Evaluating SURP MIA performance on code samples

Ísak Bieltvedt Jónsson[1]

Supervisor(s): Maliheh Izadi[1]**, Ali Al-Kaswan**[1]**, Jonathan Katzy**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 25, 2026

Name of the student: Ísak Bieltvedt Jónsson
Final project course: CSE3000 Research Project
Thesis committee: Maliheh Izadi, Ali Al-Kaswan, Jonathan Katzy, Prof.dr.ir. R.L. (Inald) Lagendijk

Abstract

Code language models are pretrained on massive datasets scraped from public repositories which are rarely disclosed. Membership Inference Attacks (MIAs) aim to predict whether specific samples were used in training but attack performance is contested. Previous work has shown that many attacks on LLMs perform randomly when evaluated on independent and identically distributed (i.i.d.) members and non-members. We consider three MIAs: LOSS, MinK%, and SURP (where each attack extends the last with additional filtering of tokens considered for the membership signal), on StarCoder2-3B and Mellum-4B using the AISE MIA dataset, which contains 100,000 Java files with verified membership labels. We address a gap in the evaluation of these attacks on i.i.d. code samples and in the detailed comparison of SURP and MinK%. A bag-of-words (BoW) classifier is used to measure distribution shift with an expected ROC-AUC of 0.5 under i.i.d. conditions. We achieve a ROC-AUC of 0.91 confirming substantial distribution shift. We apply two debiasing procedures to construct evaluation subsets: Taking samples close to the BoW decision boundary reduces BoW ROC-AUC performance to 0.66, while selecting BoW misclassified samples fails to reduce shift. After debiasing, all attacks perform at or below the bag-of-words baseline, with ROC-AUC between 0.55 and 0.63 and TPR at 5% FPR between 0.05 and 0.16; suggesting random performance under strict i.i.d conditions. Hyperparameter ablation reveals that SURP collapses to MinK% under optimization: optimal configurations disable SURP filtering or have classification agreement exceeding 94% excluding one outlier. These results extend prior natural language findings to code: reference-free attacks exploit distributional differences rather than detecting membership.

1 Introduction

Code language models are trained on datasets scraped from public repositories, often without disclosure of contents [11]. This creates challenges for copyright compliance and benchmark integrity, as non-permissive licenses may be violated and evaluation data may leak into training sets [3; 21]. Membership Inference Attacks (MIAs) aim to determine whether specific samples were used in training, enabling auditing of these concerns. We evaluate three MIAs on code samples: LOSS thresholds average per-token loss [19], MinK% focuses on the lowest-probability tokens [17], and SURP extends MinK% by additionally filtering to low-entropy positions [20]. However, the effectiveness of these attacks remains contested [7; 13]. MIAs assume member (IN) and non-member (OUT) samples are identically distributed; when this assumption fails, performance reflects distribution shift rather than membership detection [6; 7]. Meeus et al. [13] show that under controlled conditions, reference-free attacks perform at random on natural language, but exclude code from evaluation due to benchmark artifacts. The relationship between SURP and MinK% also remains unclear. Zhang et al. [20] report SURP advantages on WikiMIA and other benchmarks, yet performance is near-equivalent on MIMIR [7], the only benchmark with partial distribution shift control. Prior work does not conduct hyperparameter ablation on datasets where the attacks show similar performance, leaving open whether SURP's entropy filtering provides genuine benefit or whether observed differences are benchmark-specific. We address these gaps by evaluating LOSS [19], MinK% [17], and SURP [20] on the AISE MIA dataset [1; 2], containing 100,000 Java files with membership labels verified against The Stack V2 [11]. We evaluate on StarCoder2-3B [11] and Mellum-4B [14], both trained on The Stack V2. To control for distribution shift, we apply two debiasing procedures: recursive misclassification, which iteratively removes samples a bag-of-words classifier correctly labels, and No-Class [8], which selects samples near the classifier decision boundary.

We investigate three questions. **RQ1:** Does distribution shift exist between IN and OUT samples in the AISE MIA dataset? **RQ2:** How do LOSS, MinK%, and SURP perform on StarCoder2-3B and Mellum-4B after controlling for shift? **RQ3:** How does SURP differ from MinK% on code samples under hyperparameter optimization?

First, we confirm substantial shift (BoW ROC-AUC 0.91) and show that No-Class debiasing partially reduces it with approximately 0.66 BoW ROC-AUC, while recursive misclassification fails entirely. Second, after debiasing, all attacks perform at or below the BoW baseline, extending prior natural language findings to code. Third, hyperparameter ablation suggests that SURP and MinK% are effectively equivalent on code, with optimal configurations typically disabling entropy filtering entirely; we observe isolated exceptions on one debiased subset, but high cross-fold variance limits confidence in these cases.

2 Background and Related Work

2.1 Membership Inference Attacks

Membership inference attacks (MIAs) predict whether a sample was part of a model's training set. We formalize this using a modified *membership inference security game* [4] where the adversary lacks access to the underlying training distribution.

Definition 1 (*Membership Inference Security Game* [4])**.** *The game proceeds between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$:*

1. *The challenger samples a training dataset $D \leftarrow \mathcal{D}$ and trains model $f_\theta \leftarrow \mathcal{T}(D)$.*
2. *The challenger flips bit b: if $b = 0$, samples $x \leftarrow \mathcal{D}$ such that $x \notin D$ (an OUT sample); otherwise selects $x \leftarrow D$ (an IN sample).*
3. *The challenger sends x to the adversary.*
4. *The adversary outputs $\hat{b} \leftarrow \mathcal{A}^{f_\theta}(x)$.*
5. *Output 1 if $\hat{b} = b$, else 0.*

We refer to $x \in D$ as IN samples and $x \notin D$ as OUT samples throughout.

For code LLMs, f_θ is an autoregressive language model and x a token sequence. The attacks we consider output a continuous score thresholded into a binary prediction: $\mathcal{A}(x) = 1[\mathcal{A}'(x) > \tau]$ [4].

2.2 MIAs for LLMs

Shokri et al. [18] introduced membership inference attacks for machine learning models by training *shadow models* that mimic target behavior. A binary classifier then distinguishes IN from OUT samples based on differences in model confidence. This approach was designed for traditional classifiers and smaller neural networks.

Carlini et al. [4] developed the Likelihood Ratio Attack (LiRA) specifically for Language Models, treating membership as a hypothesis test. For each candidate sample, LiRA compares the target model's loss against a reference distribution computed from 64 to 256 shadow models, half trained *with* the sample (IN models) and half *without* (OUT models). A sample is likely IN if its loss under the target is unexpectedly low relative to this reference. LiRA and its derivatives achieve SOTA performance but require access to the target models pretraining distribution [4; 9].

LiRA's requirements are problematic for modern LLMs. Pretraining corpora contain trillions of tokens from often undisclosed sources, making shadow model training computationally prohibitive and distribution matching infeasible [7; 17]. Recent work by Hayes et al. [9] successfully scaled LiRA to up to 1B parameters trained on 20B tokens, but how it translates to modern LLM sizes is unclear. These challenges have motivated *reference-free* attacks using only target model outputs.

The simplest reference-free approach exploits overfitting: since models minimize loss on training data, IN samples should exhibit lower loss than OUT samples [19]. The LOSS [19] attack thresholds average per-token loss as a membership signal. Shi et al. [17] observe that averaging dilutes signals from memorized tokens and propose MinK%, which averages loss over only the $k\%$ of tokens with lowest log-likelihood. Zhang et al. [20] extend this with SURP, additionally filtering to tokens with low entropy, targeting positions where the model is confident but incorrect.

These attacks require only output logits (*grey-box* access) and no shadow model training, making them practical for auditing deployed models, but their effectiveness remains contested [13]. Zhang et al. [20] report SURP improvements over MinK% primarily on benchmarks with known distribution shift; on the MIMIR benchmark where shift is better controlled, both attacks achieve near-identical scores [20; 13]. Whether this reflects functional equivalence or insufficient evaluation has not been systematically tested. We address this gap through detailed comparison on code samples.

2.3 Evaluating MIAs

MIAs are binary classifiers traditionally evaluated using ROC-AUC (Receiver Operator Characteristic - Area Under Curve), the probability that an attack assigns a higher score to a randomly chosen IN sample than a randomly chosen OUT sample. However, Carlini et al. [4] argue this metric obscures practical utility. In copyright litigation or benchmark contamination detection, false positives are costly (falsely accusing a model provider), and auditors need high-confidence identification of *some* IN samples rather than average-case performance. They advocate evaluating TPR at low FPR (e.g., 1% or 5%), isolating the regime where attacks provide actionable evidence.

2.4 Challenges

Several factors complicate membership inference for LLMs beyond the computational barriers discussed above. Modern pretraining uses massive datasets, often trillions of tokens, typically for one epoch, reducing memorization and the signal MIAs exploit [7]. Membership boundaries are inherently fuzzy: code samples share boilerplate, patterns, and forked content, creating substantial overlap between IN and OUT samples within the same domain [7]. Pretraining sets are rarely disclosed, denying adversaries access to the underlying distribution required for shadow model training and fair evaluation set construction.

The most significant challenge, however, is *distribution shift*: violations of the assumption that IN and OUT samples are independant and identically distributed (i.i.d.). We dedicate the following section to this issue given its centrality to MIA evaluation.

2.5 Distribution Shift

The membership inference security game (Definition 1) assumes IN and OUT samples are drawn from the same distribution $\mathcal{D}$. When this assumption fails, we say there is *distribution shift*, and attack performance becomes confounded: high scores may reflect distributional differences rather than membership signals [7; 6; 13].

Prevalence. Distribution shift is pervasive in MIA evaluation. Adversaries lack pretraining access and must construct evaluation sets *post-hoc*, identifying probable IN samples through metadata like timestamps, licenses, or domain. This process systematically differs from how pretraining data was selected, introducing shift. Early benchmarks used temporal cutoffs: WikiMIA labels Wikipedia pages added after model release as OUT samples. Duan et al. [7] first demonstrated that such temporal shift inflates attack performance, with significant drops on their MIMIR benchmark derived from The Pile's held-out test split. However, MIMIR's postprocessing, including aggressive deduplication of the GitHub subset, introduced new artifacts. Das et al. [6] showed that "blind baselines" (bag-of-words classifiers, date detection, rare word frequency) outperform sophisticated MIAs on shifted benchmarks, achieving 98.7% ROC-AUC on WikiMIA versus 83.9% for the best MIA.

Consequences. Meeus et al. [13] systematically evaluated 14 MIAs across benchmarks with varying shift. Using three methodologies to ensure i.i.d. conditions (held-out test splits from pretraining data, Regression Discontinuity Design on ArXiv papers near a training cutoff, and injecting random token sequences with controlled membership), they found that

all reference-free attacks performed randomly. They conclude that apparent MIA success on standard benchmarks reflects distribution shift, not membership detection. Critically, their evaluation excludes code samples: MIMIR's GitHub subset was omitted due to deduplication artifacts. Whether reference-free MIAs perform randomly on code under i.i.d. conditions is therefore untested.

