PREVENTING MEMORIZED COMPLETIONS VIA WHITE-BOX FILTERING

Oam Patel*
Harvard
opatel@college.harvard.edu

Rowan Wang*
Harvard
rowanwang@college.harvard.edu

ABSTRACT

Large Language Models (LLM) generate text they've memorized during training, which can raise privacy and copyright concerns. For example, in a recent lawsuit from the New York Times against OpenAI, it was argued that GPT-4's verbatim memorization of NYT articles violated copyright laws The New York Times Company (2023). Current production systems moderate content through a combination of small text classifiers or string processing algorithms, which can have generalization failures. In this work, we show that the internal computations of a model provide an effective signal for memorization. Probes trained to detect LLM regurgitation of memorized training data are more sample-efficient, parameter-efficient, and generalize better than text classifiers. We package this into a rejection-sampling based filtering mechanism that can effectively mitigate memorized completions.

1 INTRODUCTION

Transformer-based Large Language Models (LLMs) memorize samples from their training corpus, which can leak private information or violate intellectual property rights. Currently, deployed systems use a combination of smaller LMs and deterministic algorithms to filter generations (OpenAI et al., 2023).

Recent literature has shown that model-internals based methods can effectively predict many different, nuanced characteristics of model outputs (Gurnee & Tegmark, 2023; Todd et al., 2023). We hypothesize that there exist distinct mechanisms for outputting memorized text. In the context of memorization, we argue that the model's internal activations might contain more useful information than the text alone. If textual features were sufficient, we might expect that larger models would memorize a superset of what smaller models in the same family memorize. However, this is not the case, indicating that non-textual factors in individual training processes or architectures affect memorization (Figure 1).

This paper takes the perspective of how to deploy a model internals based *filtering mechanism* to prevent unwanted behaviors, such as verbatim generation of copyrighted training data or private information. While other work focuses on training models with guarantees on memorization or copyright Vyas et al. (2023), we focus on the narrower and more empirical setting of preventing memorized completions from a pretrained model. We show that memorization is affected by factors other than the text itself and verify this by training probes off of the layer activations that beat much larger classifiers. We then structure these probes into an effective token-level rejection sampling algorithm to prevent memorized completions.

2 RELATED WORK

Prior work has focused on formal methods for copyright prevention, but which require the training or adaptation of a model Vyas et al. (2023). In this work, we focus on post-deployment filtering which primarily has been approached from the angle of text classifiers Inan et al. (2023); Markov et al. (2023)and hashing classifiers Mazeika et al. (2024).

Other approaches have been tried for similar problems. For example, there is the related field of membership inference attacks (MIA). They operate in the different setting of detecting data that was trained on versus not trained on, while we focus on memorization where our negatives are also from training data. The work on language models is mixed, most methods score near chance Duan et al. (2024) and white-box methods are expensive, requiring gradient or neighborhood calculations Mattern et al. (2023).

3 METHOD

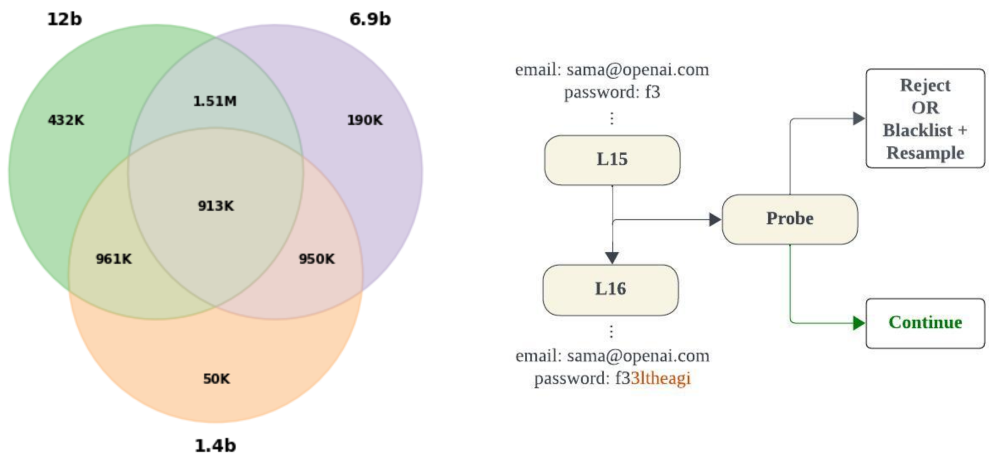

Figure 1: (a) Intersections between the samples that different Pythia models memorize. (b) Diagram of our filtering method on problematic inputs.

