The Unreasonable Ineffectiveness of Nucleus Sampling on Mitigating Text Memorization

Luka Borec[1], Philipp Sadler[1], David Schlangen[1,2]

[1]CoLabPotsdam / Computational Linguistics
Department of Linguistics, University of Potsdam, Germany
[2]German Research Center for Artificial Intelligence (DFKI), Berlin, Germany
Correspondence: firstname.lastname@uni-potsdam.de

Abstract

This work analyses the text memorization behavior of large language models (LLMs) when subjected to nucleus sampling. Stochastic decoding methods like nucleus sampling are typically applied to overcome issues such as monotonous and repetitive text generation, which are often observed with maximization-based decoding techniques. We hypothesize that nucleus sampling might also reduce the occurrence of memorization patterns, because it could lead to the selection of tokens outside the memorized sequence. To test this hypothesis we create a diagnostic dataset with a known distribution of duplicates that gives us some control over the likelihood of memorization of certain parts of the training data. Our analysis of two GPT-Neo models fine-tuned on this dataset interestingly shows that (i) an increase of the nucleus size reduces memorization only modestly, and (ii) even when models do not engage in "hard" memorization – a verbatim reproduction of training samples – they may still display "soft" memorization whereby they generate outputs that echo the training data but without a complete one-by-one resemblance.

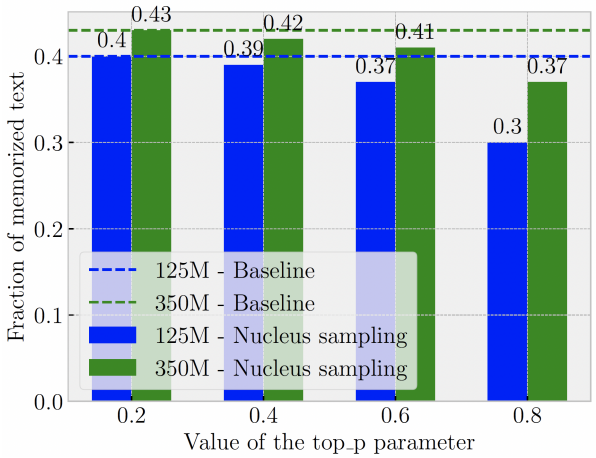


Figure 1: The effect of different `top_p` values (x-axis) on the fraction of the duplicated texts memorized by the models (y-axis). The `top_p` parameter determines the maximally considered accumulated probability mass for the output token selection during nucleus sampling. Higher `top_p` values generally lead to reduced memorization, yet the decrease is less significant than expected. This effect is observed across two models of different model sizes, with the larger model showing a somewhat less pronounced reduction in memorization compared to the smaller model. The dashed lines show the baseline behavior using greedy decoding.

1 Introduction

Recent developments in LLMs have led to impressive capabilities in generating human-like text. However, there is growing concern about these models' potential to memorize and regurgitate text from their training data, raising privacy, security, and copyright issues (Huang et al., 2022; Lee et al., 2023; Karamolegkou et al., 2023). These concerns culminated in a legal dispute between the New York Times and OpenAI which is largely based on the finding that the LLM "*can generate output that recites Times content verbatim, closely summarizes it, and mimics its expressive style*"[1].

And indeed Carlini et al. (2021) have observed qualitatively that GPT-2 can memorize data from which it was trained, such as HTML pages and logs, and later demonstrated that duplicated texts significantly contribute to memorization when deterministic decoding is at work (Carlini et al., 2023). Could the use of a probabilistic decoding technique like nucleus sampling have prevented the lawsuit?

In this paper, we analyze the impact of nucleus sampling (Holtzman et al., 2020) on the degree of text memorization. Nucleus sampling is notable for its ability to effectively blend randomness with a focus on likely outcomes. This decoding method operates by sampling from a truncated output distribution (the "nucleus") which includes only the

[1]`https://nytco-assets.nytimes.com/2023/12/NYT_Complaint_Dec2023.pdf`, *visited at: 29.05.2024*

Proceedings of the 17th International Natural Language Generation Conference, pages 358–370
September 23–27, 2024.

highest-probability tokens whose cumulative probability reaches a predefined threshold specified by `top_p` . While the method still focuses on the more probable tokens, it introduces randomness by allowing sampling among the tokens that are otherwise less likely to be generated. This makes nucleus sampling a good choice for our study as it aligns with our objectives to explore if and how stochasticity in decoding can mitigate text memorization.

We experiment with a range of nucleus sizes to measure their effects on a model's text memorization behavior (see Figure 1). However, quantifying this impact precisely for current very large models is challenging because enumerating duplicates in their training datasets (if they are even accessible) is computationally infeasible. To address this, we select a manageable portion of the OpenWebText dataset (Gokaslan and Cohen, 2019) and introduce duplicates in a controlled way. This allows us to precisely measure the influence of duplication on memorization, and the degree to which the choice of the decoding strategy can reduce it.

