Randomization Techniques to Mitigate the Risk of Copyright Infringement

Wei-Ning Chen [1] Peter Kairouz [2] Sewoong Oh [2 3] Zheng Xu [2]

Abstract

In this paper, we investigate potential randomization approaches that can complement current practices of input-based methods (such as licensing data and prompt filtering) and output-based methods (such as recitation checker, license checker, and model-based similarity score) for copyright protection. This is motivated by the inherent ambiguity of the rules that determine substantial similarity in copyright precedents. Given that there is no quantifiable measure of substantial similarity that is agreed upon, complementary approaches can potentially further decrease liability. Similar randomized approaches, such as differential privacy, have been successful in mitigating privacy risks. This document focuses on the technical and research perspective on mitigating copyright violation and hence is not confidential. After investigating potential solutions and running numerical experiments, we concluded that using the notion of Near Access-Freeness (NAF) to measure the degree of substantial similarity is challenging, and the standard approach of training a Differentially Private (DP) model costs significantly when used to ensure NAF. Alternative approaches, such as retrieval models, might provide a more controllable scheme for mitigating substantial similarity.

1. Introduction

Modern machine learning relies heavily on large amounts of high-quality training data, primarily obtained from the Internet. Inevitably, these large-scale datasets contain some copyrighted material. When models are trained on this copyrighted data, they can accidentally generate outputs that closely resemble the training data, leading to potential copyright infringement. For example, despite recent advancements in foundational models (Bommasani et al., 2021), studies have shown that these models can easily memorize substantial portions of their training data (Carlini et al., 2021; 2023).

This immediately leads to the following questions: How do we define copyright infringement for models trained on potentially copyrighted data? How can one claim that a model violates copyright laws? And how can we prevent the models from generating outputs that resemble significantly copyrighted data? Copyright laws aim to promote creativity while protecting the rights, often economically, of original works. Note that copyright infringement class-action cases can be lucrative, which accounts for their popularity compared to, for instance, privacy violation cases. Under the fair use doctrine (Office, 2022), reproducing copyrighted work can be considered fair use based on four factors: the purpose of the use, the nature of the work, the amount of similarity, and potential harm. While purpose, nature, and harm are outside the scope of engineering solutions (see, for example, Sag (2018); Sobel (2017) for a discussion on whether data mining and machine learning on copyrighted text falls under "fair use"), substantial similarity can potentially be addressed with technology.

Producing outputs substantially similar to copyrighted work on which a foundation model is trained can be a key factor in determining whether it constitutes fair use. Current ongoing lawsuits that do not demonstrate that foundation models generate substantially similar works are likely to be dismissed. Determining the amount and substantiality of the portion of the generated text in relation to the copyrighted work is subjective and ambiguous as courts look at both quantity and quality. Fair use is less likely to be found if the use includes a large portion of the copyrighted work. However, some courts have found the use of an entire work to be fair under certain circumstances. In other contexts, using even a small amount of a copyrighted work was determined not to be fair because the selection was an important part, or the "heart", of the work, e.g., Ford vs. Nation magazine. Despite such challenges, there have been attempts to come up with quantifiable metrics of substantial similarity and corresponding guidelines.

Measuring substantial similarity. Suppose we have an oracle such that when presented with an original work x and an allegedly-infringing work y, outputs a binary decision $\mathsf{sim}(x, y)$ on whether they are substantially similar or not. This naturally leads to the following output filtering approach (e.g., Xu et al. (2021))to copyright protection: After generating a text output y, one enumerates all copyrighted works to check for substantial similarity. Variations of output filtering are present in most large language model services for various reasons, including copyright, safety, alignment, etc.

Divergence-based metric. Moving away from the typical output filtering framework, Scheffler et al. (2022) introduced a new framework in the context of comparing two computer programs in an attempt to make the notion of substantial similarity formal. The main idea is to use the minimum description length of a program to generate the allegedly-infringing work y, with and without access to the original work x. If the description lengths do not differ more than some threshold, then the contribution of x in generating y is small, and one can assert that the work has enough novelty and merits fair use. Although minimum description lengths are not easy to compute, the idea of comparing two scenarios with and without access to the original work was a major step forward in formalizing substantial similarity, which led to several important new approaches. This has obvious parallels with differential privacy, where paired datasets are tested on whether a piece of sensitive information was included or not. This connection between copyright and DP was initially suggested in Bousquet et al. (2020). However, there are challenges that are commonly shared among the work that follows this approach of paired scenarios.

