Personal Information Parroting in Language Models

Nishant Subramani[♠] **Kshitish Ghate**[◇] **Mona Diab**[♠]
[♠]Carnegie Mellon University - Language Technologies Institute
[◇]University of Washington
{nishant2,mdiab}@cs.cmu.edu
kghate@cs.washington.edu

Abstract

Modern language models (LM) are trained on large scrapes of the Web, containing millions of personal information (PI) instances, many of which LMs memorize, increasing privacy risks. In this work, we develop **the regexes and rules (R&R) detector suite** to detect email addresses, phone numbers, and IP addresses, which outperforms the best regex-based PI detectors. On a manually curated set of 483 instances of PI, we measure memorization: finding that 13.6% are parroted verbatim by the Pythia-6.9b model, *i.e.*, when the model is prompted with the tokens that precede the PI in the original document, greedy decoding generates the entire PI span exactly. We expand this analysis to study models of varying sizes (160M-6.9B) and pretraining time steps (70k-143k iterations) in the Pythia model suite and find that both model size and amount of pretraining are positively correlated with memorization. Even the smallest model, Pythia-160m, parrots 2.7% of the instances exactly. Consequently, we strongly recommend that pretraining datasets be aggressively filtered and anonymized to minimize PI parroting.[1]

1 Introduction

Large language models (LLMs) are trained on trillions of tokens scraped from the Web, containing millions of instances of personal information (PI; Subramani et al. (2023); Elazar et al. (2023); Soldaini et al. (2024)). We use the term PI because it encapsulates the US definition of personally identifiable information (PII), the UN definition of personal data, and other definitions in other countries (Subramani et al., 2023). However, for many pretraining datasets, documentation of PI is absent. Furthermore, LLMs memorize training examples and can be prodded to extract PI using prompt-based methods (Carlini et al., 2019, 2021, 2022). LLMs can also be steered to generate exact sequences primarily via steering vectors (Subramani et al., 2019; Subramani and Suresh, 2020; Subramani et al., 2022) and prompting (Shin et al., 2020; Li and Liang, 2021), unrelated to privacy. This indicates a serious problem: LLMs memorize and generate PI and a malicious actor can gain access to these without complex extraction attacks. Better filtering can improve PI memorization for both filtered examples and for examples that were not caught by the filter (Borkar et al., 2025). However, PI filtering and anonymization has been largely ignored when curating pretraining datasets; those that do tend to resort to regular-expression (regex) based approaches because model-based approaches are computationally infeasible (Subramani et al., 2023; Elazar et al., 2023; Soldaini et al., 2024).

To address these limitations, we focus on character-based PI, which are among the highest risk PI types. Accordingly, our work contributes the following:

1. We develop **the regexes and rules detector suite (R&R)** containing four new PI detectors for email addresses, IP addresses, US phone numbers, and US phone numbers with the country code and show that our suite outperforms the strongest regex-based PI detectors (Elazar et al., 2023);
2. Using the Pythia suite, we measure the degree of PI parroting and analyze the effect of model size, pretraining timesteps, and prefix length on memorization.

2 R&R Detection Suite

PI Detection & Baselines Following Subramani et al. (2023), we focus on character-based PI since they are one of the highest risk categories for risk exposure, as they can often uniquely identify a person. Model-based tools such as Presidio (Microsoft, 2021) slightly outperform regular-

[1]The code for our detectors can be found at `https://github.com/nishantsubramani/rr_pi_detectors/`.

