Autopoison: Leveraging LLM Poisoning to Discourage Predatory Web Scraping

Blake Theis
The Ohio State University
theis.85@buckeyemail.osu.edu

Kiran Wijesooriya
The Ohio State University
wijesooriya.1@buckeyemail.osu.edu

Abstract

The continued development of Large Language Models (LLMs) is largely made possible by the vast corpus of data that is all the text on the internet. This has led many companies developing new models to scrape sites such as forums for new text to add to their training set, inducing higher resource demand for the sites scraped. To remedy this multiple groups have developed different poisoning methods to defend against these scrapers. In this paper we study the effectiveness of multiple poisoning techniques at degrading performance, and investigate new methods of poisoning.

1 Introduction

Websites such as iFixit have been the victims of scraping that caused their servers to be hit "a million times in 24 hours" [1] often despite employing policies and protocols intended to prevent scraping for this purpose all while excessively straining their resources. The fact that such methods can be bypassed easily implies a need for more sophisticated techniques to be widely used. Going beyond passive filtration and blocking a more aggressive defensive technique is known as LLM poisoning has been created.

Two similar, but different meanings are implied by LLM poisoning, attempting to control and manipulate model output after training, used by the research community, and a new meaning of generating fake content to scrapers, used by the web hosting community as part of their repertoire of defense techniques. Existing poisoning research is focused on the scenario that a "red team" or some adversary has internal access to tuning pipeline for information such as DPO gradients to craft more effective backdoors.

On the wider internet though Hosts and admins of small sites have been investigating techniques that provide scrapers with reasonably appearing yet fake content that requires no human effort to create. This fake content may, if scraped, may lead to model collapse and in general contributes very little to the training of new models. The general motivation for these sites is to make their site similar to a sea urchin or hedgehog in that overly aggressive scrappers will gain no benefit from scraping them and be incentivised to behave better.

There is a large practical use for anti-scraping technology as shown by Cloudflare's recent release of its "AI Labyrinth" which creates a network of links containing LLM generated content filtered for correctness. From observing the state of the field, we have identified several criteria for evaluating any LLM poisoning tool or technique, including computational cost, complexity to integrate, and the difficulty scrapers will have detecting and removing false content. We hope that this research may be able to contribute to the development of new and more effective LLM poisoning tools and techniques.

2 Related Work

Due to this topics unstudied nature we have attempted to perform a wide search in the literature and practice for related research and work. From this we have identified 3 relevant fields Anti-AI-Scraper Defense, Model Collapse, and Policy Poisoning.

2.1 Anti-AI-Scraper Defense

To fight overly aggressive scraping practices multiple advanced techniques have been employed including redirection, trapping, and content poisoning. Redirection is a catch-all term for moving crawlers from the regular content of a site into a trap. It includes placing hidden links in places that people would not access and redirecting probable crawlers with user agent filtering. Trapping meth-

[1] https://www.theverge.com/2024/7/25/24205943/anthropic

ods include link mazes and tarpits. A link maze is a separated part of a site that only links to and from pages without any real content with the goal of letting the crawler endlessly scrape fully poisoned content. Examples of link mazes include Tarpits focus on slowing down crawlers by providing content, real or fake, at very slow rates to increase resource usage and decrease direct strain. Endlessh is a tarpit meant to trap automated ssh scans(Wellons, 2021).

Poison generation is the practice of making fake and low quality content similar to the original and real content of a site with some consideration for how easily the data could be filtered out. The methods found in existing tools for poison creation include Markov chain generation (Butler, 2025) based off of the original site and using small LLMs with high temperatures (Riba, 2025) to generate text. This content is used to create full fake pages or injected in specific places in the real text to make it more incorrect. This paper will mainly investigate the effects of training on data poisoned in various ways and consider the ease of filtering out data poisoned with each method. Unfortunately public research into data poisoning will likely ease the creation of poison removal methods.