- **Difficulty in specifying the unit of original work**: When we compare the current trained model against one that leaves a piece of copyrighted work out, the designer is left to choose how much of a work to leave out. Should it be one volume in the Harry Potter series, the entire series, or a chapter in a book? Since the expressions, and not the styles or ideas, are copyrighted, Vyas et al. (2023) argues that smaller units that qualify as an isolated piece of expression should suffice. For example, a painting as opposed to the entire collection of an artist. This choice is important but will likely evolve over time, and algorithmic solutions should be flexible to changes in the choice of the unit of copyright. For now, it can be treated as a hyper parameter to be chosen at the training time.
- **Derivative works:** A related concern is that internet-scale data is interconnected. Excerpts, quotes, and fan fictions are abundant and it is impossible to completely isolate all derivative work of an original work, which most approaches in this direction require.

Some preprocessing to identify the boundaries of each original work in an internet-scale dataset is necessary for copyright protection techniques, which might be orthogonal to the technical solutions investigated in this paper as long as they are flexible to changing boundaries.

In this paper, we first overview recently proposed techniques to mitigate the production of substantially similar outputs, examine their merits and weaknesses, and then evaluate and enhance these techniques by leveraging the potential for randomization in modern language models.

2. Prelimenary on Near Access-Freeness

A crucial property of typical modern generative models is that the outputs are generated randomly. There are various techniques to sample the outputs, but most of them involve some randomness. Leveraging this randomness, Vyas et al. (2023) proposed Near Access-Freeness (NAF) as a quantifiable metric for determining substantial similarity. This hinges on the inherent randomness in modern generative models, where the output is sampled from some distribution. This is based on the divergence between the output distribution of the potentially infringing language model and a *safe* model that does not have access to the original work in question.

Specifically, (Vyas et al., 2023) use the abstraction of a function safe that maps a datapoint $C \in \mathcal{C}$ into a generative model $\mathsf{safe}(C) \in \mathcal{M}$ that is assumed to have been trained without any access to C. For example, the leave-one-out-safe function is one such example. In this construction, the safe model is trained on all data except for C.

Since $\mathsf{safe}(C)$ is a generative model that was learned without access to C, in many realistic scenarios, the probability that $\mathsf{safe}_C(|x)$ generates material that is similar to C itself will be exponentially small in the length of C. Moreover, even if this unlikely event happened, this generation can be said to be fortuitous.

Formally, (Vyas et al., 2023) defines the following notion of NAF:

Definition 2.1. Let $\mathcal{C}$ be a set of copyrighted data points (i.e., samples) and $\mathcal{M}$ be a collection of (trained) models. Let $\mathsf{safe} : C \to \mathcal{M}$; and let Δ be a divergence measure between distributions. We say that a generative model p is k_x-near access-free (k_x-NAF) on prompt $x \in \mathcal{X}$ with respect to $\mathcal{C}$, safe, and Δ if for every $C \in C$,

$$\Delta\left(p(\cdot|x) \| \mathsf{safe}_C(\cdot|x)\right) \leq k_x.$$

If a model $p(\cdot|x)$ satisfies NAF with $k_x = 0$, then it is exactly the same as a safe model. Any generation of text that is substantially similar to a copyrighted work is by chance, and this chance is the same as a model that has never seen the original work. Therefore, it is safe to claim that the model $p(\cdot|x)$ is not infringing copyright. Generally, if a generative model satisfies NAF with a small k_x, then one can claim it is less likely that the model is outputting something substantially similar with a probability that is much larger than a random chance.

2.1. Achieving Near Access-Freness

In Vyas et al. (2023), two algorithms, CP-Δ and CP-κ, are provided to achieve provable NAF guarantees, which we briefly overview in the following.

The CP-Δ algorithm (Algorithm 1) can be viewed as a model ensemble method, which combines multiple models trained on different partition of the training data and is used to protect from copyright infringement.

Algorithm 1 CP-Δ: Copy Protection w.r.t. divergence Δ (Algorithm 3 of Vyas et al. (2023)

```
Require: Dataset $\mathcal{D}$, and divergence $\Delta \in \{\Delta_{\max}, \Delta_{\mathsf{KL}}\}$.
    Partition $\mathcal{D}$ into two disjoint sets $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ (ideally with similar size).
    Train two safe generative models $\mathcal{M}_1 = q_1(\cdot|x)$ and $\mathcal{M}_2 = q_2(\cdot|x)$ with $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, where $x$ is the prompt to the model.
    For any given prompt $x$, generate sample $y$ according to
        $p(y|x) = \begin{cases} \frac{\min\{q_1(y|x), q_2(y|x)\}}{Z(x)}; \\ \frac{\sqrt{q_1(y|x) \cdot q_2(y|x)}}{Z(x)}, \end{cases}$
    where $Z(x)$ is the partition function (i.e., the normalization constant).
```

While CP-Δ is proven to be k_x-NAF for $k_x \leq -\log(1 - \mathsf{TV}(q_1(\cdot|x), q_2(\cdot|x)))$ under $\Delta_{\max}$ and $k_x \leq -2\log\left(1 - \mathsf{H}^2(q_1(\cdot|x), q_2(\cdot|x))\right)$ under Δ_{KL} (Vyas et al., 2023, Theorem 3.1), there are several drawbacks that make CP-Δ infeasible in many practical deployments. First, implementing CP-Δ may be computationally difficult, especially if q_1 and q_2 are autoregressive models for text sequences or diffusion models for image generation[1]. Second, the bound on k_x is a model-dependent quantity that is hard to calculate or estimate in practice. Therefore, a more flexible algorithm, CP-κ, is proposed.