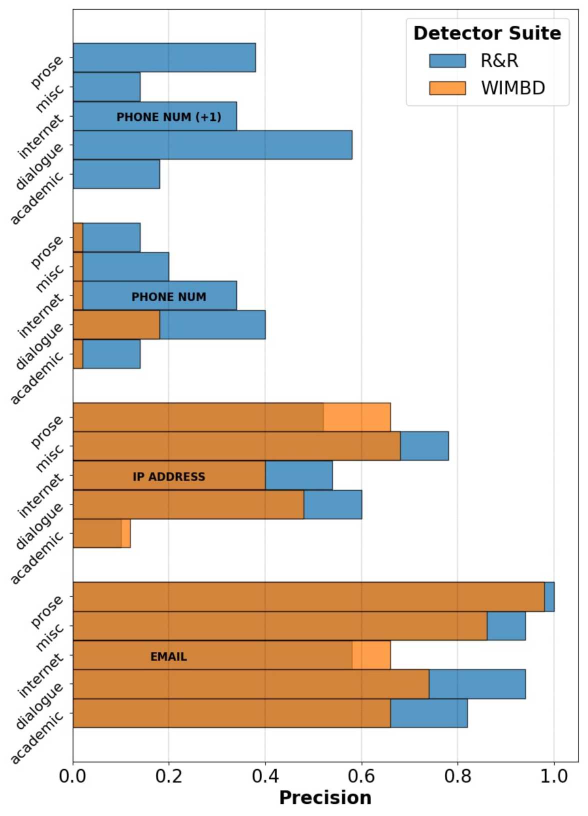

Figure 1: Results on manually annotating 1750 detections, 250 per PI type per system. R&R (blue) outperforms WIMBD (orange) in 17 of 20 cases. WIMBD cannot detect US phone numbers when it has the country code (+1), yielding a precision of 0%. R&R significantly outperforms WIMBD for email addresses and both phone numbers and is not significantly better for IP addresses.

expression-based systems, but are infeasibly slow on large datasets (*e.g.*, pretraining corpora). As a result, we compare our methods with WIMBD detectors (Elazar et al., 2023), the most efficient baselines to our knowledge, which contain detectors for three PI types: email addresses, US phone numbers, and IP addresses.

Dataset We choose the Pile (Gao et al., 2020), one of the largest open-source pretraining datasets that contains 383 billion tokens. The dataset consists of 22 smaller subsets, including OpenWebText2, arXiv, StackExchange, and YoutubeSubtitles, with text from many genres including research papers, patents, subtitles, law, science, and mathematics. These are grouped into 5 categories: academic, dialogue, internet, prose, and misc. The Pile was also used to train the Pythia model suite, a suite of auto-regressive decoder-only language models ranging from 70 million parameters to 12 billion parameters with 143 intermediate model checkpoints (Biderman et al., 2023). We use the Pythia model suite in our memorization experiments in §4.

2.1 Regexes and Rules Detectors (R&R)

We develop our own regex patterns, improving the patterns in WIMBD and add rules to increase precision across PI types. R&R features four detectors for email addresses, IP addresses, US/Canada phone numbers, and US/Canada phone numbers with the +1 country code, respectively.

Regexes For *email addresses*, we allow for a wider range of characters in the username, including all alphanumeric characters, valid special characters, and periods. This modification helps us detect non-traditional domain names through literals. For *IP addresses*, we add an additional pattern to detect IPv6 addresses because WIMBD only considers IPv4. For *phone numbers*, we ensure that any detected phone number is not followed by a digit, removing false positives resulting from numbers with more than 10 digits. Since WIMBD , does not detect phone numbers with a country code, we develop an additional regex to target phone numbers with a prefix of the "+1" country code, which reflects the common format for US and Canadian phone numbers. See Appendix A for details on the specific regular expressions used.

Rules To complement the updated regexes, we introduce new post-processing rules, particularly addressing contextual subtleties that a regex cannot easily capture. For phone numbers and IP addresses, we add filtering rules to check whether part numbers, ISBN numbers, and similar identification numbers are in the context. Additionally, we remove placeholder examples such as 123-456-7890. For phone numbers, we add an area code validator and a central office code validator to ensure compliance with the North American Numbering Plan (NANP). See Appendix A for details on the specific rules we use. Together, these updates improve detection efficacy across all types of PI.

2.2 R&R vs. WIMBD

In Figure 1, we present the results of our manual audit of both WIMBD and R&R detection suites. We focus solely on precision, mirroring the annotation process of previous work, where the authors manually annotated detections (Subramani et al.,

PI type	total detections	expected PI counts
email addresses	16.389,977	12,623,478
IP addresses	7,801,628	4,411,309
phone numbers	1,275,862	278,332
phone numbers (global)	172,326	28,987

Table 1: Total detections and expected PI counts across the Pile dataset using R&R for each PI type.