Our findings confirm the previously measured strong correlation between data duplication and memorization (Carlini et al., 2023) and deliver new insights about the effects of nucleus sampling: Small nucleus sizes produce effects similar to greedy decoding, and interestingly, even larger nuclei show an "unreasonable ineffectiveness" on the mitigation of text memorization, because in cases of peaked distributions a model's memorized token dominates the output distribution, so that even larger nuclei are highly susceptible to generate them. Our contributions are as follows:

1. We create OpenMemText, a diagnostic dataset based on OpenWebText (Gokaslan and Cohen, 2019) that contains a controlled number of copies to induce, measure and analyse the memorization behavior of LLMs.
2. We replicate the results from Carlini et al. (2022) with two GPT-Neo models (Black et al., 2021) of different sizes and our results show similar memorization trends with respect to (a) the models' size, (b) the number of duplicates, and (c) the length of the prefix.
3. We present a comprehensive analysis of the text memorization behavior of the models when using nucleus sampling instead of greedy decoding and find it to be surprisingly ineffective in mitigating text memorization.

2 Related work

Text Memorization in Large Language Models. Bender et al. (2021) raised concerns about the magnitude of LLMs, highlighting environmental and accessibility issues, but also noting that these models, much like parrots, tend to repeat the data they have seen during training, leading to issues such as amplifying biases. Magar and Schwartz (2022) evaluated pre-trained BERT models concerning data contamination and argued that a model's test performance may be inflated by the model's ability to memorize training examples and reproduce them almost verbatim at test time. And indeed Tirumala et al. (2022) found that larger models can memorize large portions of the text without showing overfitting signals. Hernandez et al. (2022) argue that the number of data duplicates induces a shift from generalization to memorization. Haviv et al. (2023) suggest probing for memorized text with specifically constructed English idioms and compare the models' behavior for memorized and non-memorized inputs. Zhang et al. (2023) propose counter-factual memorization and measure how the prediction of an LLM changes when specific pieces of information are not shown during training. Kandpal et al. (2023) confirm that LLMs are sensitive to the number of duplicates seen during training for fact-based question answering and found that deduplication mitigates privacy risks in language models (Kandpal et al., 2022). Marone and Van Durme (2023) introduce Data Portraits, which enable querying of training datasets for membership inference, deduplication, and overlap analysis.

Decoding Methods for Text Generation. Decoding methods transform the probabilistic outputs of language models into readable text. Traditional approaches like greedy decoding follow deterministic rules by choosing the highest probability word at each decision point. Although efficient, text generated in this way is often monotonous and predictable (Kulikov et al., 2019). Sampling-based methods and various decoding heuristics can enhance the diversity and richness of the generated text. Klein et al. (2017) propose n-gram blocking to further refine the output quality by preventing the repetitive generation of the same sequence. Garneau and Lamontagne (2023) propose an extension to beam search to mitigate hallucinations and omissions. A common decoding technique used with LLMs is temperature sampling (Ficler and Goldberg, 2017) which adds control over the uniformity

of the output distribution, so that a higher temperature leads to likely more versatile outputs because the overall distribution becomes more uniform.

3 Memorization Effects in GPT-Neo Models for Greedy Decoding

Carlini et al. (2023) uncovered log-linear relationships between memorization and model size, number of duplicates, and input length, respectively. In particular, they measured the effects of greedy decoding on the memorization behavior of GPT-Neo models using The Pile (Gao et al., 2021) dataset. But they could only approximate the impact of duplicates due to dataset's unknown duplicate count. Thus, while their study represents one of the most comprehensive quantitative analyses of memorization to date, their findings are based on estimates from their sampled data. In this section, we present the replication of their results using a diagnostic dataset that allows us to measure the amount of text memorization for greedy decoding more precisely.

3.1 OpenMemText: A Diagnostic Dataset for Text Memorization Research

Biderman et al. (2023) has shown that a highly controlled setup is fruitful for the analysis of LLMs and leads to novel insights. Following this paradigm, we create a modified version of the OpenWebText (Gokaslan and Cohen, 2019) dataset, an open-source replica of OpenAI's WebText that was used for GPT-2 training. OpenWebText contains texts from diverse platforms such as Reddit and news websites. It is 38 GB uncompressed and consists of over 8 million curated and *deduplicated* plaintext files each of which represents a separate data point (see Appendix A.3 for an example data point). Large datasets present significant challenges in measuring duplicates due to their vast size. However, the deduplicated nature of OpenWebText allows us to manually introduce a known number of duplicates with precise control over their distribution. This enables us to quantify the effect of duplicates in the data on a model's memorization behavior accurately without the computational burden of enumerating duplicates.

To create the dataset in a controlled way, we first sample 0.5% of the OpenWebText files at uniform random which amount to roughly 500K files. Then we introduce a balanced distribution of duplicates as follows: We select from the files 280 and duplicate each of them once, so that they appear twice in the dataset. Then we repeat this process by selecting from the remaining files another set of 280 data points and duplicate them twice, so that they appear three times in the dataset. We repeat this process, each time increasing the duplicate count, until we have files that appear 30 times. This results in approximately 680K data samples (4.4GB) for training, including 180K duplicates and 500K files that are not duplicated. We perform the same procedure for the validation set (1.4GB) by sampling 0.1% of the OpenWebText files after exclusion of the training samples which resulted in about 400,000 file.

3.2 Experimental Setup

First, we ensure that our experimental setup is correct by replicating the results from Carlini et al. (2023) with our newly proposed diagnostic dataset.