Detection. Shift can be detected via model-free classifiers. Under i.i.d. conditions a bag-of-words (BoW) classifier should get 0.5 ROC-AUC distinguishing IN from OUT samples. If it performs significantly above chance, exploitable distributional differences exist [6; 13]. Following Meeus et al., we treat BoW performance as a baseline against which to compare MIA attacks: performance not exceeding the BoW cannot be attributed to membership detection beyond distributional exploitation.

Debiasing. Methods to reduce distribution shift remain limited. Held-out test splits from pretraining data guarantee i.i.d. IN and OUT samples but require cooperation from model providers [13]. Regression Discontinuity Design mitigates temporal shift but not other sources [13]. Eichler et al. [8] propose No-Class (NoC) debiasing, selecting samples near a classifier's decision boundary where distributional features are ambiguous; they report residual BoW ROC-AUC around 0.58 after debiasing on natural language data. No study has applied debiasing methods to code samples. We adapt NoC to the code domain and evaluate whether it sufficiently removes shift for fair MIA evaluation.

Effects on shadow model attacks. The distribution shift literature focuses on reference-free attacks. How shift affects shadow model attacks like LiRA, which require training data from the same distribution as the target, remains understudied. If the shadow training distribution differs from the true pretraining distribution, reference scores may be miscalibrated. We note this as an open question but focus our empirical work on reference-free attacks where shift effects are better characterized.

3 Approach

Post-hoc MIA evaluation sets often exhibit distribution shift between IN and OUT samples, inflating attack performance. Our approach addresses this through three steps. First, we detect shift using Bag-of-Words classifiers. ROC-AUC above 0.5 indicates exploitable distributional differences. Second, we apply two debiasing procedures: recursive misclassification, which iteratively removes correctly-classified samples, and No-Class selection, which retains samples near the classifier decision boundary. We measure success by residual BoW ROC-AUC. Third, we evaluate SURP, MinK%, and LOSS on debiased datasets, comparing attack performance against the BoW baseline rather than random chance. We focus on reference-free attacks since shadow-model approaches require training data from the target distribution, unavailable for most code LLMs. We conduct hyperparameter ablation over SURP's entropy filtering to test whether it benefits code samples.

4 Experimental Setup

4.1 Research Questions

RQ1: *Are IN and OUT samples in the AISE MIA dataset i.i.d.?* We apply model-free BoW classifiers to test whether IN and OUT samples are distinguishable without model access. ROC-AUC above 0.5 indicates exploitable distribution shift.

RQ2: *How do LOSS, MinK%, and SURP perform on StarCoder2-3B and Mellum-4B after controlling for distribution shift?* We apply debiasing procedures and evaluate attack performance. Following prior work [13], we expect ROC-AUC to approach random baseline under i.i.d. conditions.

RQ3: *How does SURP differ from MinK% on code samples under hyperparameter optimization?* We compare SURP and MinK% through performance metrics, hyperparameter ablation, and classification agreement. SURP extends MinK% with entropy-based filtering; we investigate whether this provides benefit on code samples.

4.2 Target Models

We evaluate StarCoder2-3B [11] and Mellum-4B [14], both trained on The Stack V2 [11], enabling membership verification via hash matching. Models are evaluated at native precision (bfloat16), accessing only output logits consistent with the grey-box threat model assumed by SURP [20], MinK% [17], and LOSS [19].

4.3 Dataset

We use The Heap [10] AISE MIA subset [2; 1] containing Java files from public GitHub repositories with non-permissive licenses (e.g., GPL).

Membership Determination. Labels are derived through deduplication against The Stack V2. Exact duplicates are identified via SHA-256 hashing. Near duplicates are detected using MinHash LSH with 7-character shingles, 128 permutations, and Jaccard threshold of 0.7.

Evaluation Set. The subset contains 50,000 members (exact hash matches) and 50,000 non-members (no exact or near matches), providing a balanced classification task. Near-duplicate samples are excluded from non-members to create cleaner class separation, though this may inflate attack performance (see Threats to Validity).

Preprocessing. Files are tokenized to a maximum of 8,192 tokens using each model's native tokenizer.

4.4 Detecting Distribution Shift

Following [6; 13; 7], we train Bag-of-Words (BoW) classifiers to detect distributional differences between IN and OUT samples. ROC-AUC above 0.5 (the random baseline) indicates distribution shift that MIAs may exploit rather than true membership signals.

We evaluate multiple BoW configurations varying vocabulary size and feature type (Appendix A.1) on the 50/50 train/validation split. We report validation ROC-AUC and use the best classifier for subsequent debiasing.

4.5 Debiasing the Dataset

We apply two debiasing procedures to reduce distribution shift, evaluating effectiveness using BoW performance on the resulting datasets.

Recursive BoW Misclassification (RMC)
The BoW classifier is applied recursively: correctly classified samples are removed, IN & OUT labels balanced, and the classifier is retrained on misclassified samples until fewer than 100 samples remain. Misclassified samples have weaker distributional signals, so each iteration removes exploitable shift. We evaluate RMC Iter 1 and RMC Iter 2 datasets.

No-Class Procedure (NoC)
We adapt the No-Class procedure from Eichler et al. [8]. The original method averages predictions from multiple classifiers; we instead use a single BoW classifier trained on the full training split. Samples with predictions near the decision boundary ($|\text{score} - 0.5| < h$) are selected. To ensure label balance, we undersample to equal counts of true positives, false positives, true negatives, and false negatives. We use $h = 0.10, 0.05$. To verify debiasing, we train a new BoW on each subset and report its ROC-AUC.

4.6 SURP Attack Implementation

We implement SURP following Zhang et al. [20]. Given document $d = (x_1, \ldots, x_n)$ and model $\mathcal{M}$, we extract two feature vectors per position: ground truth log probability and entropy

$$L_i^{GT} = \log p(x_i \mid x_{<i}; \mathcal{M})$$

$$E_i = -\sum_{v \in \mathcal{V}} p(v \mid x_{<i}; \mathcal{M}) \log p(v \mid x_{<i}; \mathcal{M})$$

.

The SURP score averages log probabilities over "surprising" tokens—low-probability predictions ($L_i^{GT} < L_k$, the k-th percentile) in high-confidence contexts ($E_i < \varepsilon$):

$$\text{SURP}(d, \mathcal{M}) = \frac{1}{|S_e \cap S_p|} \sum_{i \in S_e \cap S_p} L_i^{GT}$$

where $S_p = \{i \mid L_i^{GT} < L_k\}$ and $S_e = \{i \mid E_i < \varepsilon\}$.

MinK% and LOSS. Setting ε arbitrarily high disables entropy filtering, yielding MinK%. Setting $k = 100$ additionally yields LOSS. All three attacks are evaluated within our hyperparameter optimization.

Document-Level Aggregation. The attack formula is applied across all tokens (up to 8,192), yielding one membership score per document.

4.7 Evaluation Metrics

We report ROC-AUC and TPR@5%FPR following Carlini et al. [4].

4.8 Hyperparameter Optimization

We use 5-fold cross-validation, reporting mean and standard deviation across folds.

For SURP, we perform grid search over $k \in \{1, 2, 3, 4, 5, 6, 8, 10, 15, 20, 30, 50, 100\}$ and $\varepsilon \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 7.0, 9.0, 12.0\}$, optimizing for ROC-AUC or TPR@5%FPR. For MinK%, we search over k with ε set high (disabling entropy filtering). LOSS uses $k = 100$ with no filtering.

Ablation Study. We examine performance across the full grid, fixing ε at maximum to isolate MinK% behavior, then fixing k at maximum to isolate entropy filtering benefit. We also report mean AUC over all ε (vs k) and mean AUC over all k (vs ε) to show the marginal effect of each hyperparameter.

4.9 SURP and MinK% Comparison

We compare SURP and MinK% across: (1) ROC-AUC and TPR@FPR on all datasets and models, (2) performance surfaces over ε and k to identify conditions where entropy filtering helps or hurts, and (3) classification agreement—the percentage of samples receiving identical predictions.

4.10 Implementation Details

Experiments are conducted on NVIDIA V100 GPUs (32GB). Models are loaded via Hugging Face Transformers at native precision (bfloat16). BoW classifiers use scikit-learn's LogisticRegression. Features are cached to HDF5 for efficient hyperparameter search. Code is available at https://github.com/Bieltvedt/RP-Evaluating-SURP.

5 Results

5.1 BoW and distribution shift

A BoW classifier with a vocabulary of 50k words gets the best validation ROC-AUC of 0.91 (Appendix A.1) providing a strong indication of distribution shift [6; 7; 13]. These BoW parameters will be used unless otherwise stated moving forward.

5.2 Debiasing

Table 1: BoW ROC-AUC across debiasing conditions

Training	Full	RMC1	RMC2	NoC.10	NoC.05
Full	.915	.217	.647	.524	.518
Subset	.915	.874	.890	.666	.656
n	50,000	6,836	1,224	6,176	3,360

No-Class debiasing according to Section 4 results in a BoW ROC-AUC drop from ~ 0.91 to ~ 0.66 (Table 1). We are able to partially remove the distribution shift between In and Out samples.

Recursive misclassification according to Section 4 results in BoW ROC-AUC remaining around ~ 0.9 until too few samples remain (Table 1); RMC is not able to significantly reduce IN/Out distribution shift.

We emphasize results on the NoC 0.10 dataset moving forward.

5.3 Attack performance & best configurations

Metrics are lower on debiased datasets with ROC-AUC of $\sim [0.55 : 0.63]$ and TPR@5%FPR of $\sim [0.05 : 0.16]$ indicating inability of attacks to reliably distinguish between members

Table 2: MIA performance overview

	StarCoder2-3B		Mellum-4B	
Method	AUC	TPR@5%	AUC	TPR@5%
Full dataset				
SURP	0.693	0.207	0.700	0.254
MinK	0.693	0.207	0.700	0.254
LOSS	0.652	0.157	0.643	0.168
BoW	0.915	0.665	0.915	0.665
No-Class (NoC) 0.10				
SURP	0.614	0.121	0.634	0.160
MinK	0.614	0.121	0.631	0.139
LOSS	0.586	0.080	0.580	0.087
BoW	0.666	0.162	0.652	0.162
Recursive Misclassification (RMC) iteration 1				
SURP	0.572	0.058	0.568	0.084
MinK	0.547	0.060	0.568	0.092
LOSS	0.537	0.045	0.551	0.065
BoW	0.873	0.449	0.873	0.449

and non members. None of the attacks/configurations are able to outperform a BoW baseline. SURP and MinK% perform better on mellum than starcoder. (Tables 2,1)

SURP and MinK% performance is identical on full dataset, SURP finds a slight advantage on NoC 0.10 with mellum-4b ($\Delta 0.003$ ROC-AUC; $\Delta 0.021$ TPR@5%FPR), and a major advantage on the RMC dataset for starcoder ($\Delta 0.025$ AUC) (Table 2). Looking at the actual configurations where the attacks differ:

We observe minimal effects from entropy filtering for configurations where including it was found optimal. Advantage over MinK% is minor, thresholds filter $\sim 1\%$ of tokens, and classification agreement remains high. RMC 1 ROC-AUC optimization for starcoder2 is an outlier and is the only configuration where entropy filtering provides a real performance benefit (Table 3).