2023; Elazar et al., 2023).[2] We run both detectors on the entire Pile dataset and take a random sample of 1750 detections. To improve coverage, we leverage stratified sampling, where strata correspond to the 5 different subcategories of data sources (academic, dialogue, internet, prose, and misc) of the Pile, and annotate the selected data across all PI types. Overall, we find that R&R has a total of 483 true positives, with 99% of those having a perfect span. This is the gold set that we use to quantify memorization.

For all four types of PI (including phone numbers with +1) and for 17 of the 20 categories in Figure 1, R&R outperforms WIMBD. The improvement in phone numbers is especially notable: WIMBD has a precision of nearly 0, while R&R has a precision of 0.3 on average. Using both the total detected counts and the precision values calculated from the manual annotation, we compute the expected PI counts across the Pile dataset. Both total counts and expected counts are shown in Table 1. Email addresses and IP addresses are orders of magnitude more prevalent than phone numbers in the Pile. We run permutation tests, specifically a two-sample difference of means test, with 10,000 resamples to test whether R&R is significantly better than WIMBD across each of the four types of PI. We find that R&R is *significantly* better for email addresses and both sets of phone numbers (p-value < 0.05), but not for IP addresses.

3 Quantifying Memorization and Risk

To quantify memorization and its associated model parroting, we use the definition of p-memorization (Carlini et al., 2022). A string s is said to be p-memorized if a model $\mathcal{M}$, when prompted with a string s' of length p generates s verbatim with greedy decoding and concatenation $[s||s']$, is in the training data of $\mathcal{M}$. This definition gives us a framework to quantify the extent to which a model has memorized a training instance in a deterministic fashion.[3]

Metrics Since character-based PI instances are strings, we use the Levenshtein distance between a candidate instance of PI and its ground-truth to compute a similarity score, which we call ParrotScore. More formally for a candidate string s_1 and a ground-truth string s_2:

$$\textsc{ParrotScore}(s_1, s_2) = 1 - \frac{d_{levenshtein}(s_1, s_2)}{|s_2|} \quad (1)$$

In practice, if $|s_1| > |s_2|$, we take every substring of s_1 of size $|s_2|$ and choose the one that has the maximum ParrotScore with s_2. Since ParrotScore $\in [0, 1]$, a score of 1 signifies verbatim parroting, while a score of 0 indicates that there is not a single character that overlaps between the two strings. A low score such as 0.1 can still pose privacy risks; just three characters at the end of an email address can expose geographic information such as the country in which a person lives (*e.g.*, .nl).

4 Experiments

To measure PI parroting and memorization, we use the manually annotated and curated set of detections from our R&R detection suite, which contains 483 instances of PI. We experiment with 6 models from the Pythia suite with 160m, 410m, 1b, 1.4b, 2.8b, and 6.9b parameters, respectively. For each instance of PI, we find its associated prefix in the Pile and truncate this to a maximum of 80 tokens. Using this potentially truncated prefix as a prompt, we generate from the LM using greedy decoding and evaluate whether it parrots the ground truth PI instance using ParrotScore in equation 1.[4]

5 Results & Analysis

Model size vs. memorization: Figure 2 shows how different models with varying model sizes parrot PI across all PI types considered. Email

[2]Not only would annotating pretraining documents be infeasible without a large pool of annotators, but also using that pool would reveal PI publicly, drastically increasing privacy risks.

[3]A model $\mathcal{M}$ verbatim parroting an instance when prompted with a prefix of size p is similar to saying it has been p-memorized, if s' is the ground-truth PI.

[4]This is similar to measuring p-memorization for $p = 80$.