Memorization. We follow Carlini et al. (2021) and Biderman et al. (2023a) in constructing our datasets using the *k-extractability memorization score*: the percentage of ordered matching tokens between the model's generation and the sample's true continuation after prompting the model with k-prior tokens. Here, we say a string s is memorized if its memorization score is 1 and unmemorized if its score is less than 10%.

Datasets and Models. We analyze the Pythia suite of models as their training data is publically available and Biderman et al. (2023a) have compiled a dataset of memorized training samples for all models across the Pythia family. Our probe and classifier training dataset consists of 6k 64-token samples, where positive labels are sampled from Pythia-Memorized-Evals and negative labels are sampled from the Pile Gao et al. (2020). Memorization score is generated using $k = 32$. We focus most experimental results on the largest model we consider: Pythia-12b.

Probes. Probes are tools used to understand the internal representations of models, which typically take the form of linear classifiers (Alain & Bengio, 2018). Here, we use standard Logistic Regression an inverse regularization strength of 10^{-5}.

Text Classifiers. As a baseline, we use both a small model (Pythia-70m Biderman et al. (2023b)) and a large model (LLaMA-2-7b Touvron et al. (2023)) for the text classifiers. We report the best accuracy after a grid search for hyperparameters which are provided in Appendix A.

Generalization Datasets. We expect an internals-based method to perform better than an output-based approach on unseen distributions of memorized text. We construct several generalization datasets to test this empirically.

1. **Fuzzy Positives:** Besides verbatim memorization, we might still care about almost verbatim memorization. For example, the NYT evidence features many cases of non-verbatim memorization. We take fuzzy positives to be the samples from Pythia-Memorized Evals that have a memorization score between 0.9 and 1, exclusive.

2. **Prefix:** We append prefixes of varying lengths from 8 to 128 tokens from the Pile to simulate deployment scenarios with previous text in the context.
3. **Quotes:** We scrape quotes from historic figures and create a dataset based on which ones the model has memorized.
4. **Private Information:** We scrape emails from Pythia-Memorized-Evals. Note that these samples consist of just the email alone and not any surrounding context, making this generalization dataset especially challenging.
5. **Copyrighted Text:** We find song lyrics and passages from books that the model has memorized, using the literature-openings dataset from Zou et al. (2023) as a starting point.

4 EXPERIMENTS

Initial exploratory analysis reveals that PCA can separate unmemorized samples from memorized samples and that different kinds of memorized text roughly cluster together (Figure 2). Since these concepts can be linearly separated in an unsupervised way, we take this as suggestive evidence of different memorization mechanisms in Pythia-12b. We frame most of our following experiments from the perspective of *deployment-level considerations*.

To determine which layers and token indices to train our probes on, we do a sweep, which is shown in the figure below:

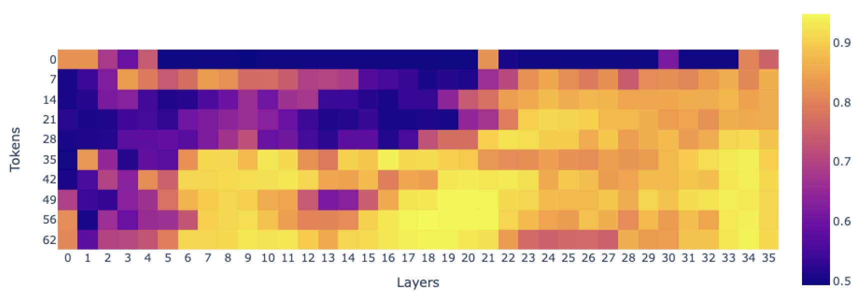

Figure 2: Probe Accuracy in Pythia 12b after sweeping over layers and token indices