5.4 Hyperparameter ablation study

When fixing epsilon to max we observe peak metric scores at $k \in [1:5]$ with performance generally degrading as k becomes 100 across models, metrics, and datasets with the least impact on RMC 1 (Figure 2, Appendix B.2).

When fixing k to 100 we observe increasing ROC-AUC as epsilon becomes larger, with occasional local maxima at epsilon [1:3], converging to a max ROC-AUC when equivalent to LOSS at high epsilon values (Figure 2). Exclusively applying epsilon filtering only beats LOSS on RMC dataset w.r.t. ROC-AUC (Appendix B.2)

RMC 1 generally has much higher uncertainty than other datasets.

Taking the mean effect of epsilon over all values of k we see that on average epsilon filtering has a negative effect on performance (Figure 3). Mean ROC-AUC is max when epsilon filtering is turned off except for the RMC 1 outlier case.

Mean effect of k over all values of epsilon shows that on average applying epsilon filtering degrades MinK performance such that it converges to LOSS (Figure 3).

6 Discussion

6.1 BoW confirms distribution shift between IN vs OUT samples

BoW should perform randomly if In & Out samples are i.i.d. [13; 6; 7]. ROC-AUC of 0.915 (Table 1) confirms In vs Out distribution shift.

We hypothesize that this stems from two primary sources:

1. **Structural Selection Bias:** The StackV2 was constructed using license-based filtering intended to include only permissively licensed code. The heap [10] is constructed from non-permissively licensed code. Consequently, the IN samples (copyleft) represent "filtering failures" that leaked into the training set due to imperfect license attribution [11; 15]. OUT samples consist of copyleft files that were either successfully excluded or never scraped. There is a structural discrepancy.
2. **Temporal Shift:** The Heap [10] was collected at a later date than the Stack [11; 15]. This introduces a temporal discrepancy between the two sets.

Future work should explore stronger shift detection methods. Embedding based classifiers may capture semantic similarities that BoW misses and their embedding spaces could reveal distributional differences.

6.2 Debiasing Effectiveness

Neither debiasing procedure reduces BoW ROC-AUC to the random baseline expected under i.i.d. conditions [7; 13].

Recursive misclassification fails entirely: after three iterations with only 214 samples remaining, BoW ROC-AUC persists at approximately 0.9 (Table 1). This suggests distribution shift in our dataset is *pervasive* rather than concentrated in easily-separable samples. License-based filtering during Stack V2 construction likely introduced systematic differences affecting most files, so removing correctly-classified samples simply retrains the classifier on the remaining signal rather than eliminating it.

No-Class debiasing achieves partial success, reducing BoW ROC-AUC to ~ 0.65. This matches Eichler et al.'s [8] results on Gutenberg and suggests boundary-selection is more effective than iterative removal for code datasets with pervasive shift.

Constructing truly i.i.d. evaluation sets for code MIAs remains an open challenge. Two approaches merit investigation: (1) using held-out test splits from models with disclosed pretraining data and (2) applying Regression Discontinuity Design around known training cutoffs for models with dated pretraining corpora [13]. Whether embedding based sample selection can achieve tighter i.i.d. conditions than BoW based debiasing also warrants further study.

6.3 Performance After Debiasing

Across all debiasing conditions and both models, MIA ROC-AUC and TPR@5%FPR falls below the corresponding BoW

Table 3: Configurations where SURP outperforms MinK. Δ values indicate positive target metric gain vs MinK. %Filt shows percentage of tokens filtered compared to MinK. Hyperparameter uncertainty reflects per-fold variation in optimal values from 5-fold CV. "Target" indicates optimization objective. Agreement measured at optimal threshold (Youden's J) or target tpr.

Model	Dataset	Target	SURP Metrics		SURP Hyperparams			MinK Metrics		Comparison		
			AUC	TPR@5%	k	ε	%Filt	AUC	TPR@5%	MinK k	Δ	Agree
Mellum-4B	NoC	AUC	0.621±.009	0.149±.027	2±1	**7**±1	0.8	0.619±.014	0.136±.027	2±1	**+0.002**	96.8
Mellum-4B	NoC	TPR	0.612±.009	0.153±.026	1±1	**7**±1	1.0	0.614±.016	0.140±.023	1±1	**+0.013**	94.3
StarCoder2-3B	RMC Iter 1	AUC	0.572±.012	0.059±.008	10±4	**4**±0	**18.0**	0.547±.014	0.059±.005	6±3	**+0.025**	**72.5**
StarCoder2-3B	RMC Iter 1	TPR	0.548±.014	0.083±.004	2±4	**7**±0	1.5	0.531±.013	0.077±.005	1±3	**+0.006**	86.8
Mellum-4B	RMC Iter 1	AUC	0.573±.024	0.111±.016	3±1	**7**±2	0.7	0.569±.021	0.117±.024	3±1	**+0.004**	97.4
Mellum-4B	RMC Iter 1	TPR	0.572±.023	0.130±.010	2±1	**7**±2	0.8	0.559±.019	0.129±.017	1±1	**+0.002**	91.6

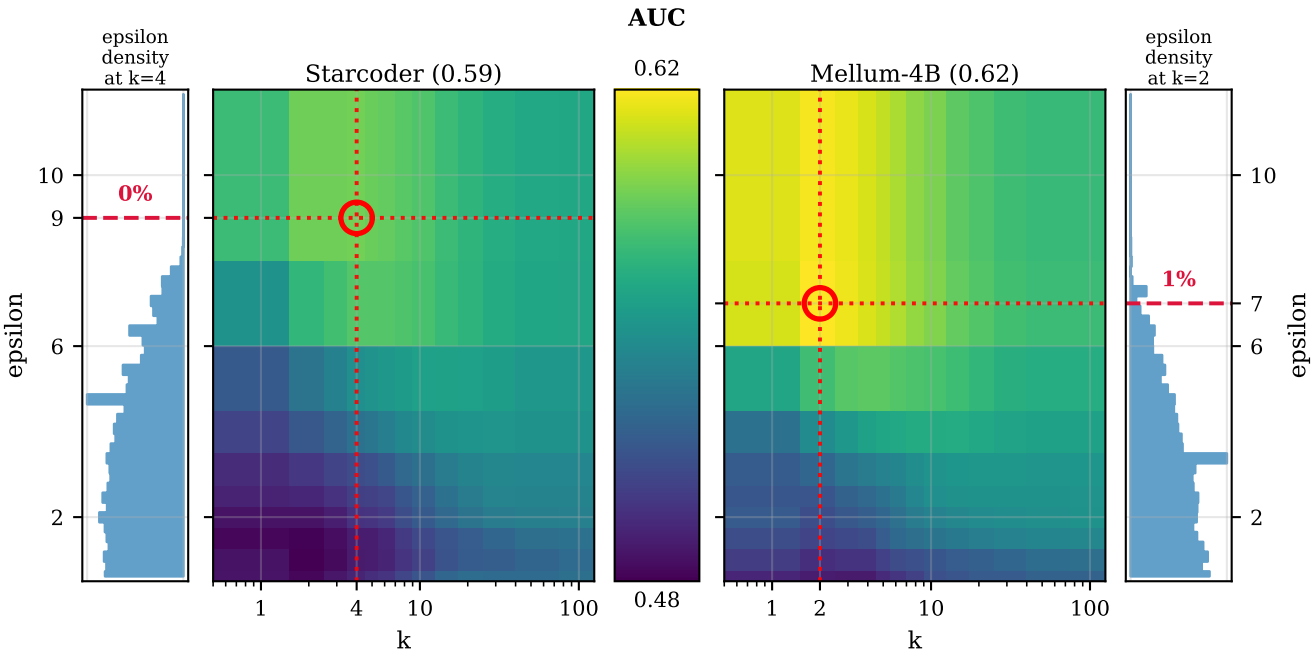

Figure 1: ROC-AUC heatmaps for starcoder2-3b & mellum-4b on NoC 0.10 dataset with best config, epsilon density, and % of tokens filtered

baseline (Table 2). Reference-free attacks fail to detect membership signal beyond what trivial distributional features already capture.

This extends Meeus et al.'s natural language findings [13] to code: when IN/OUT shift is even partially controlled, SURP, MinK%, and LOSS provide no evidence of true membership detection. The residual performance above the random baseline reflects incomplete debiasing, not membership signal.

6.4 SURP Converges to MinK%

Under optimization, SURP collapses to MinK%. On Full and NoC datasets, optimal ε values filter 0–1% of tokens with classification agreement exceeding 94% (Table 3) and no entropy filtering in most cases. Mellum-4B on NoC shows marginal filtering (0.8%) with negligible gain (Δ0.002 AUC).

StarCoder2-3B on RMC Iter 1 is an outlier: ε=4 filters 18% of tokens and agreement drops to 72.5%, yielding +0.025 AUC. But high cross-fold variance (k=9±3.8 , $\epsilon = 4.2 \pm 0.4$) suggests unstable optimization rather than genuine benefit (Table 3).

Zhang et al. [20] report SURP advantages on WikiMIA and Dolma-Book, with larger gaps on LLaMA than Pythia. Performance is near equivalent to MinK% on MIMIR [7]. These gains may be benchmark and model specific. We hypothesize that SURP and MinK% are practically equivalent on code samples, particularly under i.i.d. conditions. Given that both attacks perform randomly when distribution shift is controlled [13], this equivalence may have limited practical significance.

Testing the SURP-MinK% equivalence hypothesis requires evaluation across additional code models (CodeLlama, DeepSeek-Coder) and natural language models under controlled conditions. If equivalence holds broadly, this would simplify the MIA landscape by eliminating entropy filtering as a useful dimension.