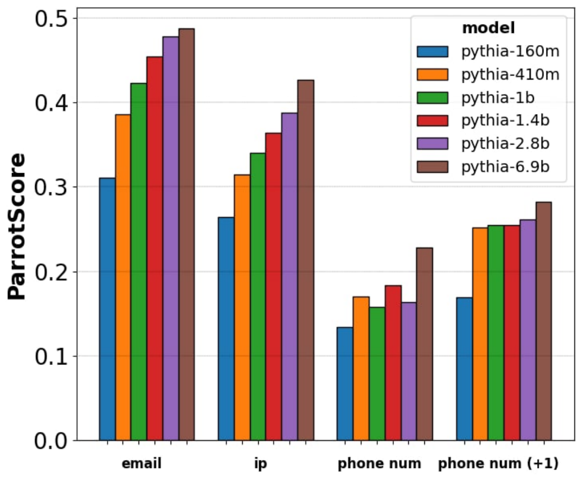

Figure 2: Here, we show the results of prompting each Pythia model with the prefix that occurs before each instance of PI and measuring the PARROTSCORE between the ground-truth PI and the model's greedily decoded generation across all PI types.

model sizes	email	ip	phone num	phone num +1
160m	2.34%	4.72%	1.64%	1.23%
410m	8.41%	7.09%	4.92%	2.47%
1b	12.62%	7.87%	3.28%	1.23%
1.4b	16.36%	11.81%	3.28%	1.23%
2.8b	19.63%	14.17%	1.64%	1.23%
6.9b	19.63%	14.17%	3.28%	4.94%

Table 2: Percent of total instances that are verbatim parroted. Note that this corresponds to the percent of instances that achieve a PARROTSCORE of 1. We find that verbatim parroting increases with model size and email addresses are the most likely to be parroted exactly.

addresses are the most parroted, with an average PARROTSCORE of greater than 0.3 for all models. This is closely followed by IP addresses and then by phone numbers. Model size and PARROTSCORE are positively correlated, but even the smallest models have high PARROTSCORE indicating that even small models are prone to memorization and verbatim parroting. In fact, the model with 410m parameters, the second smallest model we tested, has a similar PARROTSCORE to the 2.8b model for both types of phone numbers and has only a 0.1 lower PARROTSCORE on email and IP addresses.

Model size vs. verbatim parroting: Table 2 presents the percent of total instances that are parroted verbatim by each model. Verbatim parroting and model size are positively correlated, and email addresses and IP addresses contribute mostly to this trend. Both PI types are increasingly parroted: nearly 20% of all detected email addresses and more than 14% of IP addresses are exactly parroted by the two largest models. Phone numbers have a much lower verbatim parrot rate, which is not correlated with model size, indicating that phone numbers can be challenging for LMs to memorize.

Pretraining steps vs. memorization: The top plot of Figure 3 shows that even from only half of the pretraining (70,000 steps) to fully pretrained (143,000 steps), PARROTSCORE remains consistent, indicating that parroting exists for undertrained models and persists, even as models improve. Figure 3 shows results for the Pythia-6.9b model.

Prefix length vs. memorization: Recall that we are measuring p-memorization, which is highly dependent on the prefix length p. In all preceding experiments, we set this number to at most 80 tokens. To measure how the prefix length affects PARROTSCORE , we experimented with reducing p to 40, 20 and 10 tokens. This is the maximum size of the prefix that precedes the target PI in the original document. The lower part of Figure 3 indicates that PARROTSCORE is positively correlated with the prefix length, but, even with a token prefix of 10 tokens, the 6.9b model can parrot, achieving an average PARROTSCORE of 0.34. This indicates that models memorize PI rampantly and can be prompted with short prompts to parrot PI.

Memorization of constituent parts: We measure how each constituent part of a type of PI is parroted verbatim by each model in Tables 3–5. We split email addresses into two groups via the '@' symbol: usernames and domains, IP addresses into four groups via each of the three '.' symbols, and phone numbers into two groups: area code and rest of the numbers. For email addresses, both usernames and domains are verbatim parroted often, while for IP addresses each of the four constituents is parroted less often than the preceding one. For phone numbers, area codes are five times more likely to be parroted than the full number, increasing privacy risks, as area codes can be tied closely to location. For more details, see Appendix C.