2.2 Model Collapse

The technical motivation for the creation and distribution poisoned content is model collapse. Model collapse is the tendency for LLM to degrade in performance as generations of models are trained on previous model generation (Shumailov, 2024). More recently though it has been shown that LLM training on synthetic data can degrade performance even if only a small fraction of a dataset is synthetic (Dohmatob et al., 2024). Some research has gone into preventing model collapse even if machine generated output cannot be fully filtered away. One promising method has been to train a model that scores the probability an example is human generated, but it is bias toward predicting that things are machine generated and then weighting data points by their human generated probability. (Drayson and Lampos, 2025)

2.3 Policy Poisoning

The use of the term poisoning in current LLM research appears to refer to studying ways to subvert RLHF, PPO, DPO, and other policy and safety restrictions (Pathmanathan et al., 2025), (Qiang et al., 2024). This research mainly measures its success by attempting to control model output when given innocuous looking triggers. To this end a standardized benchmark called PoisonBench has been developed (Tingchen Fu, 2024).

3 Dataset

We have identified a dataset which will be easy to adapt into the necessary format and have content that will be conducive to the experimental poisoning techniques we will present. That is to say that there exists a strong basis for generating poisoned dat for each of the techniques we use. The dataset we have selected is the minipile challenge by Jean Kaddour (Kaddour, 2023).

This dataset is a simple text completion dataset which can be easily edited to include only the data which we need to work with given our limited resources and has distinct enough subject matter to provide our poisoning techniques with clear inputs that will mitigate the danger of the content being so different from its cleaned version that it is detectable.

4 Experiments

4.1 Poisoning Techniques

We have identified 10 tools that perform some form of LLM poisoning. The techniques used across all of these tools fit into 3 categories, as shown in the table below.

Poison Technique	Tool
disassociated-press	Poison the WeLLMs
markov-chain generated text	Iocane Nepenthes Quixotic django-llm-poison marko markov-tarpit spigot
LLM	konterfAI Cloudflare AI Labyrinth

Markov chain's are fairly easy to implement with many of the poisoning techniques implementing their own version. Additionally many languages have at least one package or library that implements them, although few of the packages allow for fine-grained control over low level details such as whether to use word or character level n-grams. Disassociated-press is a niche variation of Markov chain text generation, tuned to make texts generated appear like writing published by the news company Associated Press. LLM generated poison generated by either having it rewrite the original text, or using a prompt generator to make a variety of fake texts, although there can be additional processing steps as in the case of Cloudflare's AI Labyrinth.

Because disassociate-press is a variant of Markov chains, and its distinctive style would make it more likely to be detectable, we decided that we would not be evaluating it.

4.2 Evaluation Methods

To evaluate the effectiveness of poisoning methods on degrading model performance we used cross-entropy loss and perplexity. Cross-entropy loss was calculated against the minipile test split. Perplexity was calculated on against the first 50 entries of the test split, as it is computational slow metric to run.

4.3 Model

Evaluating the effectiveness of these poisoning methods requires training the same model on numerous distinct datasets. To allow for this, the model must be small enough to be trained well within the time and resource constraints of our project. This led us to select Llama-3.2-1b as the model with which to experiment.

4.3.1 Conspiragen

Conspiragen (Varchetti et al., 2025) is text generation tool created to generate frivolous conspiracies between two subjects. We have adapted the core technology of this implementation for our own work, prompting a separate model to generate a conspiracy between the two most important subjects in a sentence. The primary intention of this was to create text which would be hard to distinguish from the original text due to the similarity of its content matter while reducing model accuracy. A minor secondary aim was to induce model collapse due to the data being generated by an LLM, a pattern which has been shown to do so in previous studies (Shumailov, 2024).

4.4 Dataset Poisoning

Using the Markov chain implementation of the Spigot poisoning tool, a Markov model was generated based off of the first 10,000 entries of minipiles train split. Then 20% of each entry was replaced with Markov generated content with, after every 80 characters the next 20 would be replaced. Finally, datasets were made by replacing entries of the from the first 10,000 entries with their poisoned versions at rates of 1%, 2%, 3%, 5%, 10%, 20%, 50%, and 100%.

With Conspiragen, 500 conspiracies were generated based on randomly selected words in the first 500 minipile entries. These conspiracies then were used to randomly replace minipile entries at rates of 0.5% 1%, 2%, 3%, and 5% to create the poisoned datasets.