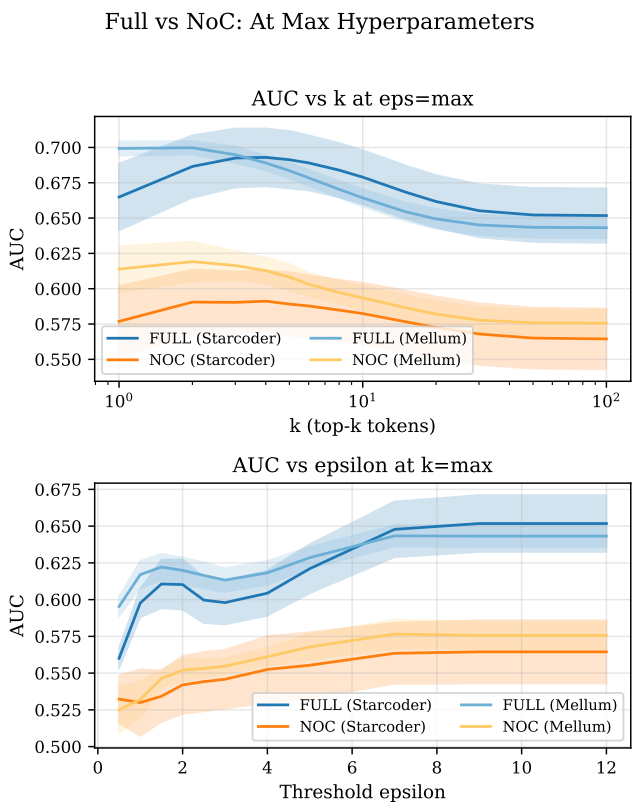

Figure 2: Mean & std deviation of roc-auc across folds vs k at max epsilon and vice versa

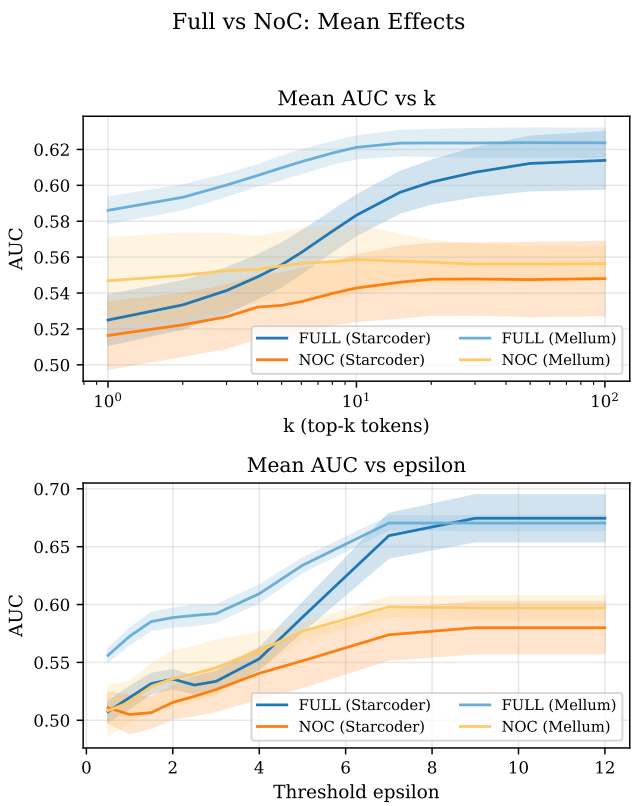

Figure 3: Mean & std deviation across folds of roc-auc vs epsilon averaged over all k or vice versa

6.5 Hyperparameter Ablation

The ablation confirms that k selection drives attack performance while ε filtering provides no benefit on code. Varying k at maximum ε shows peak ROC-AUC at low k values, degrading as k approaches 100 (Figure 2). Varying ε at maximum k shows ROC-AUC increasing as ε increases, converging to LOSS when filtering is disabled. Averaging over the full grid: mean AUC versus k shows that ε filtering degrades MinK% performance toward LOSS; mean AUC versus ε is maximized when entropy filtering is disabled entirely (Figure 3).

RMC Iter 1 is the sole exception where ε filtering marginally improves performance. However, high cross-fold variance on this dataset limits confidence in this result (Table 3).

SURP assumes "surprising" tokens (low entropy, low ground-truth probability) carry membership signal. Our results suggest this signal may be weak or absent in code samples. Understanding *why* entropy filtering fails on code, whether due to different token entropy distributions, different memorization patterns, or simply insufficient signal under i.i.d. conditions, could further inform the design of code-specific attacks.

6.6 Practical Implications

Reference-free MIAs perform at effectively random levels on code samples when IN/OUT distribution shift is controlled (Table 2), mirroring results in natural language [13]. If high MIA scores primarily reflect distributional differences rather than genuine memorization, these metrics cannot reliably support copyright litigation or benchmark contamination detection.

MIAs may prove more useful as components of hybrid pipelines. Carlini et al. [5] use membership signals to filter generated samples in data extraction attacks. Recent work by Sahili et al. [16] indicates that reference free attacks perform well as part of data extraction pipelines and that candidate suffixes are guaranteed to be i.i.d.

6.7 Distinguishing Types of Distribution Shift

The literature treats distribution shift as a monolithic phenomenon [7; 6; 13]. We observe two distinct components:

IN/OUT shift: Member and non-member samples in the evaluation set come from measurably different distributions. Detectable via BoW classifiers (Table 1); partially correctable through debiasing.

EVAL/PT shift: The evaluation set as a whole differs from the actual pretraining distribution. Not detectable without pretraining data access; not addressed by current debiasing methods.

Our evaluation set contains only copyleft Java files from The Heap, while StackV2 spans multiple languages and li-

cense types [11]. Even after NoC debiasing reduces BoW ROC-AUC to 0.66, EVAL/PT shift likely persists.

Does EVAL/PT shift meaningfully affect attack performance? Meeus et al. [13] apply Regression Discontinuity Design to ArXiv papers near OpenLLaMA's training cutoff, achieving BoW AUC of 0.534 ± 0.027. Despite residual EVAL/PT shift (ArXiv papers are unrepresentative of general web scrapes), MIAs perform at random baseline. This suggests IN/OUT shift is the primary driver of inflated performance, and removing it may suffice for fair evaluation even when EVAL/PT shift remains.

Our results support this interpretation. After NoC debiasing, MIA performance (AUC $\sim$0.55–0.63) falls below the residual BoW baseline (Table 2), providing no evidence of membership detection beyond distributional exploitation. The distinction between shift types warrants further investigation, but controlling IN/OUT shift is necessary and potentially sufficient for valid MIA evaluation on code samples.

Disentangling IN/OUT and EVAL/PT shift effects requires controlled experiments varying each independently. One approach: evaluate MIAs on held-out splits (eliminating IN/OUT shift) from multiple pretraining subsets differing in domain composition (varying EVAL/PT shift). If performance remains at baseline regardless of EVAL/PT shift magnitude, this would confirm that IN/OUT control suffices for valid evaluation. Alternatively, if EVAL/PT shift affects performance, evaluation protocols must ensure representativeness of the pretraining distribution—a requirement difficult to meet for models with undisclosed training data.

6.8 Threats to Validity

Internal Validity

Residual distribution shift. NoC debiasing reduces BoW ROC-AUC but not to the 0.5 expected under i.i.d. conditions (Table 1). Our conclusion that attacks perform near-randomly is relative to this residual baseline.

Hyperparameter sensitivity. Grid search over discrete values may miss optimal configurations. High cross-fold variance on RMC datasets (Table 3) suggests unstable optimization, limiting confidence in reported performance differences.

Document-level implementation. Our naive approach averages token-level scores across up to 8,192 tokens per file. This may dilute localized membership signals that chunk-level or sliding-window approaches could capture [12].It may also reduce the effectiveness of entropy filtering. Files exceeding 8,192 tokens are truncated, potentially discarding membership-relevant content.

External Validity

Language and domain scope. All experiments use non-permissively licensed Java files from the AISE MIA dataset [1; 2]. Results may not generalize.

Model coverage. We evaluate only StarCoder2-3B [11] and Mellum-4B [14], both trained on The Stack V2 [11]. Findings may not equate for different models. Closed-source models remain untested.

Construct Validity

Near-duplicate exclusion. Excluding near-duplicates from the OUT set creates cleaner class separation than realistic adversarial settings where membership boundaries are inherently fuzzy [7]. This may inflate both attack and BoW performance relative to practical deployment.

Membership ground truth. Hash-based membership determination assumes exact file matches. Partial memorization of near-duplicates is not captured by this binary labeling.

Conclusion Validity

Sample size after debiasing. NoC and RMC procedures substantially reduce sample counts, increasing metric variance. Results on smaller debiased subsets should be interpreted with appropriate caution given wider confidence intervals.

7 Conclusion

Membership inference attacks (MIAs) predict whether specific samples were used to train a machine learning model. We evaluated three reference-free attacks (which require only model output probabilities and no auxiliary training data) SURP [20], MinK% [17], and LOSS [19] on code samples.

A core challenge for MIA evaluation is distribution shift: when member (IN) and non-member (OUT) are not independent and identically distributed (i.i.d.), attacks may exploit distributional differences rather than detecting membership [7; 6; 13]. We tested for shift using a Bag-of-Words (BoW) classifier, which achieves ROC-AUC of 0.91 on the AISE MIA dataset [1; 2; 10], far above the 0.5 expected if IN and OUT samples were i.i.d.

To control for this shift, we applied two debiasing procedures: recursive misclassification and No-Class selection [8]. No-Class partially reduces shift (BoW ROC-AUC drops to 0.66); recursive misclassification fails entirely. Crucially, across all debiased datasets, both models (StarCoder2-3B, Mellum-4B), and all metrics, every attack performs below the residual BoW baseline [13]. Since attacks cannot exceed what a simple word-frequency classifier achieves, they provide no evidence of true membership detection. This suggests attacks would perform at random under true i.i.d. conditions, extending prior natural language findings [13] to code.

We also compared SURP and MinK% through hyperparameter ablation. SURP extends MinK% with entropy-based filtering [20; 17], but under optimization both attacks converge: optimal configurations filter none (most common) or under 1% of tokens with over 94% classification agreement.

For practitioners: reference-free MIAs cannot reliably prove code membership for copyright litigation or contamination detection. Future work should construct i.i.d. evaluation sets using held-out test splits or Regression Discontinuity Design [13], explore embedding-based debiasing, and investigate MIAs as components of data extraction pipelines [5; 16] rather than standalone classifiers.

8 Responsible Research

This section reflects on the ethical dimensions of our research and addresses reproducibility considerations.

8.1 Ethical Considerations

Research Intent and Societal Impact

Our work evaluates membership inference attacks on code language models with the explicit goal of *auditing* rather

than enabling misuse. MIAs serve legitimate purposes: enabling copyright holders to verify whether their code was used in training without consent, and helping researchers detect benchmark contamination that undermines evaluation validity [3; 21]. Our findings that reference-free attacks perform near randomly under controlled conditions inform both stakeholders about the current limitations of these tools.

The double edged potential of MIA research warrants consideration. In principle, effective attacks could be misused to identify individuals whose data was used in training, enabling targeted harassment or discrimination. However, several factors mitigate this concern in our context. First, code samples lack the personal identifiability of natural language text; a Java file rarely contains information that could be traced to a specific individual beyond the repository author. Second, our evaluation uses publicly available code from GitHub repositories where authorship is already disclosed. Third, our negative results demonstrate that current reference-free attacks cannot reliably distinguish members from non-members, limiting immediate misuse potential.