6 Related Work

Documentation and PI in Data: The community prioritized documentation, especially personal information, copyright, and autonomy more strongly before the current LLM wave when datasets were smaller (McEnery, 2019). Subramani et al. (2023) analyzed both C4 and the Pile for character-based PI including emails and phone numbers. Concurrently, a preliminary version of these filters was used during the creation of the BigScience ROOTS corpus used to train the BLOOM suite of models (Scao et al., 2022; Laurenccon et al., 2022; Piktus et al., 2023). Elazar et al. (2023) built on top of this work to develop better regular expressions in the WIMBD suite. Our work improves upon WIMBD by developing better detectors for all PI types and annotating a larger set for better coverage.

Model Memorization and Privacy: Carlini et al. (2021) explore how language models like GPT-2 tend to memorize specific training examples, including PI, and that this can correlate with data frequency and model size. Other work investigate model forgetting, especially tailored to memorized examples throughout training (Jang et al., 2023; Jagielski et al., 2023; Carlini et al., 2022). Our work builds upon these: we quantify character-based PI parroting for the first time and analyze how model size, steps of pretraining, and prefix length affect it, further substantiating the claim that larger, better trained models tend to memorize and parrot more heavily. See Wei et al. (2025) for an overview of memorization in deep learning.

7 Conclusion

We develop the regexes and rules (R&R) detection suite for email addresses, US/Canada phone numbers, and IP addresses, improving the WIMBD detectors across all PI types. We measure the memorization of PI and find that verbatim parroting is rampant, especially as models get larger. This phenomenon is not isolated to larger models; even the smallest models pose privacy risks by parroting personal information verbatim. Consequently, we encourage the community to both develop better PI detectors and carefully filter and anonymize pretraining data when building language models.

8 Limitations

Annotating personal information is time-consuming. Since the data is private, out-sourcing the annotation process should not be done because it could expose PI. As a result, the sets we can annotate are small. Previous work annotated only a few hundred examples (Subramani et al., 2023; Elazar et al., 2023), whereas we annotated 1750 total detections. We hope that larger studies can be more comprehensive in annotating without exposing privacy risks. Most modern language models do not have open pretraining data, so figuring out what data a model has seen can be challenging. As a result, we focused on using the Pythia model suite because it was one of the only models that had a variety of model sizes, checkpoints during pretraining, and open pretraining data. OLMo also has different model sizes, checkpoints and open pretraining data (Groeneveld et al., 2024), but Dolma (Soldaini et al., 2024), its pretraining corpus, contains a PI filtering and anonymization step using the WIMBD detectors.

During the annotation process, we found that both detectors identify strings of 10 numbers that could be phone numbers, but they are not phone numbers. For example, `MAXINT=2147483647`. 214 is also a Dallas area code, so this could be flagged as a phone number. Additional rules to automatically eliminate detections such as these could help us build better detectors. A further extension of the post processing rules that we did not apply is to filter out subsets of data based on likelihood of false positives. With further study, adding rules about which subset of the Pile certain detectors operate on could greatly decrease the false positive rate.

9 Ethical Considerations

We hope that our R&R detectors help the community better anonymize and curate pretraining datasets such that the LMs that we deploy in the real world do not expose personal information. In addition, we hope that our analysis showing how significant personal information parroting is by models of all sizes convinces more large language modeling teams to think more carefully about sanitizing pretraining data. We choose to release the regexes used for the detector suite and code to run these detectors, hoping that the benefit of having these detectors and hoping the community uses them outweighs the harms of having an additional tool to detect PI.

Acknowledgments

We thank the anonymous reviewers for feedback, especially for suggesting the memorization of constituent parts experiment. Additionally, we thank members of MD's R3Lit lab for helpful discussions and feedback on early versions of the work.

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *ICML*. PMLR.

Jaydeep Borkar, Matthew Jagielski, Katherine Lee, Niloofar Mireshghallah, David A. Smith, and Christopher A. Choquette-Choo. 2025. Privacy ripple effects from adding or removing personal information in language model training. In *ACL Findings*.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *ICLR*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*.

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

Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hanna Hajishirzi, Noah A. Smith, and Jesse Dodge. 2023. What's in my big data? In *The Twelfth International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv*.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, et al. 2024. OLMo: Accelerating the science of language models. In *ACL*.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. 2023. Measuring forgetting of memorized training examples. In *ICLR*.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *ACL*.