Algorithm 2 CP-κ: Access-Free Reduction at Threshold κ (Algorithm 4 of Vyas et al. (2023)

```
Require: a model $p$, a set of safe models safe, a threshold $\kappa$.
    Sample $y \sim p(\cdot|x)$ and return $y$ if
        $\forall q \in \mathsf{safe},\ \log(p(y|x)/q(y|x)) \leq \kappa.$
```

By introducing a tunable parameter κ and adopting rejection sampling, CP-κ resolves the computation issue in generating a sample, provided the threshold κ is sufficiently high.

[1]Note that, however, one can implement CP-Δ with rejection sampling and partially circumvent the computational issue. See Section 4.1 for details.

In addition, under $\Delta_{\max}$, CP-κ achieves the k_x-NAF for any $k_x > \kappa + \log(1/\nu_\kappa(x))$, where $\nu_\kappa(x)$ is the probability that a sampled y is accepted in a single iteration of the while loop in Algorithm 2 (Vyas et al., 2023, Theorem 3.5). Although $\nu_\kappa(x)$ cannot be computed exactly, it can be accurately estimated by repeating the algorithm assuming κ is not too large. his approach provides a method to obtain a probabilistic upper bound on the model-dependent NAF guarantee k_x.

2.2. Connection to differential privacy

There are obvious connections between NAF and DP (Elkin-Koren et al., 2023). Different choice of Δ corresponds to different variants of DP (e.g., the standard ε-DP in Dwork et al. (2006) when $\Delta = \Delta_{\max}$ or $(1, \varepsilon)$-Rényi DP in Mironov (2012) when $\Delta = \Delta_{\mathsf{KL}}$). Adopting the original notion of DP to the context of generative models, we get the following definition:

Definition 2.2 (Differentially private generation)**.** For two neighboring datasets S and S', let $P_S(\cdot|x)$ denote the probability distribution of a generated text on input prompt x when the generative models are trained on a dataset S with an algorithm $\mathcal{A}$, where the randomness includes the internal randomness in the training algorithm $\mathcal{A}$ and the randomness in the generation. We say this generation of a single output is an ε-Differentially Private Generation (ε-DPG) if for every neighboring dataset S and S', every prompt $x \in \mathcal{X}$, it holds that

$$\Delta\left(P_S(\cdot|x) \| P_{S'}(\cdot|x)\right) \leq \varepsilon,$$

where Δ is some divergence (e.g., $\Delta_{\max}$ or Δ_{KL}).

Recall that two datasets S and S' are called neighbors if they only differ in one unit of privacy, which is typically one sample but could be larger depending on the context. If a single generative model is trained with ε-DP (i.e., the standard DP applied to the *trained model*), then any generated text satisfies ε-DPG by the data-processing inequality. The advantage ε-DPG is that it allows one the flexibility to add randomness at the generation stage rather than the training stage, which can potentially give a significant gain in the utility-privacy tradeoff. However, there are a few major differences between the standard ε-DP and ε-DPG. First, for ε-DPG, multiple generated texts can eventually reveal any private data in training, whereas ε-DP protects against any number of generations. Next, the generative model can be shared under ε-DP, whereas only the generated text can be shared under ε-DPG.

Elkin-Koren et al. (2023) points out a few differences between NAF and DPG. First, NAF is one-sided whereas DPG is symmetric. This can give some flexibility in designing algorithms that achieve better utility under the one-sided NAF. Next, NAF allows more flexibility in selecting what safe model to use in the definition. Given the similarity between the definitions of NAF and DPG, it is natural to build upon this connection and consider using DP or DPG algorithms to achieve NAF. In this paper, we propose to leverage tools from DP to address issues in previous methods that guarantee NAF.

2.3. Challenges in previous solutions and summary of our contributions

Although the solutions proposed in Vyas et al. (2023), such as $\mathsf{CP}\text{-}\kappa$ and $\mathsf{CP}\text{-}\Delta$ (see Section 2.1), have resolved some of the computation issues, there are still challenges that need to be further addressed:

- *Computational challenges for verifying NAF:* Even for a given instance of $(\mathsf{C}, \mathsf{safe}, \Delta, x)$ and an arbitrary generative model $p(\cdot|x)$, the computational cost for checking the NAF condition generally scales with the support of the output of the language model. This is astronomical for the large language models we are interested in. Furthermore, the autoregressive nature of the decoding process makes this even more challenging. Approximating it with a truncated support is problematic, as typical choices of divergences (such as Δ_{KL} or Δ_{max}) are sensitive to the tail of the distribution. This makes it difficult to compare two generative models with respect to their respective achieved NAF, even if a reference safe model and a prompt x are given.