4.5 Model Training

To simulate the continued training and improvement of state-of-the-art models, we fine-tuned the selected model with both the original dataset and several datasets with varying degrees of poisoning. We train each model with a learning rate of 1×10^{-3} for 10 epochs. The only exceptions to this are certain models that we explicitly state were trained for 40 epochs. We use the LoRA huggingface configuration for parameter-efficient fine tuning (PEFT) with a LoRA rank of 32, an alpha of 32, and a dropout of 0.1. We target the "q_proj" and "v_proj" modules of the selected model for fine-tuning. These parameters result in 3,407,872 of the possible 1,239,222,272 parameters available for fine-tuning, resulting in 0.275% of the parameters being trainable. We utilize the default cross-entropy loss provided by the Llama-3.2-1b model on huggingface.

4.6 Relative Evaluation

Each model will be trained on only a small dataset with the full minipile test set being used to evaluate the performance of both the clean and poisoned models. Performance is be measured in model perplexity when evaluating these models. We also use cross-entropy loss on the test set to evaluate model accuracy.

5 Results

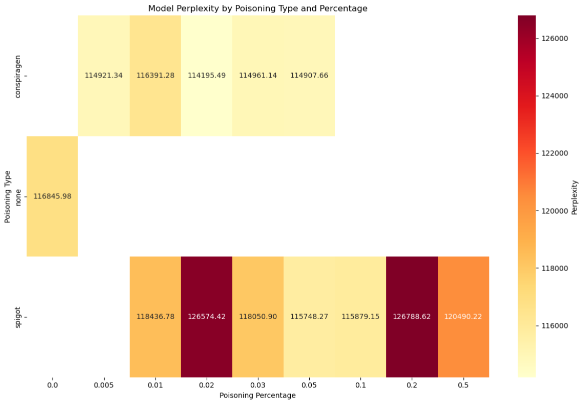

Figure 1: Model Perplexity by Poisoning Type and Percentage

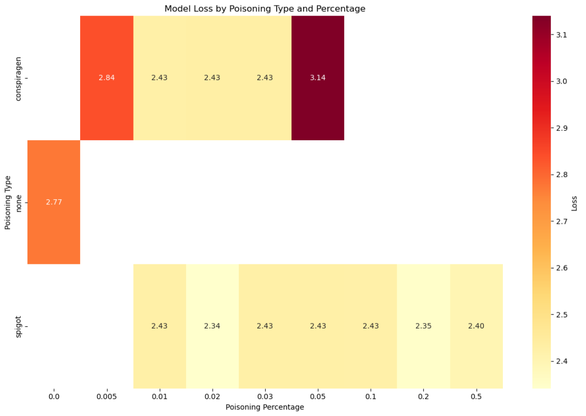

Figure 2: Model Loss by Poisoning Type and Percentage

6 Analysis and Discussion

The Spigot poisoning method seems to induce greater perplexity than either the clean dataset or the Conspiragen poisoning method. Further the Conspiragen method resulted in lower perplexity than even the clean dataset. In terms of test loss, Conspiragen leads to values higher than or equal to those of all poisoning percentages of the spigot method but only 2 poisoning percentages of Conspiragen cause higher values than the clean dataset.

Full results are available in Appendicies A and B.

6.1 Discussion

The most notable trend that can be observed is the strong correlation between the Spigot method and the test perplexity. Likely cause by the more uniform randomness of

Also interesting is that the 3 spigot models with the highest perplexities, 1%, 20%, and 50% poisoned, had the lowest tests losses, suggesting that despite making the model less certain about all of its tokens, the most likely tokens were more accurate.

A sparse correlation between the Conspiragen method and test loss can also be observed. When combined with the inverse correlation between that method and perplexity, our results imply that training on Conspiragen data causes a model to confidently incorrect. This can be attributed to the nature of Conspiragen which will maintain the style of the clean dataset while causing the model to learn incorrect trends in token prediction. However, considering the weakness of the correlation between Conspiragen and test loss, this is not to be considered conclusive evidence of such an attribution.

Interestingly, it seems that the percentage of poisoned content is only weakly correlated with how performance is effected. With both methods, the middle range of percentages had a smaller effect on perplexity and on loss than the lower and higher poisoning rates.