Data Provenance and Consent

The AISE MIA dataset derives from The Heap, which contains non-permissively licensed code [3]. While this raises questions about the ethics of using copyleft code in research, our work differs fundamentally from LLM training: we do not learn from the code's content but rather evaluate whether models have done so. Our use constitutes analysis of publicly available artifacts for research purposes rather than commercial exploitation.

The IN samples in our dataset represent code that was included in Stack V2 despite non-permissive licensing, likely due to imperfect license attribution during dataset construction [11]. Our work helps surface the extent of such inclusion, potentially benefiting rights holders seeking to understand how their code has been used.

Model Access and Compute

We evaluate only open weight models (StarCoder2-3B and Mellum-4B), avoiding ethical concerns around proprietary model access. Our experiments require only grey-box access (output logits) rather than model weights, aligning with realistic auditing scenarios. The computational requirements are modest: feature extraction for 100,000 samples completes in approximately 48 GPU-hours on NVIDIA V100 hardware, placing this research within reach of academic groups without requiring large scale infrastructure.

Potential for Misinterpretation

Our results could be misinterpreted in ways that harm legitimate interests. Rights holders might incorrectly conclude that MIAs are fundamentally incapable of detecting unauthorized training data use, when our findings apply specifically to reference-free attacks under distribution shift. We emphasize that reference-based attacks like LiRA [4] and hybrid approaches [16] may achieve stronger results under appropriate conditions. Similarly, model providers should not cite our work as evidence that their training practices are undetectable; our negative results stem from methodological challenges in MIA evaluation, not from absence of memorization.

8.2 Reproducibility

Code and Data Availability

All code required to reproduce our experiments is publicly available at https://github.com/Bieltvedt/RP-Evaluating-SURP. The repository includes:

- Attack implementations for SURP, MinK%, and LOSS
- Bag-of-words classifier training and debiasing procedures
- Hyperparameter optimization scripts with 5-fold cross-validation
- Evaluation pipelines for all reported metrics
- Configuration files specifying exact experimental parameters

The AISE MIA dataset is available at [1; 2].

Computational Environment

Experiments were conducted on NVIDIA V100 GPUs (32GB VRAM). Models were loaded via Hugging Face Transformers at native precision (bfloat16). We specify exact library versions in our repository's requirements file. Random seeds are fixed where applicable.

Expected Variance

Results may differ slightly from reported values due to inherent non-determinism in LLM inference and variations in hardware/software configurations. Based on our 5-fold cross-validation, we expect reproduced ROC-AUC values to fall within ± 0.01 of reported means for most configurations. The RMC datasets exhibit higher variance (± 0.02–0.03) due to smaller sample sizes. We report standard deviations throughout to enable appropriate interpretation of reproduced results.

References

[1] AISE-TUDelft. MIA-exact-public: Membership Inference Attack Dataset (Exact Duplicates). https://huggingface.co/datasets/AISE-TUDelft/MIA-exact-public, 2024. Dataset for membership inference attacks on code language models with exact duplicate filtering.

[2] AISE-TUDelft. MIA-public: Membership Inference Attack Dataset. https://huggingface.co/datasets/AISE-TUDelft/MIA-public, 2024. Dataset for membership inference attacks on code language models.

[3] Ali Al-Kaswan and Maliheh Izadi. The (ab)use of open source code to train large language models, 2023.

[4] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles, 2022.

[5] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2021.

[6] Debeshee Das, Jie Zhang, and Florian Tramèr. Blind baselines beat membership inference attacks for foundation models, 2025.

[7] Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models?, 2024.

[8] Cédric Eichler, Nathan Champeil, Nicolas Anciaux, Alexandra Bensamoun, Héber H. Arcolezi, and José Maria De Fuentes. *Nob-MIAs: Non-biased Membership Inference Attacks Assessment on Large Language Models with Ex-Post Dataset Construction*, page 441–456. Springer Nature Singapore, November 2024.

[9] Jamie Hayes et al. Strong membership inference attacks on massive datasets and (moderately) large language models, 2025.

[10] Jonathan Katzy, Razvan Mihai Popescu, Arie van Deursen, and Maliheh Izadi. The heap: A contamination-free multilingual code dataset for evaluating large language models, 2025.

[11] Anton Lozhkov, Raymond Li, Loubna Ben Allal, et al. Starcoder 2 and the stack v2: The next generation, 2024.

[12] Matthieu Meeus, Shubham Jain, Marek Rei, and Yves-Alexandre de Montjoye. Did the neurons read your book? document-level membership inference for large language models, 2024.

[13] Matthieu Meeus, Igor Shilov, Shubham Jain, Manuel Faysse, Marek Rei, and Yves-Alexandre de Montjoye. Sok: Membership inference attacks on llms are rushing nowhere (and how to fix it), 2025.

[14] Nikita Pavlichenko, Iurii Nazarov, Ivan Dolgov, Ekaterina Garanina, Dmitry Ustalov, Ivan Bondyrev, Kseniia Lysaniuk, Evgeniia Vu, Kirill Chekmenev, Joseph Shtok, Yaroslav Golubev, Anton Semenkin, and Uladzislau Sazanovich. Mellum: Production-grade in-ide contextual code completion with multi-file project understanding, 2025.

[15] BigCode Project. The stack v2. https://huggingface.co/datasets/bigcode/the-stack-v2, 2024.

[16] Ali Al Sahili, Ali Chehab, and Razane Tajeddine. On the effectiveness of membership inference in targeted data extraction from large language models, 2025.

[17] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models, 2024.

[18] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017.

[19] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

[20] Anqi Zhang and Chaofeng Wu. Adaptive pre-training data detection for large language models via surprising tokens, 2024.

[21] Xin Zhou, Yichen Li, Xiaobing Sun, Chunrong Fang, Yuying Xie, Xin Chen, Xin Tan, Yuan Yuan, Yuting Zhang, and Zhenyu Chen. Lessleak-bench: A first investigation of data leakage in llms across 83 software engineering benchmarks, 2025.

A Supplementary Tables

A.1 BoW configuration comparison

Table 4: Bag-of-words classifier experiments for detecting distribution shift

Experiment	Features	ROC-AUC
Word	10,000	0.895
Word	50,000	**0.915**
Word Bigram	30,000	0.909
Char Trigram	30,000	0.895
Char N-gram	50,000	0.895
SGD Word	100,000	0.891

A.2 Best 50/50 train/test split configs

Table 5: MIA performance (train/test split)

Dataset	Model	Method	k	ε	AUC	TPR@5%	TPR@1%	%Filt	ΔAUC
Full	StarCoder	SURP	4	9	0.693	0.207	0.049	0	—
	StarCoder	MinK	4	—	0.693	0.207	0.049	—	—
	StarCoder	LOSS	—	—	0.652	0.157	0.030	—	—
	Mellum-4B	SURP	2	9	0.700	0.254	0.086	0	—
	Mellum-4B	MinK	2	—	0.700	0.254	0.086	—	—
	Mellum-4B	LOSS	—	—	0.643	0.168	0.047	—	—
NoC 0.10	Mellum-4B	SURP	2	7	0.634	0.160	0.050	1.3	+0.003
	Mellum-4B	MinK	2	—	0.631	0.139	0.042	—	—
	Mellum-4B	LOSS	—	—	0.580	0.087	0.022	—	—
RMC Iter 1	StarCoder	SURP	10	4	0.572	0.058	0.010	35.3	+0.026
	StarCoder	MinK	6	—	0.547	0.060	0.009	—	—
	StarCoder	LOSS	—	—	0.537	0.045	0.008	—	—
	Mellum-4B	SURP	3	7	0.573	0.118	0.019	14.9	+0.004
	Mellum-4B	MinK	3	—	0.569	0.118	0.022	—	—
	Mellum-4B	LOSS	—	—	0.551	0.065	0.015	—	—
RMC Iter 2	StarCoder	SURP	1	9	0.571	0.085	0.031	0	—
	StarCoder	MinK	1	—	0.571	0.085	0.031	—	—
	StarCoder	LOSS	—	—	0.515	0.069	0.029	—	—

Single train/test split results. %Filt: tokens filtered by ε. ΔAUC: gain from filtering.

A.3 Best Cross Validation configs

B Supplementary Figures

B.1 Heatmaps

B.2 Ablation Plots

Table 6: Best TPR@5%FPR configurations (train/test split)

		StarCoder2-3B				Mellum-4B			
Dataset	Method	k	ε	AUC	TPR@5%	k	ε	AUC	TPR@5%
Full	SURP	2	9	0.687	0.213	1	9	0.699	0.273
	MinK	2	—	0.687	0.213	1	—	0.699	0.273
	LOSS	—	—	0.652	0.157	—	—	0.643	0.168
NoC 0.10	SURP	3	9	0.613	0.124	1	7	0.627	0.162
	MinK	3	—	0.613	0.124	2	—	0.631	0.139
	LOSS	—	—	0.586	0.080	—	—	0.580	0.087
RMC Iter 1	SURP	2	7	0.548	0.081	2	9	0.566	0.128
	MinK	1	—	0.531	0.076	2	—	0.566	0.128
	LOSS	—	—	0.537	0.045	—	—	0.551	0.065
RMC Iter 2	SURP	4	9	0.559	0.088	—	—	—	—
	MinK	4	—	0.559	0.088	—	—	—	—
	LOSS	—	—	0.515	0.069	—	—	—	—

Configs optimized for TPR@5%FPR (may differ from best-AUC configs).