Model Selection. For reasons of comparison with the work of Carlini et al. (2023) we choose similarly two commonly available GPT-Neo (Black et al., 2021) models. These models have the same architecture as the GPT-3 (Brown et al., 2020) models and were also pre-trained on The Pile (Gao et al., 2021) dataset for over 400K steps seeing about 420 billion tokens. For our experimental purposes, we select the 125M and 350M parameter variants of GPT-Neo model family. Alongside these models, we use the pre-trained GPT-2 as a baseline for the effects of greedy search on the text memorization.

Model Fine-tuning. We shuffle the data points in our diagnostic dataset and fine-tune the GPT-Neo models for a single epoch on them. For the 125M model we use a batch size of 16 (distributed across four GPUs), and for the 350M model we use a batch size of 4. We use adaptive learning rate starting at $5e-4$ and employ half floating point precision (fp16) to enhance the fine-tuning efficiency. Based on findings by Mireshghallah et al. (2022) we specifically target the model's attention heads for fine-tuning and keep the rest of the parameters

Duplicity	# Data Points	# Files
Zero	$\approx 500,000$	1
$n-1$	280	$n \in [2, 30]$

Table 1: For our analysis, we create a dataset where about 500K files occur only once and 8120 samples are duplicated multiple times. As a result, in the majority of cases a data points occurs only once and we get a balanced distribution concerning the number of copies seen more than once (2 times up to 30 times).

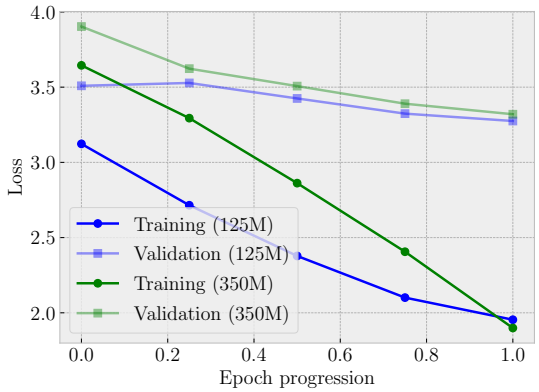


Figure 2: During fine-tuning we measure a consistent decrease in both training and validation loss which indicates that the GPT-Neo models are fitting better to the memorization dataset data over time.

frozen. The attention heads were found to be the most susceptible to memorization. We argue that a more effective fine-tuning method allows us to better measure how text memorization manifests in the language models compared to less susceptible methods. Figure 2 shows that the fine-tuning method is effective.

Model Evaluation. Carlini et al. (2023) define memorization as the behavior of a model f to reproduce an exact target string s from the training data TD when prompted with a certain number of context tokens p (the prefix) of length $\text{len}(s) - k$ such that $f(p) = s$. This behavior can be formalized as:

$$\begin{aligned} &\exists p\colon \text{len}(p) = \text{len}(s) - k \text{ and} \\ &\quad [p \,||\, s] \in \text{TD and} \\ &\quad f(p) = s \end{aligned} \tag{1}$$

where

- s represents the target string,
- p represents the context string with a length of $\text{len}(s) - k$,
- f is the model,
- TD denotes the training data for the model f,
- $[p \,||\, s]$ is the concatenation of the context string p with the target string s,
- and $f(p) = s$ signifies that the model f, when prompted with p, produces the string s.

We use this definition of memorization in our work as well. For instance, if a model's training dataset contains the sequence "*Twinkle, twinkle, little star, how I wonder what you are,*" and given the prefix "*Twinkle, twinkle, little star,*" the model outputs "*how I wonder what you are,*" this sentence would be considered memorized.

Replication Experiments. For the replication experiments we use all data points from the training dataset with a duplicity greater than zero (see Table 1). For each data point we prompt the model with an experiment specific number of context tokens p and use greedy decoding to generate tokens until an end-of-sentence token or a number of 512 tokens is produced (note that some samples only contain up to 200 tokens). We compare the resulting string s with the ground-truth in our training data and count the result as an instance of text memorization in accordance to Equation 1. In particular, we measure the memorization outcomes with respect to the following conditions:

(a) **Model Size:** This experiment explores how model size affects memorization. We use two models containing 125M and 350M parameters, respectively, and run the memorization experiment with a context length of $p = 150$. Our results confirm the findings by Carlini et al. (2023) that larger models tend to memorize more as GPT-Neo 350M memorized 43% of all duplicated data points whereas the 125M parameter model memorized only 40%.

(b) **Data Repetition:** This experiment is conducted in the same way as the one before, but measures the amount of memorization with respect to the number of duplicates. Our trends confirm the original findings by Carlini et al. (2023) that more duplicates lead to higher counts of memorized text. Furthermore, we find that the 350M parameter model memorizes faster, but both models start to saturate at similar levels.

(c) **Context Length:** This experiment is conducted as before, but we vary the context length p from 100 to 200, 200 to 300, 300 to 400, and 400 to 500, and over 500 tokens. The scores for each bucket are averaged across all duplicated files belonging to that bucket. While our results somewhat confirm the original paper's findings that an increase in memorization follows an increase in context length, there is a dip at the 300-to-400 length bucket. It is possible that this was caused by small sample sizes for each bucket (70 data points).