  While efficient estimation of k_x is possible in some limited scenarios (such as $\mathsf{CP}\text{-}\kappa$), in general, the estimation scheme does not extend to a general model $p(\cdot|x)$. Note that in $\mathsf{CP}\text{-}\kappa$, additional rejection sampling is required, so the final model, so the final model $p_\kappa(\cdot|x)$ should be treated as an ensemble of p and all the safe models. Since the NAF guarantee k_x is a data- and model-dependent quantity (i.e., it depends on the prompt x, the safe models, and the reference model $p(\cdot|x)$, and should be denoted as $k_x\,(p; \mathsf{safe}, x)$), ideally, we want to have a "audit" scheme that provides an empirical estimate of $\hat{k}_x$ with sufficient confidence.

- *Unclear advantage over DP-based methods:* While Vyas et al. (2023); Elkin-Koren et al. (2023) identified key differences between copyright protection and differential privacy, mathematically, DP remains a stricter criterion compared to NAF. Specifically, k-(model) DP implies k-DPG (for any prompt x), which in turn implies k-NAF. In Vyas et al. (2023), the NAF guarantees of the generative models range from 10^1 to 10^3, making it unclear whether, under such a large privacy budget, DP-based methods are still strictly worse, in terms of the utility, than the proposed NAF algorithms like $\mathsf{CP}\text{-}\Delta$ and $\mathsf{CP}\text{-}\kappa$. Additionally, the DP-based solution provides a *worst-case* NAF guarantee independent of the prompt x and safe models, which can be favorable in some scenarios. One can achieve a "safer" guarantee (e.g., smaller ε for DP, or, effectively, smaller k in NAF) by adjusting the injected noise accordingly. On the other hand, existing NAF algorithms solely rely on the stability of the safe models, so when the safe models do not align with each other (i.e., when the divergence between $q_1(\cdot|x)$ and $q_2(\cdot|x)$ is large), it may be impossible to achieve a pre-specified strict NAF guarantee.

- *The case of substantially similar outputs:* NAF is defined over (the distribution of) all outputs and not just those similar to the original work c of interest. This choice is unnecessarily pessimistic, and a crucial aspect of the challenge has been forgotten; fair use only concerns the substantial similarity in the expression of the output. An earlier approach Scheffler et al. (2022) is deterministic and only checks for outputs similar to c while still comparing two programs with and without access to the original work c. Scheffler et al. (2022) avoid explicitly specifying what constitutes as substantially similar by using Minimum Description Lengths (MDL) and the resilience that comes with this metric. However, one pays the heavy computational cost of computing MDLs. Furthermore, there is no efficient algorithm that guarantees a desired level of the MDL-based metric.

- *Difficulty in specifying the reference safe models:* NAF assumes a safe model is given by an external entity. In a fair use case, if NAF is to be used, the defendant will need to produce a language model that (1) did not access the original copyrighted work and (2) outputs text with distribution close to the allegedly infringing language model. The fact that the defendant can choose different safe models for each prompt x and each original work c makes the notion of NAF unreliable. One with more resources could come up with better, safer models and claim a smaller NAF. This brittleness in the definition safe model leaves NAF open to criticism. On the other hand, there are algorithmic approaches that ensure NAF with specific safe models that could be concrete or theoretical.

Our contributions. In this work, we aim to address the first and second challenges by proposing improved solutions and conducting comprehensive experiments to evaluate them empirically. First, we compare the performance of $\mathsf{CP}\text{-}\kappa$ (Algorithm 2) with a DP-based solution (trained with DP-FedAvg (McMahan et al., 2016)) on a next-token prediction task. Next, we evaluate how $\mathsf{CP}\text{-}\kappa$ and $\mathsf{CP}\text{-}\Delta$ can mitigate memorization in a fine-tuning task. As a by-product, we propose a Monte Carlo method to empirically estimate the NAF guarantee (i.e., k_x) for sentence-level generation. Finally, to further achieve a stricter NAF guarantee (e.g., stricter than the k_x that CP-Δ or CP-κ provides), we propose adding additional randomization into the generation process, such as increasing the decoding temperature, performing randomized response, or interpolating with an ε-DP model.

3. Methodology and Empirical Evaluations

In this section, we conduct experiments to evaluate prior methods and propose solutions to address the challenges described in Section 2.3.

3.1. Monte Carlo method for estimating NAF guarantees

We start by introducing the computation and estimation of NAF guarantees, specifically k_x. For token-level generation, the divergences Δ_{KL} and Δ_{max} between two models (e.g., outputs from CP-Δ/CP-κ and safe models) are tractable and can be computed exactly. However, for sentence-level generation, the computational cost grows exponentially fast, i.e., $O(K^T)$ where K is the number of tokens and T is the sentence length.