Hugo Laurenccon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro von Werra, Chenghao Mou, Eduardo Gonz'alez Ponferrada, Huu Nguyen, Jorg Frohberg, Mario vSavsko, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben Allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, S. Longpre, Sebastian Nagel, Leon Weber, Manuel Sevilla Muñoz, Jian Zhu, Daniel Alexander van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa Etxabe, Kyle Lo, Manan Dey, et al. 2022. The bigscience roots corpus: A 1.6tb composite multilingual dataset. In *NeurIPS Datasets and Benchmarks*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*.

Tony McEnery. 2019. *Corpus linguistics*. Edinburgh University Press.

Microsoft. 2021. Presidio - data protection and anonymization api. [Release Version 2.2.23, released on Nov 16, 2021].

Aleksandra Piktus, Christopher Akiki, Paulo Villegas, Hugo Laurençon, Gérard Dupont, Sasha Luccioni, Yacine Jernite, and Anna Rogers. 2023. The ROOTS search tool: Data transparency for LLMs. In *ACL*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *EMNLP*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben

Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *ACL*.

Nishant Subramani, Samuel R. Bowman, and Kyunghyun Cho. 2019. Can unconditional language models recover arbitrary sentences? In *NeurIPS*.

Nishant Subramani, Sasha Luccioni, Jesse Dodge, and Margaret Mitchell. 2023. Detecting personal information in training corpora: an analysis. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*.

Nishant Subramani and Nivedita Suresh. 2020. Discovering useful sentence representations from large pretrained language models. *arXiv*.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *ACL Findings*.

Jiaheng Wei, Yanjun Zhang, Leo Yu Zhang, Ming Ding, Chao Chen, Kok-Leong Ong, Jun Zhang, and Yang Xiang. 2025. Memorization in deep learning: A survey. *ACM Computing Surveys*, (4).

A R&R Specifics

Here we present the specifics for each detector.

A.1 Email Addresses

The regex used is

```
(?:[a-z0-9]+(?:\.[a-z0-9!#$%&'*+/=?
^_‘{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c
\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\
[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?
:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?
\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?
|\[(?:(?:2(?:5[0-5]|[0-4][0-9])|1[0
-9][0-9]|[1-9]?[0-9])\.){3}(?:2(?:5
[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]
?[0-9])|[a-z0-9-]*[a-z0-9]:(?:[\x01
-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53
-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7
f])+\])
```

As mentioned before, we check for matches with the regular expression and then begin our set of rules to further filter detections. We check whether there is an “@" character to split the addressee and domain. We make sure these are nonempty strings. Next, we check whether there exists a starting or trailing period in the domain. If the detected instance has this, we flag this as a false detection. Lastly, we make sure that there exists a period (“.") in the domain.

A.2 IP Addresses

For IP addresses, we have two regexes, an IPv4 and an IPv6 pattern:

```
ipv4 = (?:(?:25[0-5]|2[0-4][0-9]|[01]?
[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4]
[0-9]|[01]?[0-9][0-9]?)

ipv6 = (?:^|(?<=\s))(?:(?:[0-9a-fA-F]
{1,4}:){7,7}[0-9a-fA-F]{1,4}|(?:[0-9a
-fA-F]{1,4}:){1,7}:|(?:[0-9a-fA-F]{1,
4}:){1,6}:[0-9a-fA-F]{1,4}|(?:[0-9a-f
A-F]{1,4}:){1,5}(?::[0-9a-fA-F]{1,4})
{1,2}|(?:[0-9a-fA-F]{1,4}:){1,4}(?::[
0-9a-fA-F]{1,4}){1,3}|(?:[0-9a-fA-F]{
1,4}:){1,3}(?::[0-9a-fA-F]{1,4}){1,4}
|(?:[0-9a-fA-F]{1,4}:){1,2}(?::[0-9a-
fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(?:
(?::[0-9a-fA-F]{1,4}){1,6})|:(?:(?::[
0-9a-fA-F]{1,4}){1,7}|:)|fe80:(?:(?::
[0-9a-fA-F]{0,4}){0,4}%[0-9a-zA-Z]{1,
})|::(?:ffff(?::0{1,4}){0,1}:){0,1}(?
:(?:(?:25[0-5]|(?:2[0-4]|1{0,1}[0-9])
{0,1}[0-9])\.){3,3}(?:25[0-5]|(?:2[0-
4]|1{0,1}[0-9]){0,1}[0-9]))|(?:(?:[0-
9a-fA-F]{1,4}:){1,4}(?:(?:25[0-5]|(?:
2[0-4]|1{0,1}[0-9]){0,1}[0-9])\.){3,3
}(?:25[0-5]|(?:2[0-4]|1{0,1}[0-9]){0,
1}[0-9])))(?=\s|$)
```