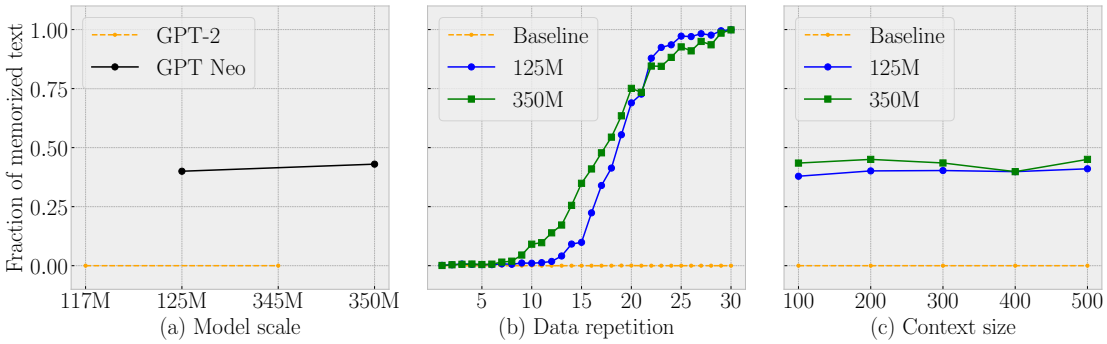


Figure 3: Results from our replication of Carlini et al. (2023). The two fine-tuned GPT-Neo models were compared to non-fine-tuned GPT-2 models of similar sizes using the same prompts. **(a)** The larger model memorized more of the training dataset than the smaller one. **(b)** Repeated data in the training set is more likely to be extractable. **(c)** There is a gradual increase in the extraction of memorized text as the length of input context increases.

Since our results as shown in Figure 3 match those of Carlini et al. (2023), we conclude that our experimental setup works and move on to our nucleus sampling experiment.

4 Analysing Nucleus Sampling-based Text Memorization Behavior

This section presents our analysis of text memorization behavior for the fine-tuned GPT-Neo models when using nucleus sampling instead of greedy decoding. In particular, we measure the amount of text memorization of the fine-tuned models under a variety of secondary conditions.

4.1 The Effect of Duplicates on Text Memorization under Nucleus Sampling

First, we are interested in the effect of the amount of data duplication on text memorization conditioned on various nucleus sampling thresholds. We conduct the experiments as described for the replication experiments, but with nucleus sampling and different `top_p` parameters $(0.2, 0.4, 0.6, 0.8)$ which determines the size of the nucleus from which the output token is sampled. For our analysis we group the measured amount of memorized text along with the according `top_p` values.

The resulting heatmap (see Figure 4) reveals that the larger model consistently shows a higher tendency to memorize across all `top_p` values. This means that the finding from Carlini et al. (2023) that larger models memorize more is also true for nucleus sampling, when all other variables are kept constant. Furthermore, we note an intriguing interaction between the duplicate count and the `top_p` parameter. Especially with high data repetitions (25 to 30 copies) memorization occurs irrespective of the `top_p` setting. Even with a `top_p` $= 0.8$ the amount of detected memorized text is nearly equivalent to that of the deterministic greedy search.

In contrast, with fewer data copies (up to 20), increasing the `top_p` value markedly reduces the amount of memorized content, creating a distinct gap compared to the greedy search which often extracts nearly double the amount.

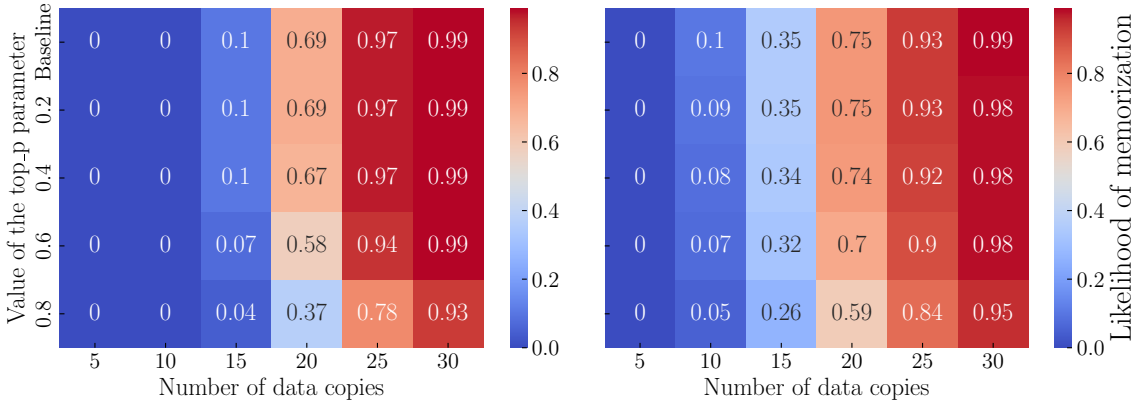
We conclude that more repetitions allow the models to better internalize sequences, boosting recall. Thus, even with large nuclei, output closely mirrors the training data, making the difference between greedy search and nucleus sampling minimal. However, with fewer data copies, models exhibit reduced memorization, leading to a greater disparity in content retrieval between greedy search and nucleus sampling with larger nuclei.