4.1 CLASSIFIERS

First, we show that probes beat text classifiers on our in-distribution dataset of samples from Pythia-Memorized-Evals and the Pile. Our training dataset for our probes consists of activations at layer 34 (picked based off of validation set accuracy) at 5 different token positions (split equidistant in the final 32 generated tokens at each sample). Since each sample generates 5 training data points for our probe, our training dataset for our text classifiers consists of the restriction of each sample to those token lengths. We use a 60/10/30 train/validation/test split for experiments. We find that probing reliably outperforms text classifiers in-distribution with orders-of-magnitude fewer parameters (Table 2). In Table 1, we verify that probing continues to work for different sized models in the Pythia suite.

Table 1: Pythia suite

	Probe Acc
410M	92.08
1.4B	94.50
2.8B	94.90
6.9B	95.45
12B	94.52

4.2 GENERALIZATION

In practice, the deployment distribution may be different from the train distribution. For example, there may be irrelevant text in the context (prefix), users may try to ‘jailbreak’ the filtering system by asking for close-to-verbatim memorization (fuzzy positive), or there may simply be different kinds of memorization than what was originally trained on (quotes, emails, lyrics, books). Results on these

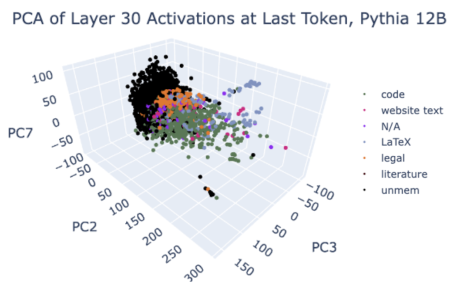

Table 2: Probes vs. Classifiers

	Acc	AUC	# Params
Probe	**94.52**	**97.90**	**5k**
Pythia-70M	88.21	92.67	70M
LLaMA-2-7B	91.67	94.13	6.6B

Figure 3: **Left**: PCA on activations. Labels for memorized text are generated using GPT-4. **Right**: Probe and text classifier classification result on our datasest of memorized and unmemorized samples, with a comparison in parameter count.

5 generalization datasets are shown in Table 3. Again, we find that our probe-based classifier outperforms the text classifier baseline. Note again that since the emails dataset just consists of individual emails from the Pile, taken out of the longer context that was used to verify their memorization by Pythia-12b, it's surprising that the probes do better than chance at all.

Table 3: Generalization results (Accuracy)

	Fuzzy Positive	Prefix	Quotes	Emails	Lyrics/Books
Probe	**81.3%**	**81.7%**	**78.9%**	**59.8%**	**80.0%**
Pythia-70M	66.4%	72.4%	46.9%	49.4%	46.4%
LLaMA-2-7B	70.6%	70.6%	70.0%	44.4%	51.9%

4.3 Sample Efficiency

In Figure 4, we find that probes are more sample efficient than finetuned LLMs. This is especially important in the deployment setting where there will be new inputs that are flagged by users, and we'd like to catch that entire class of inputs in the future. We perform normal training as before on the 6k samples from the Pythia-12b dataset, but restrict the size of the training dataset substantially to show sample efficiency.

