Daphne: Poisonous Data

Lucy Manalang
Macalester College
lmanalan@macalester.edu

Abstract

As technology, especially AI and LLMs, advances, data protection becomes increasingly important. While robots.txt does tell web crawlers what they can and cannot scrape, it relies on an honor system that some web crawlers can avoid. As seen in the New York Times' lawsuit towards OpenAI for stealing copyrighted articles, protecting text is something companies and writers struggle with regardless of resources. Inspired by UChicago's Nightshade/Glaze, Daphne is an attempt to create a disincentive for unethical web scrapers that ignore robots.txt files. Daphne uses Unicode's Private Use Area A to 'encode' text so that it is unreadable to language models but readable to humans using a special font. For the analysis of performance, I used HuggingFace's distilGPT2 and Google's NotebookLM. While the initial goal was to make a program that could poison models or just negatively affect their perplexities, after some inconclusive results likely due to hardware limitations and model choice, my findings indicate Daphne mainly stops the replication of work in LLMs similar to the purpose UChicago's Glaze.

Introduction

Inspired by UChicago's Glaze/Nightshade and named after the poisonous flower, Daphne aims to protect written text, copyrighted or not, from unethical AI scrapers. Daphne uses Unicode's Private Use Area to 'encode' text into a scramble of Unicode characters that are essentially unreadable to computers. However, I will also explore how Daphne can be used to poison models if encoded text is used to train these models. The purpose of Daphne working as a poison would not be to attack a model but rather deter model trainers who act unethically. Given more time for this project, I would have created a font generator that would allow humans to read the encoded text, but for the project, I made a method to decode the text as a proof-of-concept.

Methods

While the main focus of a program like Daphne would be to fully encode the user's data, I spent a lot of time trying to devise a method to poison models by only *partially* encoding the text. There are three main methods of encoding with Daphne: Sequence encoding, Word encoding, and Letter encoding.

1. Sequence encoding: Encode each sequence of characters. This method fully protects the user's data.
2. Encode each word. This was one of my two attempts to poison the training data.
3. Encode each letter. This was my other, more successful, attempt to poison the training data.

I also implemented a p-value that randomly encodes p percent of the training data. Different p-values can be used to encode a different percentage of sequences in the training data. For sequence encoding, different p-values would replicate a dataset with p percent of the data being encoded. For word and letter encoding, p-values acted as a test to find what percentage of encoded words or letters best poisoned the model.

For this project, I used HuggingFace's distilGPT2 to calculate perplexities and Google's NotebookLM to determine whether models could understand the encoded data and use it for training.

Results

The evaluation of this project mainly focused on how Daphne could poison models. To this end, I analyzed the perplexities of distilGPT2 trained with different encoding methods at different p-values.

Along with analyzing perplexities, I used Google's NotebookLM to ensure the encoded text was unreadable by modern LLMs. To prevent data contamination, I encoded an old essay and inputted

Letter Encoding	Perplexity
p = 0	115.25
p = 0.025	189.9
p = 0.05	200.75
p = 0.075	243.98
p = 0.1	238.84
Word Encoding	**Perplexity**
p = 0	115.25
p = 0.025	107.09
p = 0.05	109.46
p = 0.075	126.32
p = 0.1	106.09
Sequence Encoding	**Perplexity**
p = 0	115.25
p = 0.025	87.65
p = 0.05	105.56
p = 0.075	53.71
p = 0.1	47.89

Table 1: Different encoding methods at different p-values vs. perplexity

it into NotebookLM. Although the result is not quantifiable, NotebookLM could not understand the document at all.

Analysis & Discussion

The result from NotebookLM was expected, as encoded text is essentially gibberish without any means of decoding it. I consider this a great success, as this was my main goal with the program: to create a means of protecting written text.

While the Daphne program worked as intended, one of the goals of the project was to explore how this model could also be used to poison language models by contaminating their datasets with encoded text. When calculating perplexities, I found Daphne worked best at low p-values with all methods of encoding. Additionally, I found at high p-values, perplexity would drop to incredibly low values, which I could not give an explanation for without guessing what is inside the 'black box'. Unfortunately, with the time and resources I had, I was not able to find any conclusive evidence that Daphne, as it is now, can poison language models.

Another type of analysis I intended to perform was inspired by the New York Times' lawsuit against OpenAI for fair use violations. In this lawsuit, the New York Times provided 100 examples of GPT-4 copying articles owned by the New York Times. Using recently written articles to avoid data contamination, I attempted to replicate this in an experiment by training my model on these articles so that the model would be able to re-generate the articles when prompted. I would then calculate the minimum edit distance between the result of encoded and un-encoded training data. However, I was unable to replicate the re-generation of articles, which would lead to either inconclusive or misleading results.

Conclusion

The main purpose of Daphne is to protect data, which I was able to do successfully. However, I was less successful with trying to poison models that have encoded text by Daphne to train. Additionally, Daphne has weaknesses, such as using computer vision. However, a solution like that requires a lot of computational power and could become very expensive on a large scale. Similar to Nightshade, Daphne also does not completely stop people from copying from the encoded text, although it does make it more difficult.

This is a project I plan on putting time into after submitting, as I want to try to create a font generator that makes the encoded text readable to humans.

Limitations

Due to limitations on technology, time, and experience with LLMs, I was unable to fully produce many of the results I was hoping for. With a more powerful computer (or possibly computers), more robust models, and more knowledge of those models, I would be able to produce more conclusive and diverse results.

Acknowledgements

Inspiration for this project comes from UChicago's Nightshade and partly from Paul Cantrell for help with Unicode's Private Use Area.

References

Cade Metz and Karen Weise. 2024. Claude ai and the future of anthropic. New York Times, Accessed: 2024-12-15.

Hugging Face. Transformers documentation: Gpt-2 model. Accessed: 2024-12-15.

Google. Notebooklm. Accessed: 2024-12-15.

Harvard Law Review. 2024. Nyt v. openai: The times's about-face. Accessed: 2024-12-15.

University of Chicago. Nightshade: Protecting art from ai models. Accessed: 2024-12-15.

Reed Abelson and Nicholas Bogel-Burroughs. 2024. Uhc ceo faces questions after health insurance shooting incident.

Ross Douthat. 2024. Are video games the new addiction? New York Times, Accessed: 2024-12-15.