Finding 1: *At high data repetition, significant memorization occurs across all* `top_p` *values in nucleus sampling. However, with lower repetition, lower* `top_p` *values lead to higher memorization compared to higher* `top_p` *values.*

4.2 The Emergence of Ramp-up and Saturation Points

In our analysis we identify stages when a model starts to significantly memorize data from its training set and define these as *ramp-up points*. In addition, we identify *saturation point* as such when further data additions do not significantly improve learning, indicating diminishing returns.

We find these points prominently illustrated in the middle columns of Figure 4. During the decoding experiments with nucleus sampling, the memo-



(a) GPT Neo 125M (b) GPT Neo 350M

Figure 4: Heatmap illustrating the inverse relationship between top_p parameter values and extracted memorized text, modulated by the number of data repetitions in steps of five. It highlights the unexpected trend that for a high number of data copies, memorization levels remain significant for all top_p values, while fewer data repetitions lead to markedly lower memorization when top_p is increased, reflecting the models' shift from rote memory to learned generalizations.

rization rates of smaller models significantly "ramp up" from 10% at 15 duplicates to nearly 70% at 20 duplicates, eventually saturating at 93% at 25 duplicates. In the larger 350M GPT-Neo model, noticeable increases in memorization occur as follows: at 10 duplicates, memorization stands at 10%. This rises to 35% at 15 duplicates, further escalates to 75% at 20 duplicates, and peaks at 93% by 25 duplicates. We have a closer look at these ramp up points and provides a more detailed view for each duplicate count from 15 to 20 in Figure 5. Given this we see that in the case of GPT-Neo 125M , memorization remains minimal, with only 1.8% of data memorized up to 12 data copies. And already at 13 data copies the amount drastically doubles to 4.1%, and doubles again to 9% at 14 copies. GPT-Neo 350M shows a similar pattern. This illustrates how even a single increase in the number of duplicates significantly impacts memorization.

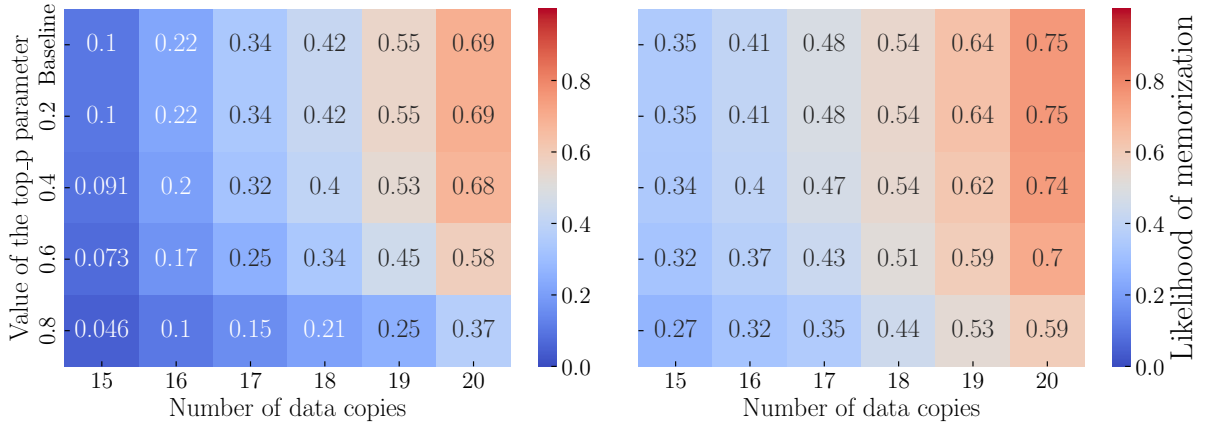
We find that especially at these pivotal *ramp-up points*, where a slight increase in duplicates leads to substantial increases in memorization, employing a larger nucleus size proves effective in reducing text memorization. However, once the models seem to reach a *saturation point*, the efficacy of increasing nucleus size to mitigate memorization diminishes significantly.

Finding 2: *Higher* top_p *values reduce memorization significantly at ramp-up points but are much less effective near saturation points where additional data yields diminishing returns.*

A closer look into the top_p values in Figure 5 and their effect on memorization rates fosters this finding. When looking at the numbers for the smaller 125M GPT-Neo model, then the transition from a more deterministic top_p of 0.2 to a more stochastic top_p of 0.8 significantly reduces memorization rates. The memorization decreases from 10% at top_p 0.2 to 4% at top_p 0.8 when considering 15 duplicates, and from 69% to 37% when considering 20 duplicates.

These levels can be considered ramp-up points where the difference between top_p 0.2 and 0.8 is substantial. However, at 25 duplicates, where the model appears to be reaching its saturation point, the memorization rates are 97% for top_p 0.2 and 78% for top_p 0.8 are showing a lesser though still notable reduction. In the larger 350M GPT-Neo model, this trend towards saturation is evident: for data points with 25 duplicates, the measured text memorization is at 93% under top_p 0.2 compared to 84% at top_p 0.8.

A possible explanation for this effects is the data density which significantly influence the dynamics of model behavior, especially regarding how quickly saturation points are reached. In datasets abundant with unique items, we would expect the models to experience delayed saturation due to the complexity and infrequency of duplicate data points. Conversely, our diagnostic dataset, rich in multiple copies, likely acts as a "forced attention" mechanism. This effect is particularly pronounced in the larger 350M GPT-Neo model which due to



(a) GPT Neo 125M (b) GPT Neo 350M

Figure 5: This more fine-grained view between 15 to 20 data copies delineates the ramp-up point where memorization begins to climb sharply and approaches the saturation point where further data addition has diminished effects on memorization rates. This illustrates how, despite increasing `top_p` values which typically reduce memorization, the presence of high repetition still results in substantial memorization, particularly in the GPT-Neo 350M model.

its higher capacity can better "incorporate" the duplicated data points and potentially reach the saturation points more swiftly.