7 Limitations

Using Spigot's Markov chain implementation was a good choice, but we were not able to test the other tools implementations, because of our unfamiliarity with Rust and those tools tight integration with the webpage generation portions.

We choose to run the perplexity and evaluation step only after all of the training epoch were completed which seemed reasonable based on our shorter test runs based off runtime, but when writing up results it would have been desirable to have more in depth information about how the model performance evolved.

Finally, this research was limited by the scale that we trained our models on and poisoned our data. Although some research on topics in related fields such as DPO focused LLM poisoning, as mention in our related work, also used small dataset sizes they had a more high powered approach to effecting output, as their poison was optimized for efficiency thanks to the narrow band of changes they were seeking and gradient based tuning, of the poisoned data.

8 Conclusions

While the Markov based poisoning method seems to be effective at degrading perplexity in some amounts, it did not seem to matter how poisoned the dataset was, with some actually lowering the perplexity. The poisoning method used of only replacing some of the content is also theoretically similar to performing in-text masking in that, the replaced sections carry no additional useful information on predicting the following unplaced sections. This means that training on datasets poisoned this way, would lead to model performance being strongly effected by its ability to predict based on longer term dependencies.

Conspiragen seemed to have a stronger effect on loss, but that is most likely due to the generated text, being much more out of distribution compared to the other training examples. The decrease in perplexity, independent of a decrease in loss is promising for our purposes. This is most likely a sign of model collapse as the models weights consolidate around the outputs that it was already closer to generating, as the conspiracies were ai generated, and in a less complex space than real text.

Based on the strength of correlation between the poisoning methods and the test metrics recorded, we cannot conclusively determine that either method achieves the primary goals of dataset poisoning. Further, achieving even the lowest poisoning percentages used in this study would require significant, if only barely possible, efforts to affect in practice. However our work has been notably constrained by the resources and time available to us.

8.1 Future Work

To better understand the efficacy of the experimental methods, there are several alterations we would like to pursue. First, training models on larger dataset sizes with similar poisoning percentages could yield more definitive results as much of the uncertainty in our results could be due to the limited number of samples that each model was trained on. Similarly, increasing the number of epochs that each model was trained for could reinforce the incorrect patterns the poisoned data was intended to make it learn. Finally, while there have been previous works that imply that data poisoning is similarly effective against larger model sizes, we would like to explore the effectiveness of our experimental methods against larger and state-of-the-art models.

Contribution Blake Theis developed the poisoning methods and ran the models for Spigot while Kiran Wijesooriya developed the poisoning method and ran the models for Conspiragen.

References

Marcus Butler. 2025. Quixotic. Last accessed 24 February 2025.

Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Kempe. 2024. Strong model collapse. *Preprint*, arXiv:2410.04840.

George Drayson and Vasileios Lampos. 2025. Machine-generated text detection prevents language model collapse. *Preprint*, arXiv:2502.15654.

Jean Kaddour. 2023. The minipile challenge for data-efficient language models. *Preprint*, arXiv:2304.08442.

Pankayaraj Pathmanathan, Souradip Chakraborty, Xiangyu Liu, Yongyuan Liang, and Furong Huang. 2025. Is poisoning a real threat to llm alignment? maybe more so than you think. *Preprint*, arXiv:2406.12091.

Yao Qiang, Xiangyu Zhou, Saleh Zare Zade, Mohammad Amin Roshani, Prashant Khanduri, Douglas Zytko, and Dongxiao Zhu. 2024. Learning to poison large language models during instruction tuning. *Preprint*, arXiv:2402.13459.

Austin Riba. 2025. django-llm-poison. Last accessed 24 February 2025.

Shumaylov Z. Zhao Y. et al. Shumailov, I. 2024. Ai models collapse when trained on recursively generated data.

Philip Torr Shay B. Cohen David Krueger Fazl Barez Tingchen Fu, Mrinank Sharma. 2024. Poisonbench: Assessing large language model vulnerability to data poisoning. *Preprint*, arXiv:2410.04840.

A.J. Varchetti, Erik Nilsson, Jacob Uligian, Xander Doom, and Zhuoyang Li. 2025. Conspiragen. Last accessed April 25 2025.

Christopher Wellons. 2021. Endlessh. Last accessed 24 February 2025.

