TF-IDF, Word2Vec representations (Mikolov et al., 2013), and BERT embeddings (Devlin et al., 2019). Full details on the training details of these classifiers is available in the Appendix.

4. Results

Setup. We instantiate our evaluation on the Pythia family of models, which were trained for 143,000 optimizer steps on a deduplicated version of The Pile, representing approximately 1.5 epochs. We evaluate the model at step 97,000 (over 95% of the way through a full epoch), uniformly sampling data points before and after this step.[2]

We also benchmark our method on the OLMo family of models trained on the Dolma dataset (Groeneveld et al., 2023; Soldaini et al., 2024). Due to computational constraints, we only benchmark OLMo results on the 7B parameter model. We use the checkpoint at step 400,000, which represents 88% of the way through the first epoch of training. Full results are available in Appendix C.

Blind MIAs. First, we validate that blind MIAs without any access to the model, like supervised learning techniques on a split of member and non-member data points, get no signal in our setting (Table 1). In all settings, we train three different classifiers (logistic regression, random forest, and a neural network) on 4,000 points from member and non-member classes; then we evaluate our classifier on 1,000 points from each class. We report the best test AUC among these three methods in Table 1 (and Table 3 in the Appendix). For every AUC we report, we also provide standard error intervals computed over 1,000 bootstraps. Full details on the supervised classifiers are available in Appendix A.

[2]Note that the use of intermediate checkpoints is an additional access assumption, but most literature already assumes access to a 1-epoch intermediate checkpoint to ensure that each datapoint is seen a uniform number of times.

Table 1. Sanity check: supervised model-free ("blind") MIAs get no signal on our cleaned split of The Pile.

Blind MIA	Best AUC	AUC SE	$\text{TPR}_{1\%}$
BoW (Tokens)	0.505	0.0128	0.01
TFIDF (Tokens)	0.503	0.0127	0.006
W2V (Text)	0.490	0.0131	0.015
BERT (Text)	0.497	0.0127	0.012

Existing MIAs. After validating our dataset, we benchmark the MIAs listed in Section 3, which we implement in our library. We find that current MIAs have limited success on our dataset drawn from the pre-training data (see Table 2). Note that this does not exclude the possibility that on certain subsets of the training data (in particular, the GitHub documents in The Pile), MIAs can still be performant, which was found in previous works (Duan et al., 2024).

5. Conclusion

In this paper, we identify subtle distributional pitfalls of previous MIA evaluations for LLMs and propose a principled framework to avoid them. Our pipeline supports several popular open model families and we benchmark many existing attacks on it. Finally, this work provides a clear call-to-action for future model releases: the underlying dataset should be processed and deduplicated in tandem, then split into training and validation sets for minimal contamination and matching distributions.

6. Limitations

While our setting is general and can be instantiated across many open model families, there are several limitations and areas for future work. First, our framework requires open training data, model checkpoints, and the exact permutation of training examples throughout training. While many open model families provide this information, others, e.g., Llama and Qwen, do not (Touvron et al., 2023; DeepSeek-AI et al., 2025; Yang et al., 2025). Despite this important limitation to any checkpoint-based approach, we argue that this methodology enables us to investigate MIA efficacy much more rigorously than existing work, and hope that it encourages model providers to release this auxiliary information in the future. Second, due to computational and data constraints, we only evaluated MIAs on models pretrained almost exclusively on English web data. Finally, in our experimental setup, we do not benchmark any models larger than 7 billion parameters, again due to compute constraints. We are excited by future work verifying the extent of our negative results across scales, data subsets, languages, and training paradigms.

Table 2. We benchmark several published MIAs against the Pythia family of models on our cleaned split of The Pile.

Model	MIA	AUC	AUC SE	$\text{TPR}_{1\%}$
70m	LOSS	0.513	0.0130	0.006
70m	Min-K	0.500	0.0130	0.012
70m	Min-K++	0.496	0.0128	0.011
70m	Zlib	0.499	0.0129	0.016
70m	MoPe	0.487	0.0126	0.025
70m	ReCaLL	0.490	0.0130	0.010
160m	LOSS	0.513	0.0130	0.007
160m	Min-K	0.500	0.0127	0.011
160m	Min-K++	0.499	0.0126	0.014
160m	Zlib	0.499	0.0127	0.016
160m	MoPe	0.495	0.0132	0.015
160m	ReCaLL	0.490	0.0124	0.006
410m	LOSS	0.512	0.0128	0.007
410m	Min-K	0.498	0.0132	0.006
410m	Min-K++	0.496	0.0126	0.014
410m	Zlib	0.499	0.0127	0.016
410m	MoPe	0.504	0.0130	0.014
410m	ReCaLL	0.493	0.0131	0.008
1b	LOSS	0.513	0.0122	0.007
1b	Min-K	0.502	0.0125	0.007
1b	Min-K++	0.498	0.0126	0.013
1b	Zlib	0.500	0.0129	0.016
1b	MoPe	0.495	0.0126	0.015
1b	ReCaLL	0.487	0.0123	0.008
2.8b	LOSS	0.506	0.0131	0.007
2.8b	Min-K	0.489	0.0124	0.006
2.8b	Min-K++	0.489	0.0126	0.014
2.8b	Zlib	0.499	0.0129	0.016
2.8b	MoPe	0.493	0.0131	0.006
2.8b	ReCaLL	0.498	0.0131	0.010