After running the regular expression based detectors, we filter the detected IP addresses using the following set of rules. First, we check if any of the following common words occur in the micro context window of 20 characters preceding the detected PI span:

```
‘isbn’, ‘doi’, ‘#’, ‘grant’, ‘award’,
‘nsf’, ‘patent’, ‘usf’, ‘edition’,
‘congress’, ‘appeal’, ‘claim’,
‘exhibit’, ‘serial’, ‘pin’, ‘receipt’,
‘case’, ‘tracking’, ‘ticket’, ‘route’,
‘ wo ’, ‘volume’
```

These words often indicate a type of number that could have a syntax similar to an IPv4 address. This is a much larger problem for phone numbers, so we also do this for phone numbers. Next, we check whether the prefix has alphabetic characters. We take at most the 50 characters preceding the detected pi span and see if at least 10% of them are alpha numeric. This is to filter out arbitrary sets of numbers.

A.3 Phone Numbers

For phone numbers we have two regular expression based detectors *phone numbers*:

```
\s+\(?(\d{3})\)?[-\. ]*(\d{3})[-. ]?
(\d{4})(?!\d)
```

and *global phone numbers* (US/Canadian phone numbers in a global context with the country code):

```
\s+(?:\+1|1)[-\. ]*\(?(\d{3})\)?[-\. ]
*(\d{3})[-\. ]?(\d{4})(?!\d)
```

After running the regular expression based detectors, we filter using a set of rules. First, we check if any of the common words (the words used when filtering IP addresses above) occur in the micro context window of 20 characters preceding the detected PI span. These words often indicate a type of number that can often have 9, 10, or 11 digits. Next, we check whether the prefix has the sufficient number of alphabetic characters, identical to how IP addresses were processed. For phone numbers, this helps filter out things like html polygons or random sets of numbers without context like dumps of arbitrary numbers. Next, we standardize the detected number and validate the area code: excluding numbers with an area code starting with 0 or 1 and verify that the area code is a valid one. After doing this, we validate the central office code. Lastly, we exclude a set of placeholder numbers that include 1234567890, 2345678910, MAXINT, 73737373 and 3141592653 (digits of π).

B Ablation Analysis

Here, we look at the impact of pretraining steps and prefix length on memorization. In Figure 3, we find that the models parrot even when only halfway through training. The Pythia models are trained for 143,000 steps, and even from 70,000 steps as mentioned before, PARROTSCOREremains constant. Additionally, prefix length is highly correlated with PARROTSCORE. However, even with as little as 10 tokens in the prefix, PI memorization is rampant, indicating severe risk.