Table 7: Cross-validated MIA performance across datasets

Dataset	Model	Method	k	ε	AUC	TPR@5%	%Filt	ΔAUC
Full	StarCoder	SURP	$4_{\pm 0.5}$	9	$0.693_{\pm 0.020}$	$0.206_{\pm 0.013}$	0	—
	StarCoder	MinK	$4_{\pm 0.5}$	—	$0.693_{\pm 0.020}$	$0.206_{\pm 0.013}$	—	—
	StarCoder	LOSS	—	—	$0.652_{\pm 0.019}$	$0.158_{\pm 0.009}$	—	—
	Mellum-4B	SURP	$2_{\pm 0.4}$	$9_{\pm 0.8}$	$0.700_{\pm 0.005}$	$0.251_{\pm 0.017}$	0	—
	Mellum-4B	MinK	$2_{\pm 0.4}$	—	$0.700_{\pm 0.005}$	$0.251_{\pm 0.017}$	—	—
	Mellum-4B	LOSS	—	—	$0.643_{\pm 0.007}$	$0.167_{\pm 0.017}$	—	—
NoC 0.10	StarCoder	SURP	$4_{\pm 1.0}$	9	$0.591_{\pm 0.022}$	$0.094_{\pm 0.022}$	0	—
	StarCoder	MinK	$4_{\pm 1.0}$	—	$0.591_{\pm 0.022}$	$0.094_{\pm 0.022}$	—	—
	StarCoder	LOSS	—	—	$0.564_{\pm 0.021}$	$0.063_{\pm 0.008}$	—	—
	Mellum-4B	SURP	$2_{\pm 0.6}$	$8_{\pm 1.0}$	$0.621_{\pm 0.009}$	$0.149_{\pm 0.027}$	1.3	+0.002
	Mellum-4B	MinK	$2_{\pm 0.6}$	—	$0.619_{\pm 0.014}$	$0.136_{\pm 0.027}$	—	—
	Mellum-4B	LOSS	—	—	$0.576_{\pm 0.009}$	$0.072_{\pm 0.013}$	—	—
RMC Iter 1	StarCoder	SURP	$9_{\pm 3.8}$	$4_{\pm 0.4}$	$0.572_{\pm 0.012}$	$0.059_{\pm 0.008}$	35.3	+0.026
	StarCoder	MinK	$6_{\pm 3.0}$	—	$0.547_{\pm 0.014}$	$0.059_{\pm 0.005}$	—	—
	StarCoder	LOSS	—	—	$0.537_{\pm 0.014}$	$0.045_{\pm 0.002}$	—	—
	Mellum-4B	SURP	$3_{\pm 1.4}$	$6_{\pm 2.2}$	$0.573_{\pm 0.024}$	$0.111_{\pm 0.016}$	14.9	+0.004
	Mellum-4B	MinK	$4_{\pm 1.2}$	—	$0.569_{\pm 0.021}$	$0.117_{\pm 0.024}$	—	—
	Mellum-4B	LOSS	—	—	$0.551_{\pm 0.024}$	$0.066_{\pm 0.010}$	—	—
RMC Iter 2	StarCoder	SURP	1	9	0.571	0.085	0	—
	StarCoder	MinK	1	—	0.571	0.085	—	—
	StarCoder	LOSS	—	—	0.515	0.069	—	—

5-fold CV results. Values shown as mean$_{\pm \text{std}}$. %Filt: tokens filtered by ε. ΔAUC: gain from filtering.

Table 8: Best TPR@5%FPR configurations (5-fold CV)

		StarCoder2-3B					Mellum-4B				
Dataset	Method	k	ε	AUC	TPR@5%	ΔTPR	k	ε	AUC	TPR@5%	ΔTPR
Full	SURP	2	9	$0.687_{\pm 0.022}$	$0.213_{\pm 0.011}$	+0.007	1	9	$0.699_{\pm 0.005}$	$0.272_{\pm 0.012}$	+0.021
	MinK	2	—	$0.687_{\pm 0.022}$	$0.213_{\pm 0.011}$	+0.007	1	—	$0.699_{\pm 0.005}$	$0.272_{\pm 0.012}$	+0.021
NoC 0.10	SURP	2	9	$0.591_{\pm 0.023}$	$0.104_{\pm 0.024}$	+0.011	1	7	$0.612_{\pm 0.009}$	$0.153_{\pm 0.026}$	+0.004
	MinK	2	—	$0.591_{\pm 0.023}$	$0.104_{\pm 0.024}$	+0.011	1	—	$0.614_{\pm 0.016}$	$0.140_{\pm 0.023}$	+0.004
RMC Iter 1	SURP	2	7	$0.548_{\pm 0.014}$	$0.083_{\pm 0.004}$	+0.024	2	7	$0.572_{\pm 0.023}$	$0.130_{\pm 0.010}$	+0.019
	MinK	1	—	$0.531_{\pm 0.013}$	$0.077_{\pm 0.005}$	+0.018	1	—	$0.559_{\pm 0.019}$	$0.129_{\pm 0.017}$	+0.012

Configs optimized for TPR@5%FPR. ΔTPR: improvement vs. best-AUC config.