References

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling, 2023.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D. X., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security Symposium*, 2020. URL https://api.semanticscholar.org/CorpusID:229156229.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. Membership inference attacks from first principles, 2022.

Das, D., Zhang, J., and Tramèr, F. Blind baselines beat membership inference attacks for foundation models, 2025. URL https://arxiv.org/abs/2406.16201.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Duan, M., Suri, A., Mireshghallah, N., Min, S., Shi, W., Zettlemoyer, L., Tsvetkov, Y., Choi, Y., Evans, D., and Hajishirzi, H. Do membership inference attacks work on large language models?, 2024.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Tafjord, R. K. O., Jha, A. H., Ivison, H., Wang, I. M. Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohany, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Merrill, T. K. W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. Olmo: Open language model by ai2, 2023.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. Olmo: Accelerating the science of language models, 2024. URL https://arxiv.org/abs/2402.00838.

Hayes, J., Shumailov, I., Triantafillou, E., Khalifa, A., and Papernot, N. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy, 2024.

Hayes, J., Swanberg, M., Chaudhari, H., Yona, I., Shumailov, I., Nasr, M., Choquette-Choo, C. A., Lee, K., and Cooper, A. F. Measuring memorization in language models via probabilistic extraction, 2025. URL https://arxiv.org/abs/2410.19482.

Kim, G., Li, Y., Spiliopoulou, E., Ma, J., Ballesteros, M., and Wang, W. Y. Detecting training data of large language models via expectation maximization, 2025. URL https://arxiv.org/abs/2410.07582.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. Towards unbounded machine unlearning, 2023.

Li, M., Wang, J., Wang, J., and Neel, S. MoPe: Model perturbation based privacy attacks on language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13647–13660, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.842. URL https://aclanthology.org/2023.emnlp-main.842.

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., and Lin, Z. When machine learning meets privacy: A survey and outlook. *ACM Comput. Surv.*, 54(2), mar 2021. ISSN 0360-0300. doi: 10.1145/3436755. URL https://doi.org/10.1145/3436755.

Liu, K. Z., Choquette-Choo, C. A., Jagielski, M., Kairouz, P., Koyejo, S., Liang, P., and Papernot, N. Language models may verbatim complete text they were not explicitly trained on, 2025. URL https://arxiv.org/abs/2503.17514.

Liu, Z., Zhu, T., Tan, C., Lu, H., Liu, B., and Chen, W. Probing language models for pre-training data detection, 2024. URL https://arxiv.org/abs/2406.01333.

Maini, P. and Suri, A. Reassessing emnlp 2024's best paper: Does divergence-based calibration for membership inference attacks hold up? 2024. URL https://www.anshumansuri.com/blog/2024/calibrated-mia/. Accessed May 1, 2025.

Mattern, J., Mireshghallah, F., Jin, Z., Schoelkopf, B., Sachan, M., and Berg-Kirkpatrick, T. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11330–11343, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719.

Meeus, M., Shilov, I., Jain, S., Faysse, M., Rei, M., and de Montjoye, Y.-A. Sok: Membership inference attacks on llms are rushing nowhere (and how to fix it), 2025. URL https://arxiv.org/abs/2406.17975.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Pawelczyk, M., Neel, S., and Lakkaraju, H. In-context unlearning: Language models as few shot unlearners, 2023.

Shi, W., Ajith, A., Xia, M., Huang, Y., Liu, D., Blevins, T., Chen, D., and Zettlemoyer, L. Detecting pretraining data from large language models, 2023.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models, 2017.

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A. H., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Ravichander, A., Richardson, K., Shen, Z., Strubell, E., Subramani, N., Tafjord, O., Walsh, P., Zettlemoyer, L., Smith, N. A., Hajishirzi, H., Beltagy, I., Groeneveld, D., Dodge, J., and Lo, K. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024. URL https://arxiv.org/abs/2402.00159.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

Xie, R., Wang, J., Huang, R., Zhang, M., Ge, R., Pei, J., Gong, N. Z., and Dhingra, B. Recall: Membership inference via relative conditional log-likelihoods, 2024. URL https://arxiv.org/abs/2406.15968.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

Zhang, J., Sun, J., Yeats, E., Ouyang, Y., Kuo, M., Zhang, J., Yang, H. F., and Li, H. Min-k URL https://arxiv.org/abs/2404.02936.