While some methods, such as CP-κ (with a sufficiently high threshold κ), can yield an efficient and straightforward estimate of the corresponding k_x, these estimators are typically tailored to the specific algorithm and do not extend to general scenarios. Therefore, we propose using the following Monte Carlo estimator: for $\Delta = \Delta_{\mathsf{KL}}$, a model $p(\cdot|x)$ and pre-specified safe models $\mathsf{safe} = \{q_1(\cdot|x), q_2(\cdot|x), ..., q_m(\cdot|x)\}$,

$$\hat{k}_x\left(p; \mathsf{safe}, x\right) \triangleq \max_{j\in[m]} \hat{\Delta}_j, \text{ where} \tag{1}$$

$$\hat{\Delta}_j \triangleq \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{p(y_i|x)}{q_j(y_i|x)}\right),$$

and $y_1, ..., y_n$ are independent samples generated from $p(\cdot|x)$. Note that this yields a consistent estimate of the true upper bound (i.e., k_x), as each term of $\hat{\Delta}_j$ is an unbiased estimator of the divergence. Moreover, we can reduce the variance by replacing these estimates with

$$\hat{\Delta}_j \triangleq \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{p(y_i|x)}{q_j(y_i|x)}\right) - \left(\frac{p(y_i|x)}{q_j(y_i|x)} - 1\right),$$

which ensures that $\hat{\Delta}_j$ is always positive.

When $p \ll q_j$ (e.g., when p is obtained from CP-Δ or CP-κ), one can further apply the empirical Berstein inequality (Maurer and Pontil, 2009, Theorem 3) to derive a high probability bound, which we state in the following lemma:

Lemma 3.1. *Assume $p(\cdot|x) \ll q_j(\cdot|x)$ for all $j \in [m]$. If $p(y|x) \geq \alpha$ and $q_j(y|x) \geq \alpha$ for all $y \in \mathsf{supp}\left(q_j(\cdot|x)\right)$, then it holds that, with probability $1-\delta$*

$$\left|\hat{k}_x - k_x\right| \leq \sqrt{\frac{8\max_{j\in[m]} V_n\left(r_j^n\right)\log(1/\delta)\log^2(m/\alpha)}{n}} + \frac{14\log(2/\delta)\log(m/\alpha)}{3(n-1)}, \tag{2}$$

where $k_x \triangleq \max_j \Delta_{\mathsf{KL}}(p(\cdot|x), q_j(\cdot|x))$, $r_j^n \triangleq (p(y_i|x)/q_j(y_i|x))_{i\in[n]}$ and $V_n(r_j^n)$ is the sample variance of r_j^n.

The proof follows from a simple application of the empirical Berstein inequality, together with the fact that $|\log(p(y|x)/q_j(y|x))| \leq 2\log(1/\alpha)$. This implies that the sample complexity is roughly $O\left(\log^2(1/\alpha)\right)$ (when other factors are constant).

For general language models, α can be arbitrarily small as there is no constraint on the generation probability. However, if one adopts popular techniques such as nucleus sampling (Holtzman et al., 2019) or top-p decoding (where we only keep the largest tokens whose cumulative probability exceeds p at each stage of decoding), an upper bound on α can be obtained:

$$\alpha \geq \left(\frac{1-p}{K}\right)^T,$$

where K is the token size and T is the sequence length. This suggests that the sample complexity of obtaining an accurate estimate is roughly $O\left(T^2\log(1/K)\right)$ [2].

3.2. Evaluating NAF bounds of CP-κ and CP-Δ

Comparison to (model) differential privacy. In the first experiment, we focus on the language generation task and compare methods based on differential privacy and CP-Δ. To better define the "unit" of the text corpus, we use the federated StackOverflow dataset[3], treating each user's data as a unit. We then train a 4M-parameter LSTM model (details are given in Appendix 4.2) on this data for the next-word prediction task with differential privacy and compare the results with those obtained using CP-Δ. For CP-Δ, we split the training data into two disjoint sets and train two safe models, respectively.

[2]While the sample complexity may still seem pessimistic, it is poly-logarithmic in the support size. For example, if we want to evaluate the NAF bound on a 20-token sentence generation, 100-1000 independent samples suffice.

[3]Admittedly, the model is less expressive than modern transformer-based architectures, given the constraints of available resources for DP/federated training. However, this experiment allows us to effectively assess the trade-offs between NAF and utility.