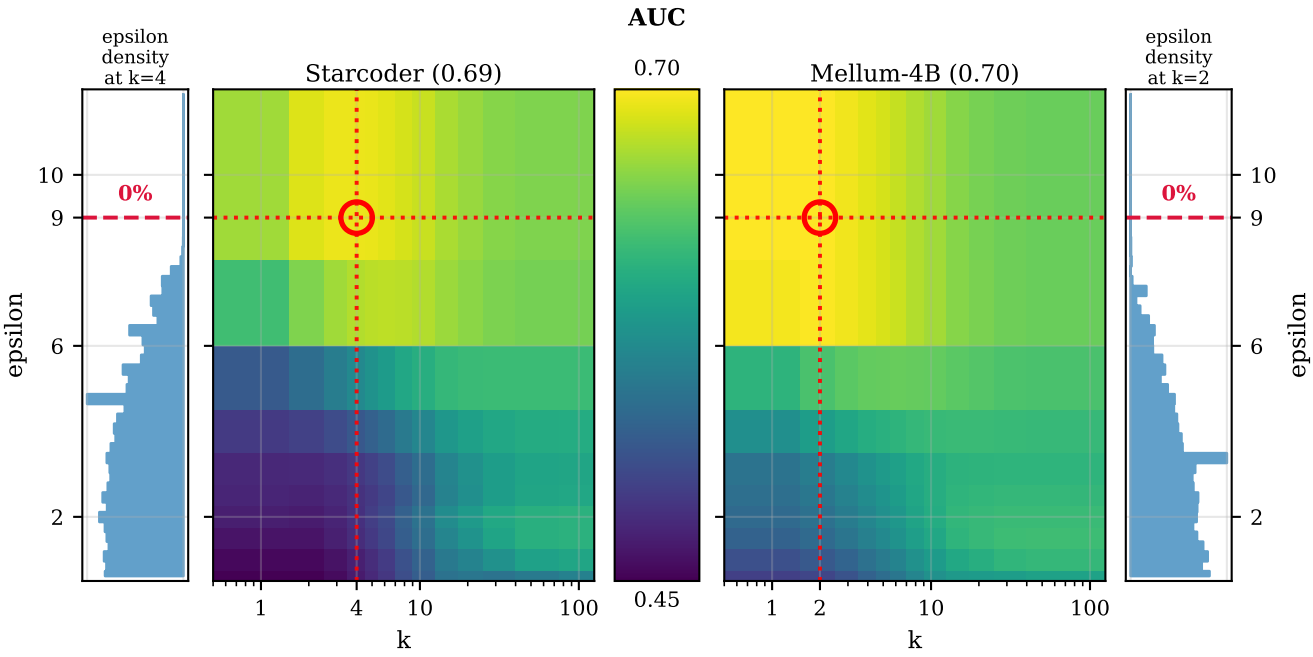

Figure 4: Caption

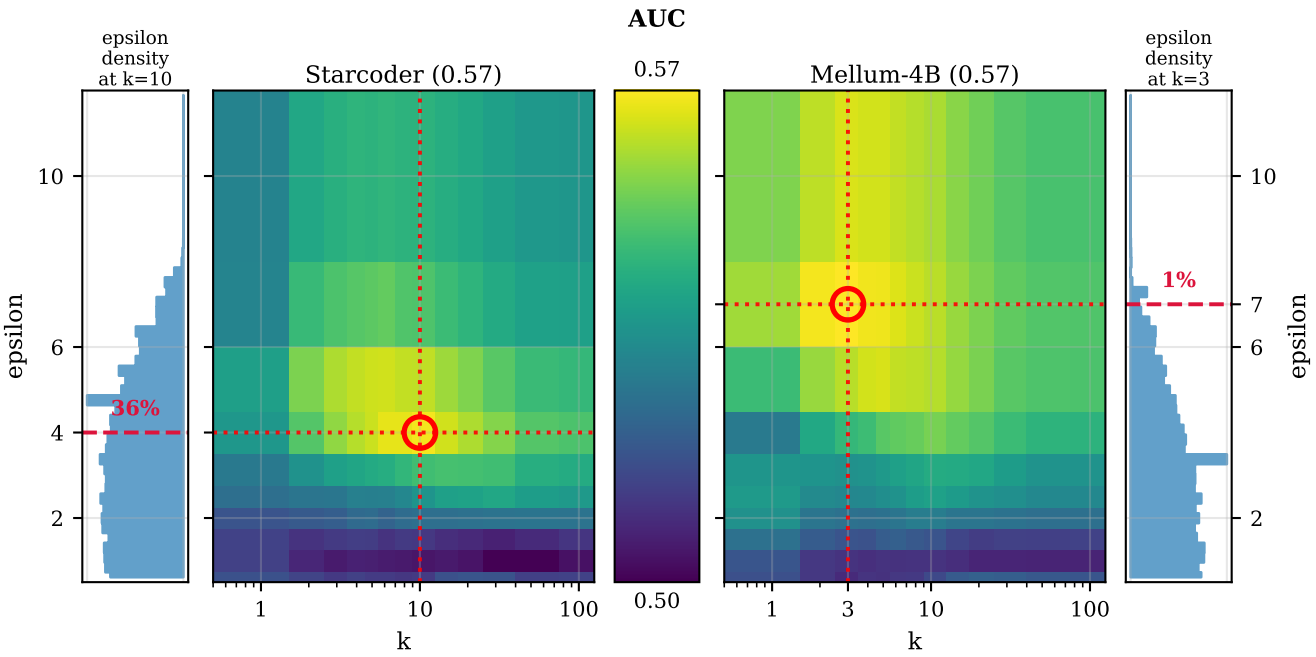

Figure 5: Caption

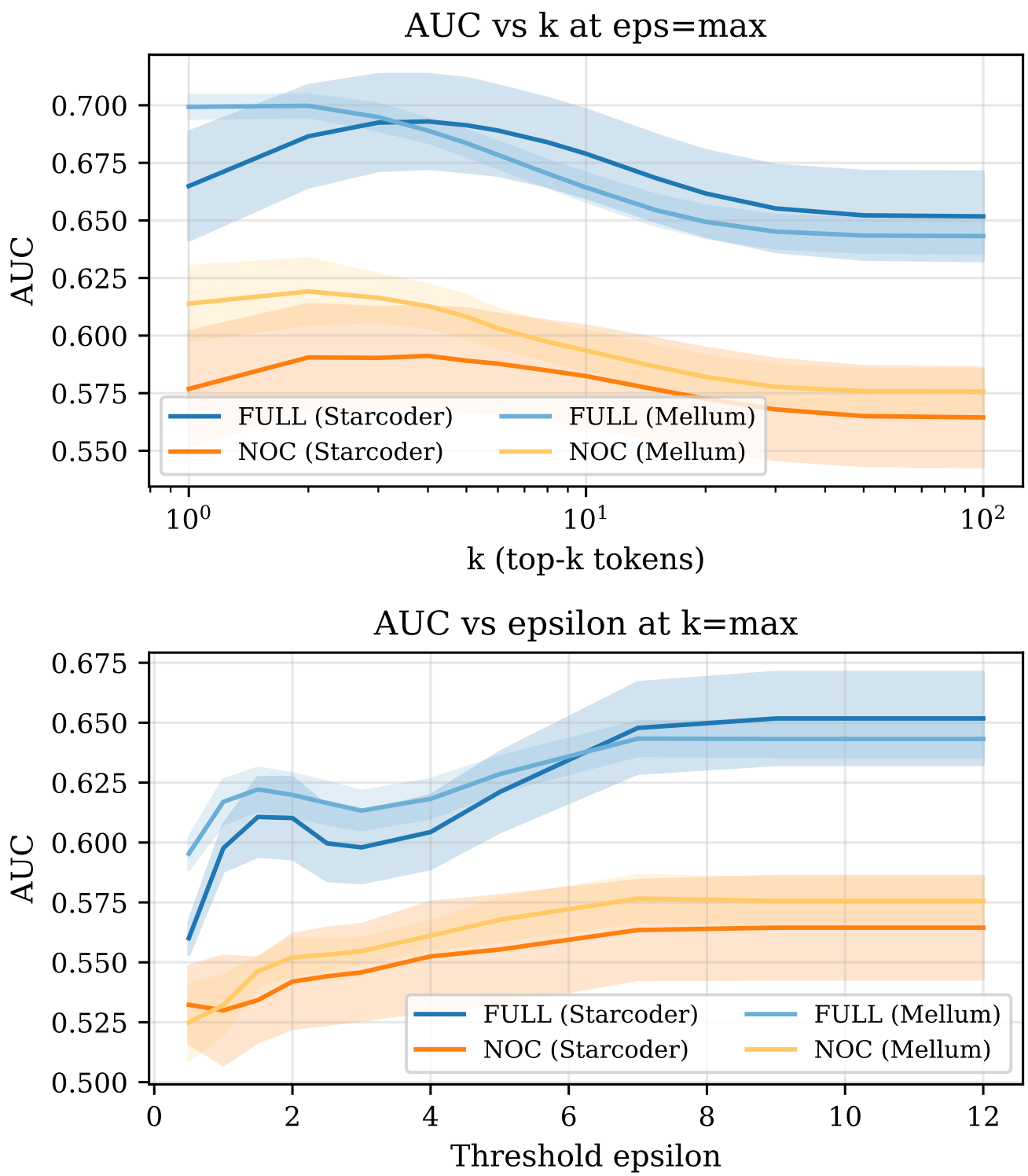

Figure 6: Caption

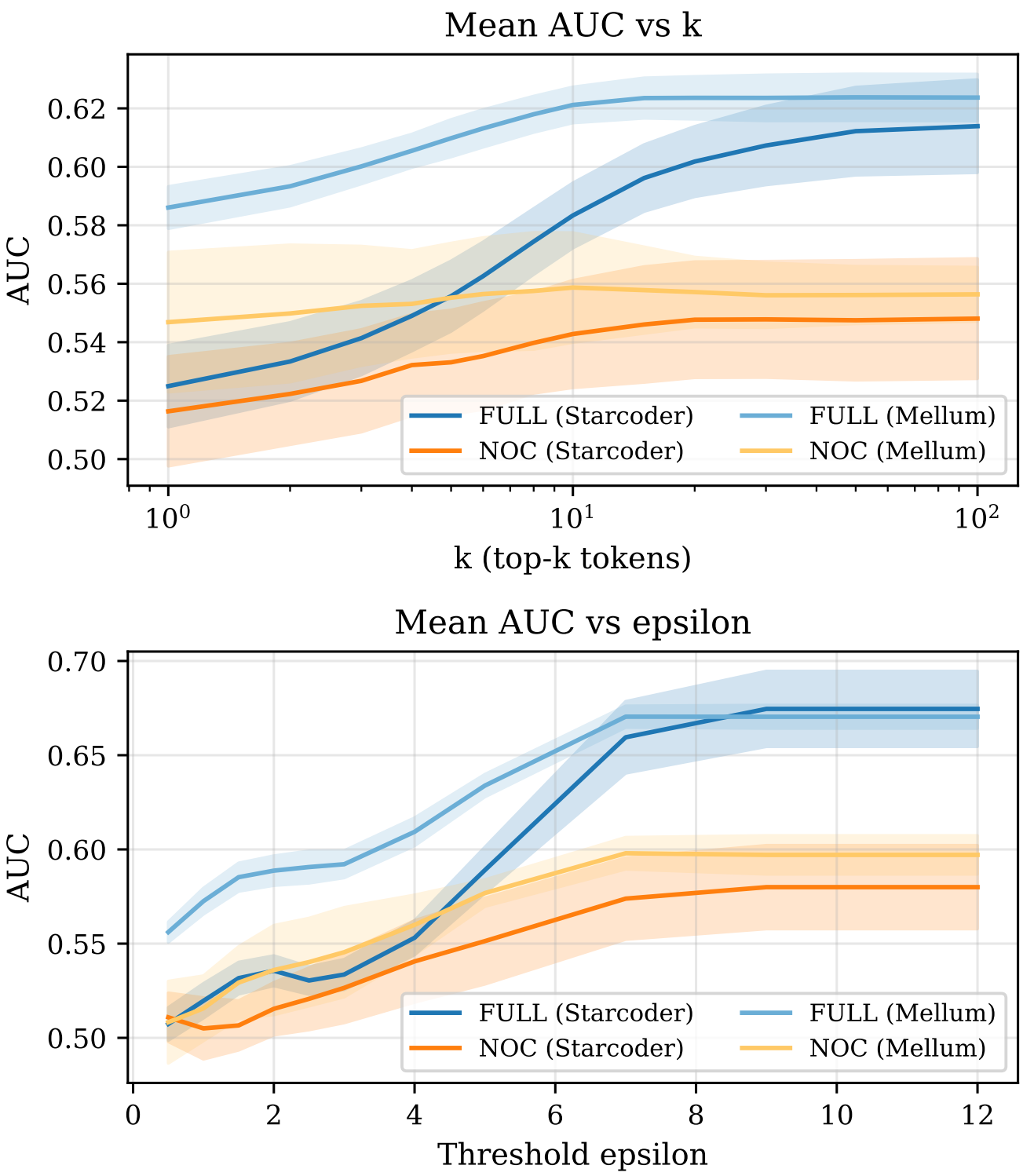

Figure 7: Caption