4.3 The Disturbing Effects of Peak Distributions on Nucleus Sampling

We intensify our analysis and have a detailed look on the output distributions of our fine-tuned GPT-Neo models. We select four data points from the diagnostic training set which appear increasingly often (1, 5, 15, and 25 times) and measure the probability of the most likely token to be produced as shown in Figure 6. The results show that the models tend to assign a higher probability to the individual tokens which would lead to an exact continuation of the training text when such texts are seen more often during fine-tuning.

We also examine the differences in token-level probabilities between the tokens used as the context p and those generated by the model. Generated tokens are derived from a subset that the model predicts as most likely for the next position in the sequence. This typically results in higher probabilities for these tokens. In contrast, the probabilities of context tokens can vary widely, as they are not constrained to belong to a sorted group of tokens with cumulative probabilities meet a predefined threshold. For example, when the nucleus threshold is set to so that `top_p` $= 0.2$, then only tokens (or sometimes even just a single token) whose cumulative probabilities do not exceed the threshold are considered for selection. This effectively excludes other token from being generated. This pattern is illustrated in Figure 6, where such a selection process often occurs for a `top_p` of 0.2, especially as the number of duplicate tokens increases.

We conclude that using low `top_p` values is often less effective for mitigating memorization issues. This occurs because snippets that the model has memorized, which usually have high token-level probabilities, tend to dominate the selection process. When these probabilities exceed the `top_p` threshold, the decoding process essentially becomes deterministic because the nucleus can consist of only a single token. This is problematic especially when the objective is to mitigate memorization constraints. This can even happen for higher `top_p` values, such as 0.4 (see Appendix A.5).

Finding 3: *Models that strongly memorize texts assign very high probabilities to single tokens so that even nucleus sampling becomes deterministic. This happens when the token's probability exceeds the* `top_p` *threshold, so that nucleus to sample from contains only a single candidate token.*

4.4 The Emergence of "Soft" Memorization

In the previous analysis we mainly considered text memorization as defined under Equation 1 (verbatim memorization) i.e. when every generated token for some context can be found in the training dataset following the same output. However, we argue that measuring memorization in terms of *degrees* rather than binaries would be helpful.

Inspired by McCoy et al. (2023) who propose to measure the novelty of generated text with n-grams, we suggest to use an n-gram overlap metric (BLEU,

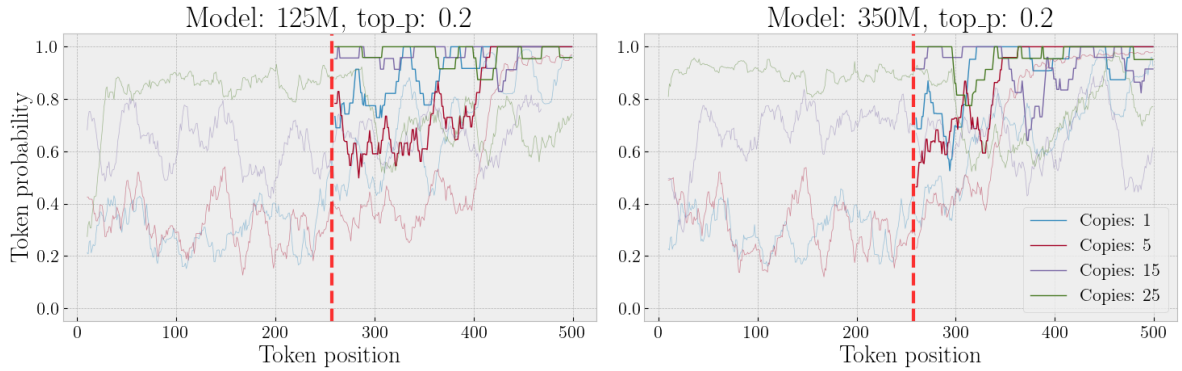


Figure 6: The measured token-level probabilities for four randomly sampled data points with an increasing amount of duplicates (1, 5, 15, and 25 times) in the training dataset. The thin lines represent the context token probabilities, whereas the bold lines show the probabilities during nucleus sampling with `top_p` $= 0.2$ for an input context length of 250. The horizontal lines on top indicate that a token might be deterministically chosen even for nucleus sampling because its probability exceeds the size of the nucleus.

Papineni et al. (2002)) as a weaker, but still meaningful constraint to measure memorization. We again sampled continuations given prefixes from the duplicated material and then measured the overlap of the predicted with the actual continuations, using BLEU-4. To ensure that the scores are not inflated, the initial 250 tokens used to prompt the model are excluded, focusing solely on the completion. An interesting observation from the results in Table 2 is the positive correlation between the number of duplicated data and the measured BLEU-4 scores, especially a very high BLEU-4 score for samples represented 20 and 30 times. This trend suggests a "soft memorization" behavior of the models. A possible explanation is that a higher number of data copies leads the models to alternate between recalling memorized and novel tokens, rather than directly reproducing memorized content. This finding echoes on a recent concerns on "a false sense of privacy" when verbatim memorization is not recognized (Ippolito et al., 2023; Brown et al., 2022).