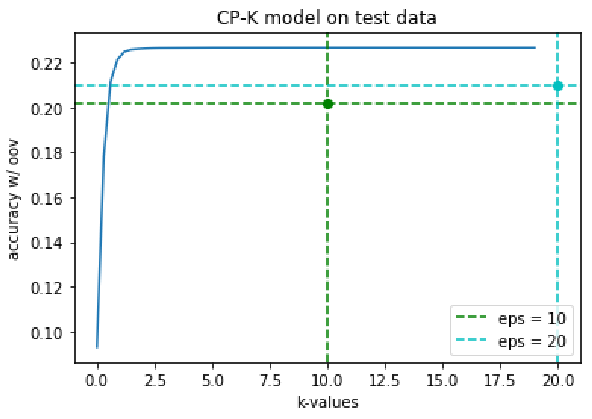


Figure 1. A comparison between k-NAF (with $\Delta_{\max}$) and ε-DP (with $\delta = 10^{-6}$).

NAF of CP-κ and CP-Δ on Federated Stackoverflow. Next, we visualize the k_x values of CP-Δ and CP-κ (with different thresholds κ) on the test data. Note that the k_x values here correspond to the divergence of the next-token prediction. We plot the histograms of k_x with different x's.

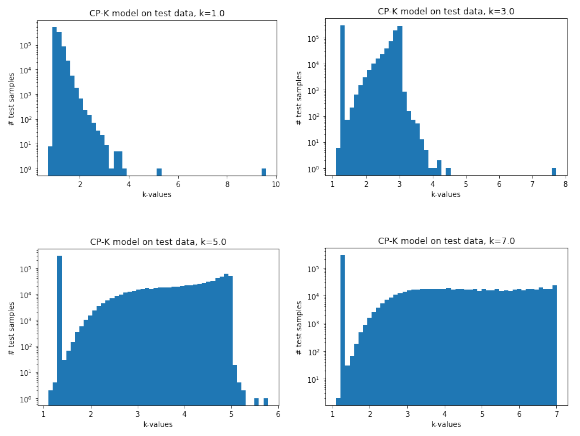


Figure 2. CP-κ with threshold $\kappa = \{1.0, 3.0, 5.0, 7.0\}$ We see that decreasing κ does not necessarily reduce k_x. For instance, when $\kappa = 1.0$, most k_x's are still greater than 1.0.

Sentence-level NAF for CP-Δ. In the next set of experiments, we evaluate the NAF bounds on a GPT-2 model fine-tuned on a PubMed dataset (Dernoncourt and Lee, 2017)[4]. We perform the Monte Carlo simulation to estimate the NAF bounds. Note that due to resource constraints, we did not perform re-sampling; instead, we calculated the empirical divergence (i.e., $\hat{\Delta}_j$ defined in (1)) and plotted the average $\hat{k}_x$ for different generation length. In Figure 3, we see that the bound scales are roughly linear with the length of the generated sequences.

[4]Note that the fine-tuned dataset is released after August 2022, later than the pre-trained GPT-2 model.

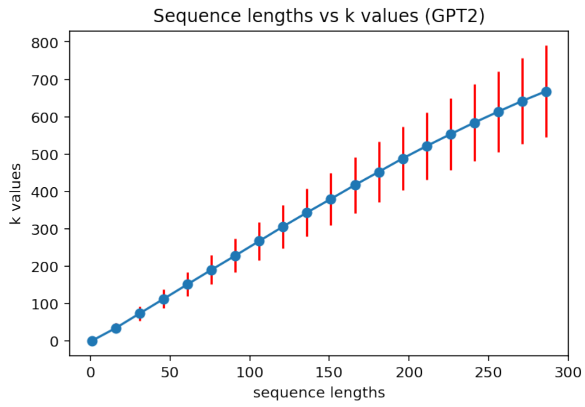


Figure 3. Sentence-level NAF bounds based on Marte Carlo estimators on a fintuned GPT-2. We use a token-level CP-Δ to ensemble models and plot the NAF with the sequence length.

3.3. Evaluating memorization for NAF

Next, we empirically show that CP-Δ and CP-κ can effectively mitigate memorization. In order to better demonstrate, we duplicate 600 training samples 40 times and trained a based model p (trained on all data without any protection) and two safe models (trained on half of the training data with duplication) q_1 and q_2. To measure memorization, we feed the initial 10 tokens of the duplicated samples and compute the normalized edit distances of the outputs to the correct answers. In Figure 4, we plot the histogram of distances with and without protection. Based on the plot, we see that token-level CP-Δ can effectively mitigate memorization for sequence generation.

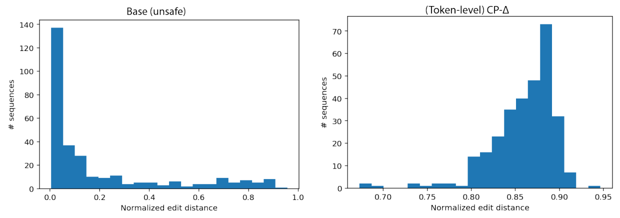


Figure 4. Normalized edit distance between the generated samples and the true samples (duplicated in the training phase).