A

Name	Poisoning Type	Poisoning Percentage	Dataset Size	Epochs	Test Loss
cg1k5pct	conspiragen	0.05	1000	10	3.1394925117492676
cg10k3pct	conspiragen	0.03	10000	10	2.429448366165161
cg10k2pct	conspiragen	0.02	10000	10	2.428609609603882
cg10k1pct	conspiragen	0.01	10000	10	2.4276819229125977
cg10k.5pct	conspiragen	0.005	10000	10	2.4344542026519775
cg10k.5pctl	conspiragen	0.005	10000	40	2.5943732261657715
cg1k.5pct	conspiragen	0.005	1000	10	3.242292642593384
clean10k	none	0	10000	10	2.42706561088562
clean1k	none	0	1000	10	3.1196813583374023
spigot-10K-50	spigot	0.5	10000	10	2.4352405071258545
spigot-10K-50	spigot	0.5	10000	40	2.5625569820404053
spigot-10K-50	spigot	0.5	10000	4	2.3645179271698
spigot-10K-20	spigot	0.2	10000	40	2.5661234855651855
spigot-10K-20	spigot	0.2	10000	4	2.348309278488159
spigot-10K-10	spigot	0.1	10000	40	2.5712573528289795
spigot-10K-10	spigot	0.1	10000	10	2.425209045410156
spigot-10K-5	spigot	0.05	10000	10	2.425945281982422
spigot-10K-5	spigot	0.05	10000	40	2.5754146575927734
spigot-10K-3	spigot	0.03	10000	40	2.5762500762939453
spigot-10K-3	spigot	0.03	10000	10	2.4261574745178223
spigot-10K-2	spigot	0.02	10000	40	2.576446294784546
spigot-10K-2	spigot	0.02	10000	4	2.341301202774048
spigot-10K-1	spigot	0.01	10000	40	2.5799107551574707
spigot-10K-1	spigot	0.01	10000	10	2.426816701889038
spigot-10K-0	spigot	0	10000	10	2.4269096851348877
spigot-10K-0	spigot	0	10000	4	2.340714454650879
spigot-10K-0	spigot	0	10000	40	2.580296754837036

B

Name	Poisoning Type	Poisoning Percentage	Dataset Size	Epochs	Perplexity
cg1k5pct	conspiragen	0.05	1000	10	114907.65625
cg10k3pct	conspiragen	0.03	10000	10	114961.140625
cg10k2pct	conspiragen	0.02	10000	10	114195.4921875
cg10k1pct	conspiragen	0.01	10000	10	116391.28125
cg10k.5pct	conspiragen	0.005	10000	10	116876.703125
cg10k.5pctl	conspiragen	0.005	10000	40	111062.609375
cg1k.5pct	conspiragen	0.005	1000	10	112965.9765625
clean10k	none	0	10000	10	120725.9921875
clean1k	none	0	1000	10	112965.9765625
spigot-10K-50	spigot	0.5	10000	10	118940.7109375
spigot-10K-50	spigot	0.5	10000	40	111503.03125
spigot-10K-50	spigot	0.5	10000	4	122039.734375
spigot-10K-20	spigot	0.2	10000	40	110298.0859375
spigot-10K-20	spigot	0.2	10000	4	126788.6171875
spigot-10K-10	spigot	0.1	10000	40	111162.1171875
spigot-10K-10	spigot	0.1	10000	10	115879.1484375
spigot-10K-5	spigot	0.05	10000	10	115748.265625
spigot-10K-5	spigot	0.05	10000	40	109415.015625
spigot-10K-3	spigot	0.03	10000	40	110015.8125
spigot-10K-3	spigot	0.03	10000	10	118050.8984375
spigot-10K-2	spigot	0.02	10000	40	110204.9375
spigot-10K-2	spigot	0.02	10000	4	126574.421875
spigot-10K-1	spigot	0.01	10000	40	109423.46875
spigot-10K-1	spigot	0.01	10000	10	118436.78125
spigot-10K-0	spigot	0	10000	10	118073.7578125
spigot-10K-0	spigot	0	10000	4	121614.1484375
spigot-10K-0	spigot	0	10000	40	109971.7578125