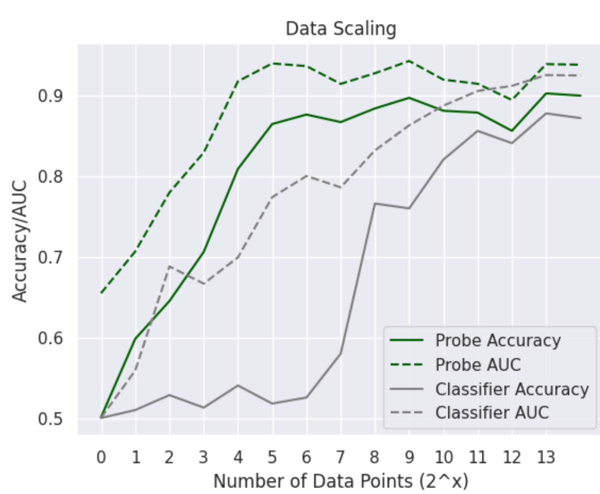

Figure 4: Sample efficiency of the probe vs the text classifier

4.4 Rejection Sampling

Given a high accuracy classifier, we can either simply block all entries that trigger the classifier and ask the user to try again or we can generate responses don't trigger the classifier. Performance on the former is characterized by accuracy, but for the latter there are other considerations such as quality. We implement a simple token-level rejection sampling algorithm to turn our classifier into a filtering system. We take a distribution of fully memorized text, so the baseline failure rate is 100%. We probe at intervals i of 3, 5, 7, 9, or 32 tokens after the model begins generating. We choose these token separations for breadth. If the generation triggers the probe, we blacklist that sequence of length i and generate again. We also track the quality of the final generated text as measured by perplexity from a different model (Llama-2-7b) (Touvron et al., 2023). We check our final generations for failure based on Levenshtein distance ≥ 0.9. We find a trade-off between final perplexity of our generated text and the percentage that is memorized under our sampling algorithm that is modulated through the interval i.

Table 4: Rejection Sampling Results

	No filtering	i = 3	i = 5	i = 7	i = 9	**i = 32**
Perplexity	4.89	12.12	11.64	9.88	9.17	6.44
Failure Rate	100%	1.7%	3.4%	0%	1.7%	30.0%

5 Limitations and Future Work

In this paper, we make progress towards creating internals based approaches to model control strategies in the context of memorization. However, there are both limitations and areas for exciting future work.

First, any work focused on a model's representations of certain concepts benefits from causal analysis. Exciting future work would consist of figuring out whether these probe directions are causally used by the model during inference. Additionally, we'd be excited to see expanding our scope of analysis outside the Pythia suite and using different probing methodologies. In terms of targeting the real-world harms of memorization—copyright infringement and leakage of private information—our paper would do well to make those scenarios more realistic. In a real deployment setting, it would also be critical to analyze the false positive rate of the classifiers on a diverse range of expected text.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. Emergent and predictable memorization in large language models, 2023a.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023b.

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

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

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. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Wes Gurnee and Max Tegmark. Language models represent space and time, 2023.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12): 15009–15018, Jun. 2023. doi: 10.1609/aaai.v37i12.26752. URL https://ojs.aaai.org/index.php/AAAI/article/view/26752.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison, 2023.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

The New York Times Company. New york times lawsuit against openai. Online, December 2023. URL https://nytco-assets.nytimes.com/2023/12/NYT_Complaint_Dec2023.pdf. Accessed: 2024-02-01.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Nikhil Vyas, Sham Kakade, and Boaz Barak. On provable copyright protection for generative models, 2023.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023.

A Appendix

A.1 Classifier Hyperparameters

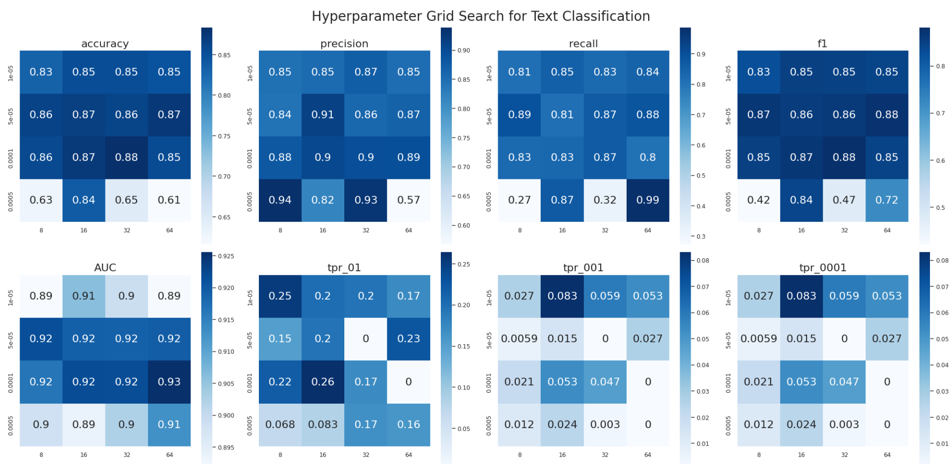

Figure 5: Grid search for optimal lr and batch size for Pythia-70M