3.4. Enhancing NAF via explicit randomization

Finally, we demonstrate that a "safer" guarantee can potentially be achieved by injecting additional randomization into the decoding process. Although the CP-κ algorithm includes a threshold parameter κ, its NAF guarantee is ultimately determined by the inherent distance between the

	Loss test	Acc test	Loss train 1	Acc train 1	Loss train 2	Acc train 2
Safe Model 1	3.4599	0.227	3.1235	0.256	2.7554	0.267
Safe Model 2	3.5625	0.218	3.2495	0.244	2.7999	0.264
All (unsafe)	3.4537	0.222	3.1159	0.257	2.7205	0.272
CP-Δ	3.4987	0.226	3.1804	0.253	2.7653	0.268

Table 1. Losses of safe models and CP-Δ models on training and testing datasets.

safe models $q_1(\cdot|x)$ and $q_2(\cdot|x)$. As shown in Figure 2, even if κ is set to a smaller value, the NAF bound k_x cannot be reduced arbitrarily. Therefore, to achieve a better bound, we need to increase the model's randomness by either injecting more noise or making the model less certain.

One simple approach to achieving this is by increasing the temperature during decoding. In Figure 5, we increase the temperature and plot the estimated sentence-level NAF. The results indicate that by appropriately increasing the temperature, we can obtain a better NAF guarantee.

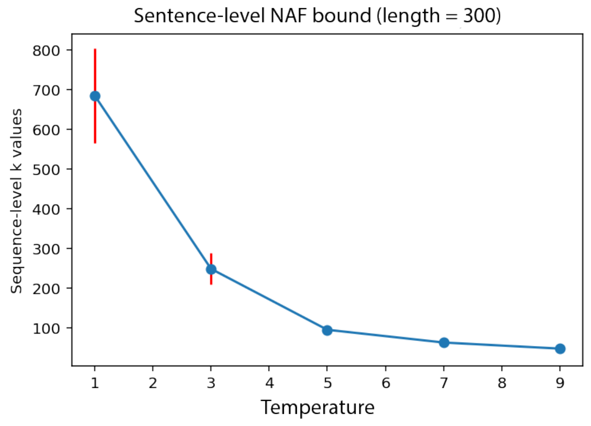


Figure 5. Sentence-level NAF bounds based with different decoding temperatures.

4. Conclusion

In this work, we propose an alternative method based on Monte Carlo simulation to evaluate the empirical NAF guarantees. We compare the performance of the CP-κ and CP-Δ algorithms and demonstrate how they can mitigate memorization in a fine-tuning task. To achieve a stricter NAF guarantee (e.g., stricter than the k_x provided by CP-Δ or CP-κ), we suggest incorporating additional randomization into the generation process, such as increasing the decoding temperature or performing a randomized response. Another promising direction for enhancing performance is to interpolate the outputs of a (possibly unsafe) model with an ε-DP model, which we plan to explore in future work.

References

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Olivier Bousquet, Roi Livni, and Shay Moran. Synthetic data generators–sequential and private. *Advances in Neural Information Processing Systems*, 33:7114–7124, 2020.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, 2023.

Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*, 2017.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

Niva Elkin-Koren, Uri Hacohen, Roi Livni, and Shay Moran. Can copyright be reduced to privacy? *arXiv preprint arXiv:2305.14822*, 2023.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

H Brendan McMahan, Eider Moore, Daniel Ramage, S Hampson, and BA Arcas. Communication-efficient learning of deep networks from decentralized data (2016). *arXiv preprint arXiv:1602.05629*, 2016.

Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661, 2012.

U.S. Copyright Office. Copyright law of the united states (title 17), 2022. URL `https://www.copyright.gov/title17/92chap1.html#107`.