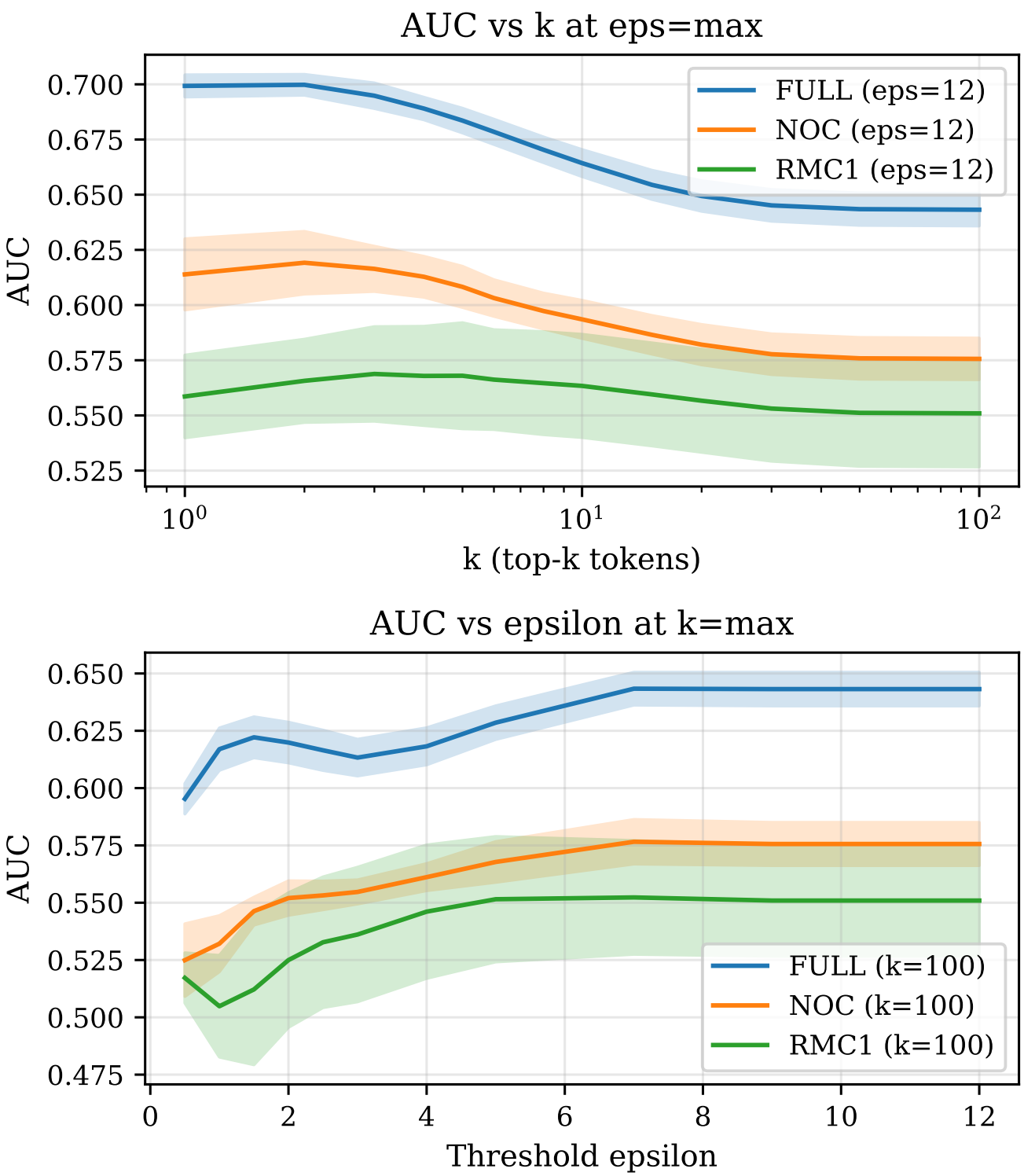

Figure 8: Caption

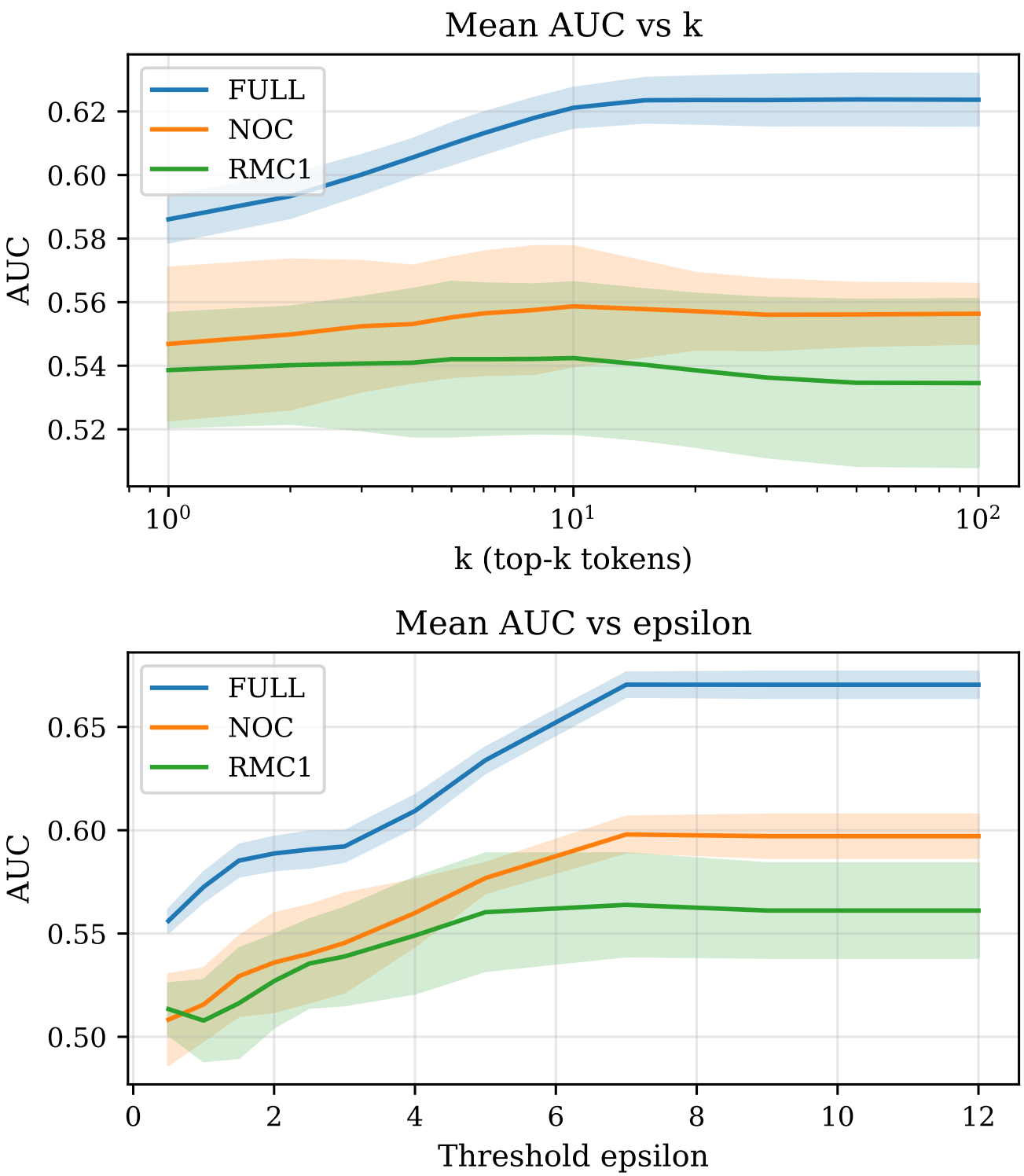

Figure 9: Caption

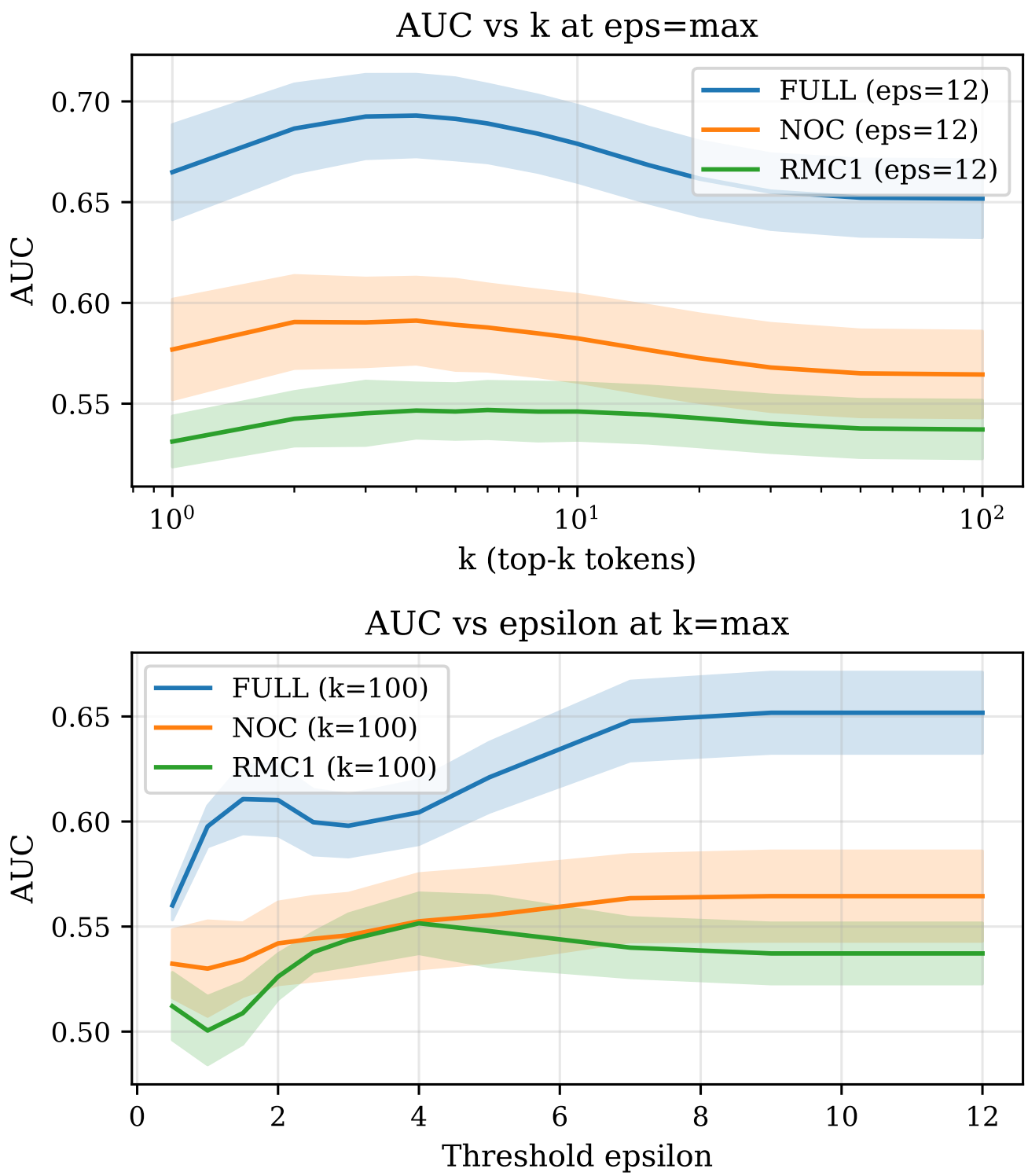

Figure 10: Caption

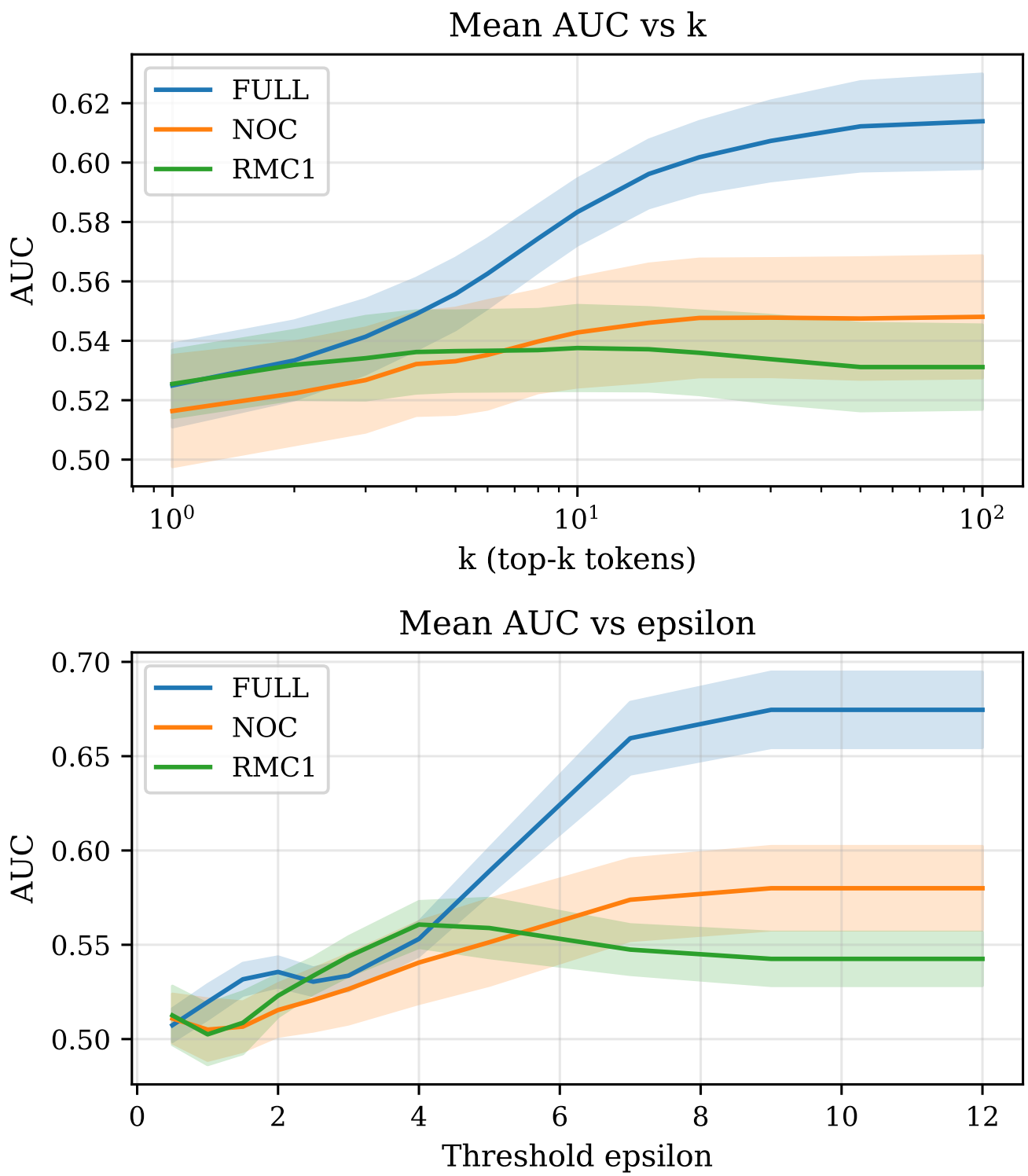

Figure 11: Caption