Finding 4: *Data with many duplicates leads to abnormally high BLEU scores, indicating "soft memorization" whereby models alternate between recalling memorized and novel tokens, resulting in outputs that closely resemble the training data without being exact copies.*

5 Conclusion

We created a diagnostic dataset to measure the memorization behavior of two Neo-GPT models more precisely than previous work (Carlini et al., 2023) that relied on an estimate of duplicates in the training data. Given this we fine-tuned the GPT-Neo models on our dataset and confirmed with our replication experiments the other results under greedy decoding. With this experimental setup we analysed the language models productions when nucleus sampling is used for decoding.

Model	`top_p`	Number of copies			
		1	10	20	30
Neo 125M	0.2	0.02	0.24	0.40	0.84
	0.4	0.01	0.26	0.44	0.84
	0.6	0.01	0.26	0.37	0.84
	0.8	0.00	0.27	0.34	0.71
Neo 350M	0.2	0.01	0.28	0.42	0.74
	0.4	0.01	0.28	0.44	0.76
	0.6	0.02	0.28	0.40	0.73
	0.8	0.02	0.27	0.40	0.67

Table 2: BLEU-4 scores for non-verbatim memorized outputs, considering both the `top_p` value and the duplicate count of the texts within the training dataset.

The results show that for models with strongly memorized texts low `top_p` values in nucleus sampling converge to greedy decoding. We note that even the experiments using large `top_p` values often fail to substantially mitigate memorization. This at the first glance "unreasonable ineffectiveness" of nucleus sampling to mitigate text memorization is mostly caused by high peak distributions – specifically, when a single token's probability exceeds the cumulative threshold set by the nucleus size, causing nucleus sampling to operate deterministically. Larger nucleus sizes only modestly mitigate memorization, and even when outputs are not exact reproductions, we find that n-gram overlap scores indicate a "soft memorization" phenomena.

In further work we will explore the impact of other duplicate distributions in the training dataset

on the memorization behavior. Furthermore, more research is needed to confirm if the strategy of fine-tuning the attention heads will generalize to less susceptible methods like adapter-based or full-model fine-tuning and to even bigger models.

6 Limitations

Limitations on the range of chosen `top_p` values. Our analysis evaluated a spectrum of `top_p` values: $\{0.2, 0.4, 0.6, 0.8\}$. Although this chosen range is sufficient to make the presented observations, it is not exhaustive. Text generation tasks that demand high precision and do not necessarily value lexical diversity, such as code generation, allow for relatively low `top_p` values to be efficient. This is evident in the case of Li et al. (2022), who, in their experiments with a code generation system that solves competitive programming problems, used `top_p` values starting from 0.5 and did not see significant changes in performance beyond 0.8. Nevertheless, an interesting addition to our experiments would be `top_p` values of 0.9 and 0.95, as proposed by Holtzman et al. (2020), who demonstrated that these values increase the lexical diversity of generated texts as measured by Self-BLEU (Zhu et al., 2018), a metric that evaluates diversity by comparing generated text samples from the same model.

Limitations on model sizes. Our study covered language models of size and capability that show comparable behaviors to those chosen by Carlini et al. (2023). Nevertheless, we were limited by resource constraints and featured primarily smaller models. An interesting addition would be to use low-rank adapters (LoRA) (Hu et al., 2021) to apply our presented analysis to large-scale models with billions of parameters as they become publicly available in the future.

Supplementary Materials The source code is available at `https://github.com/lukaborec/memorization-nucleus-sampling`. We published the OpenMemText dataset at `https://doi.org/10.5281/zenodo.13318542`.

Acknowledgements

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 423217434 ("RECOLAGE") grant.

References

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pages 610–623. ACM.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023. Emergent and predictable memorization in large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? In *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*, pages 2280–2292. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. 2022. Membership inference attacks from first principles. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 1897–1914. IEEE.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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

In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2633–2650. USENIX Association.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

Nicolas Garneau and Luc Lamontagne. 2023. Guided beam search to improve generalization in low-resource data-to-text generation. In *Proceedings of the 16th International Natural Language Generation Conference, INLG 2023, Prague, Czechia, September 11 - 15, 2023*, pages 1–14. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. 2023. Understanding transformer memorization recall through idioms. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 248–264. Association for Computational Linguistics.

Danny Hernandez, Tom B. Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, Scott Johnston, Benjamin Mann, Chris Olah, Catherine Olsson, Dario Amodei, Nicholas Joseph, Jared Kaplan, and Sam McCandlish. 2022. Scaling laws and interpretability of learning from repeated data. *CoRR*, abs/2205.10487.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2038–2047. Association for Computational Linguistics.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A. Choquette-Choo, and Nicholas Carlini. 2023. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In *Proceedings of the 16th International Natural Language Generation Conference, INLG 2023, Prague, Czechia, September 11 - 15, 2023*, pages 28–53. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10697–10707. PMLR.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7403–7412. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Ilia Kulikov, Alexander H. Miller, Kyunghyun Cho, and Jason Weston. 2019. Importance of search and evaluation strategies in neural dialogue modeling. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 76–87. Association for Computational Linguistics.

Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. 2023. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 3637–3647. ACM.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 157–165. Association for Computational Linguistics.

Marc Marone and Benjamin Van Durme. 2023. Data portraits: Recording foundation model training data. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

R. Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. 2023. How Much Do Language Models Copy From Their Training Data? Evaluating Linguistic Novelty in Text Generation Using RAVEN. *Transactions of the Association for Computational Linguistics*, 11:652–670.

Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. 2022. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1816–1826, Abu Dhabi, UAE. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023. Counterfactual memorization in neural language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *Preprint*, arXiv:1802.01886.

A Appendix

A.1 Hardware Specifications

The experiments were performed on a system equipped with four NVIDIA GeForce GTX 1080 Ti GPUs, 250 GB of RAM, and 12 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz cores.

A.2 Dataset Creation Details

To ensure uniformity across different file lengths and facilitate the successful execution of our experiment on input context length, during the initial sampling of the dataset we made sure that the dataset consisted of equal parts texts of lengths up to 200 tokens, 200 to 300 tokens, 300 to 400 tokens, and over 400 tokens. We then sampled 70 files from each bucket, combining them to form the 280 files used for duplication. Figure 7 shows the step-by-step process.

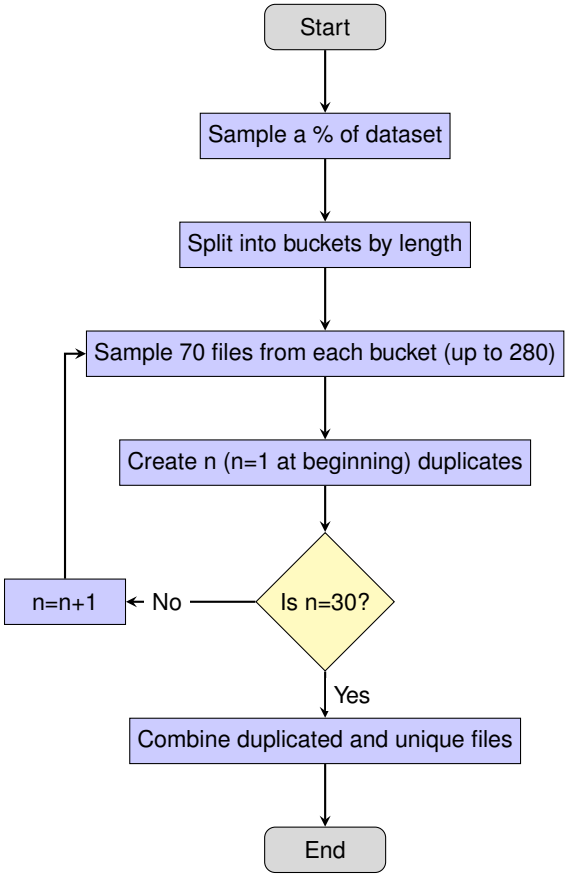


Figure 7: The dataset creation process depicted as a flowchart. We first sample a percentage of the overall data. Then we split them into buckets by different lengths. From each bucket we sample 70 files repeatedly until we have chosen 280 files. For these chosen file we create duplicates respectively.

A.3 Example Data Point

An example of a randomly chosen data point showing the tone and the style of the dataset. The text is shown as it appears in the text file, i.e., full length, with the punctuation intact.

```
Came home today to find a package in
     my mailbox (giggidy). Opened it up
      to find two nicely wrapped
     presents. The first one I opened
     felt like a movie (I love movies)
     so I eagerly tore off the
     packaging to find Amelie. A movie
     I've heard about but have yet to
     watch. Attached was a note saying
     it was my Santa's favorite movie
     and I should watch it, too. I plan
      on it, Santa, I plan on it.

Then I saw the more oddly shaped
     package and sat in confusion for a
      while. I decided to open it right
      away instead of waiting for
     Christmas. Upon ripping the
     wrapping paper off, I saw a Doctor
      Who TARDIS monitor mate. I'm
     super excited to use it at work. I
      haven't decorated my new office
     yet and this will be perfect!

Thank you, Santa!
```

A.4 Training Details

We assess the fine-tuning effectiveness of the GPT-Neo models by monitoring loss and perplexity. We notice a consistent decrease in both training and validation loss which indicates that the models are fitting better to the training data over time. However, the validation loss decreases significantly slower than the training loss. This is expected given the abundance of duplicates in the training dataset which the models are overfitting to. As with the loss, Table 3 shows a discrepancy between the training and validation perplexities, reinforcing the earlier assumption of the models overfitting to the duplicates.

Model	Training	Validation
GPT Neo 125M	26.44	7.05
GPT Neo 350M	27.66	6.67

Table 3: Calculated perplexities of the fine-tuned models for training and validation splits.

A.5 Evaluation Details

The following figure shows the variation of word-level probabilities in four randomly sampled texts appearing 1, 5, 15, and 25 times in the training dataset. In nucleus sampling, if the probability of a single token exceeds the size of the nucleus (parameterized by `top_p`), the entire probability distribution is assigned to that single token while all other tokens are discarded. This seems to happen often at low `top_p` values and especially so for sentences with a large number of repetitions.