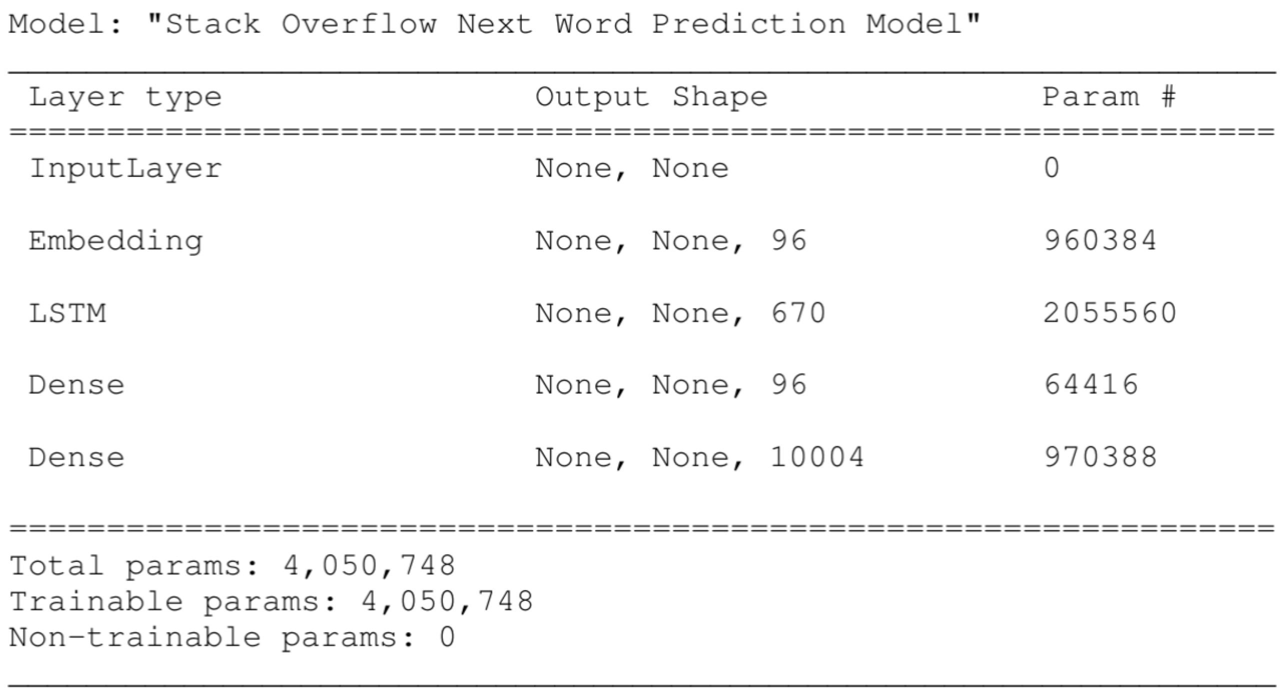
Matthew Sag. The new legal landscape for text mining and machine learning. *J. Copyright Soc'y USA*, 66:291, 2018.

Sarah Scheffler, Eran Tromer, and Mayank Varia. Formalizing human ingenuity: A quantitative framework for copyright law's substantial similarity. In *Proceedings of the 2022 Symposium on Computer Science and Law*, pages 37–49, 2022.

Benjamin LW Sobel. Artificial intelligence's fair use crisis. *Colum. JL & Arts*, 41:45, 2017.

Nikhil Vyas, Sham M. Kakade, and Boaz Barak. On provable copyright protection for generative models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35277–35299. PMLR, 23–29 Jul 2023.

Albert Xu, Eshaan Pathak, Eric Wallace, Suchin Gururangan, Maarten Sap, and Dan Klein. Detoxifying language models risks marginalizing minority voices. *arXiv preprint arXiv:2104.06390*, 2021.



```
Model: "Stack Overflow Next Word Prediction Model"
_________________________________________________________________
 Layer type                Output Shape              Param #
=================================================================
 InputLayer                None, None                0

 Embedding                 None, None, 96            960384

 LSTM                      None, None, 670           2055560

 Dense                     None, None, 96            64416

 Dense                     None, None, 10004         970388

=================================================================
Total params: 4,050,748
Trainable params: 4,050,748
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. Stack Overflow Next Word Prediction model architecture.

4.1. An improved CP-Δ for sequence generation

Algorithm 3 CP-Δ via rejection sampling for sequence generation

```
Require: Dataset $\mathcal{D}$, and divergence $\Delta \in \{\Delta_{\max}, \Delta_{\mathrm{KL}}\}$.
    Partition $\mathcal{D}$ into two disjoint sets $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ and train two safe models $q_1(\cdot|x)$ and $q_2(\cdot|x)$, respectively. Set
    $i \in \{1, 2\}$ uniformly at random. Also denote $i' \in \{1, 2\}$, $i' \neq i$. For any given prompt $x$, generate sample $y$
    according to the following rule: True Generate $y \sim q_i(\cdot|x)$. $\log(q_i(y|x)/q_{i'}(y|x)) \leq \kappa$ and $\Delta = \Delta_{\mathsf{max}}$ Return $y$.
    $\log(q_i(y|x)/q_{i'}(y|x)) \geq 1$ and $\Delta = \Delta_{\mathsf{KL}}$ Return $y$ with probability $\min\left(1, \sqrt{\frac{e^{\kappa} \cdot q_{i'}(y|x)}{q_i(y|x)}}\right)$.
```

Note that Algorithm 3 is similar to CP-κ algorithm but with $p(\cdot|x)$ being set to the mixture distribution of safe models. When $\kappa = 0$, Algorithm 3 precisely recovers CP-Δ; however, when the safe models do not align (i.e., the divergence between $q_1(\cdot|x)$ and $q_2(\cdot|x)$ is large), the re-sampling step may be the bottleneck.

4.2. Model Architecture of the 4M LSTM

The model architecture is presented in Figure 6.