Zhou, Z., Xiang, J., Chen, C., and Su, S. Quantifying and analyzing entity-level memorization in large language models, 2023.

A. Attack Details.

Supervised Learning for Classifiers. We use random forest, logistic regression, and a neural network as supervised classifiers for our model-free blind baselines (traditional word embedding features). For random forest, we use 100 trees with no limit on maximum depth and the Gini impurity splitting criterion. For logistic regression, we use a default lbfgs solver with 1,000 iteration maximum and L2-regularization. For neural network, we train for 10 epochs on a batch size of 128 using the Adam optimizer with a learning rate of 0.001, with a ReLU architecture of 4 layers going from input dimension to 250, to 100, to 10, then to 1 dimensions.

Evaluation Configuration. We evaluate all attacks using 1,000 train and validation points. For all supervised methods used in the Pythia blind baselines, we train on 4,000 points; for all supervised methods used in OLMo baselines, we train on 8,000 points.

Other evaluation details. MoPe uses 10 perturbations with $\sigma = 0.005$, as recommended in the paper (Li et al., 2023). For Min-K and Min-K++, we use $k = 0.1$. For ReCaLL, we use 100-token long prefixes as the extra conditioning.

B. Dataset Details

Pythia data construction. We evaluate a Pythia checkpoint at 97,000 steps into training. To construct the dataset for evaluation, we randomly sample data points from steps 0 to 97,000, and then from 97,000 to 98,500 (approximately the end of Epoch 1). We run all attacks on Pythia models trained using *deduplicated* training data.

OLMo data construction. OLMo 7B was trained on 1.25 epochs from the 2T token training corpus Dolma for a total of 556,000 training steps. The remaining 0.25 epochs after the first epoch are taken from another shuffling of the training corpus. Because our checkpoint-based method is only valid if the model did not see the data after the checkpoint, we restrict our attention to the model state through the first epoch, after 452,000 training steps. We then choose to evaluate MIAs for the model at checkpoint 400,000. We choose the member data by randomly sampling from data that the model saw between checkpoints 0 and 400,000, and the non-member data by randomly sampling from data the model saw between checkpoints 401,000 and 452,000. Because the entire Dolma dataset already undergoes several different kinds of deduplicating before being used to train OLMo (see Section 5.4 of (Soldaini et al., 2024) for details), this guarantees that member and non-member data have the same marginal distributions.

C. OLMo Results

Blind MIAs. As with the Pythia models in Section 4 of the paper, we run blind supervised baselines for the OLMo model as well. These results are given in Table 3.

Table 3. Sanity check: supervised model-free ("blind") MIAs get no signal on our cleaned split of Dolma.

Blind MIA	Best AUC	AUC SE	$\text{TPR}_{1\%}$
BoW (Tokens)	0.491	0.0131	0.01
TFIDF (Tokens)	0.491	0.0128	0.008
W2V (Text)	0.494	0.0129	0.012
BERT (Text)	0.496	0.0126	0.009

Other Attacks. We benchmark various MIAs from previous works, as in Section 4, this time against OLMo 7B. See Table 4 for full results.

D. Additional Details

Compute Estimates. To run the experiments, we used a compute node with an NVIDIA A100 80GB GPU. All experiments in this paper can be run on a single one of these GPUs. All results for pretrained MIAs are on model sizes 70M, 160M, 410M, 1B, and 2.8B for Pythia, and 7B for OLMo. To run blind baselines, we create features using 4,000 randomly sampled

Table 4. We benchmark several published MIAs against OLMo 7B on our cleaned split of Dolma.

MIA	AUC	AUC SE	$\text{TPR}_{1\%}$
LOSS	0.505	0.0125	0.01
Min-K	0.504	0.0129	0.013
Min-K++	0.490	0.0124	0.009
Zlib	0.499	0.0128	0.019
ReCaLL	0.521	0.0130	0.011

member/non-member points, train a classifier, and evaluate the classifier on $1{,}000$ distinct member/non-member points. Most of these steps are runnable on a consumer laptop. As noted previously, in the pretrained setting, we evaluate all MIAs on $1{,}000$ points from member and non-member splits, which requires only running inference on models. In MoPe, we run inference on ten times as many points (because we use ten perturbed models). In total these attacks took around 3 A100 GPU-days.

AI Assistants. While all work was done and checked by the authors, language models were used in the process to refine ideas, write small snippets of code, and tune writing for clarity.