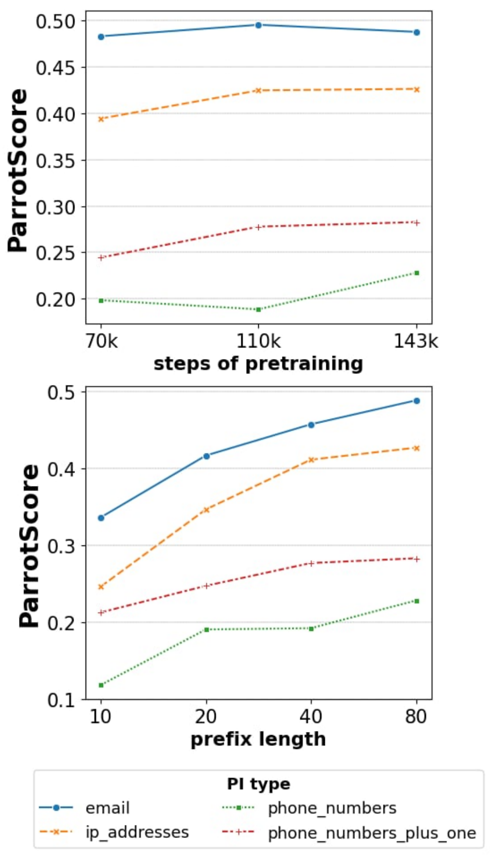

Figure 3: The effect of the number of pretraining steps (top) and prefix length (bottom) on PARROTSCORE across PI types for the Pythia-6.9b model.

C Memorization of Constituent Parts:

Here, we measure how each constituent part of a type of PI is verbatim parroted by each model. To do this, we first parse IP addresses (IPv4) into their four constituent groups separated by a period (e.g. 8.8.8.8 turns into [8,8,8,8]). Each of these 4 groups are measured separately. A candidate generation that produces "12.8.8 abcd" will turn into [12, 8, 8 abcd, ""] and comparing that to 8.8.8.8 will lead to verbatim parroting of only group 2. We parse email addresses into two groups: the username and domain separated by the symbol '@' because email addresses are usually separated into these two groups. We parse phone numbers into two groups: area code and the rest of the digits following that. Since we are only considering US/Canada phone numbers that always start with 1, we did not emphasize splitting out the country code. We measure verbatim parroting for all model sizes at the 143,000 iteration checkpoint for a prefix length of 80. This is an extension of Table 2, where we report the percent of instances that are verbatim parroted.

model sizes	username	domain
160m	11.22%	12.15%
410m	15.42%	20.09%
1b	20.09%	24.77%
1.4b	20.56%	28.50%
2.8b	25.70%	31.31%
6.9b	26.17%	30.84%

Table 3: Percent of total instances that are verbatim parroted for the constituent parts of email addresses (the username and domain). Note that this corresponds to the percent of instances with that component achieving a PARROTSCORE of 1. We find that verbatim parroting increases with model size.

In Table 3, we find that for email addresses, both usernames and domains are often parroted verbatim, as much as 31% of instances have a parroted domain. Usernames are slightly less parroted, but even those up to 26.17% of instances have a verbatim parroted username, underscoring significant risk. For IP addresses in Table 4, each successive constituent was slightly less likely to be parroted than the previous group. We hypothesize that this is due to the nature of left-to-right autoregressive decoding and not due to any other confounding factors. For the 6.9b model, in about 32% of instances either group 1 or group 2 were verbatim parroted, while only 14% of IP addresses overall were verbatim parroted.

model sizes	grp1	grp2	grp3	grp4
160m	18.11%	13.39%	13.39%	9.45%
410m	18.90%	19.69%	14.96%	11.02%
1b	21.26%	21.26%	20.47%	11.02%
1.4b	25.20%	24.41%	21.26%	14.17%
2.8b	26.77%	25.20%	22.83%	18.90%
6.9b	32.28%	31.50%	24.41%	19.69%

Table 4: Percent of total instances that are verbatim parroted for the constituent parts of IP addresses (split on each '.' character grp1-grp4). Note that this corresponds to the percent of instances with that component achieving a PARROTSCORE of 1. We find that verbatim parroting increases with model size.

For phone numbers in Table 5, which had a relatively low verbatim parrot rate ($\sim 3\%$), we find that area codes are much more likely to be parroted, increasing privacy risk as this can be closely related to location. We find that for the 6.9b model, even when only 3.28% of instances are parroted by the model, in 19.67% of cases the area code is parroted exactly. Taken together, these results further underscore our point that PI memorization and parroting is a risk that needs to be mitigated.

model sizes	area code	rest of the number
160m	8.20%	1.64%
410m	13.11%	4.92%
1b	11.48%	3.28%
1.4b	14.75%	3.28%
2.8b	9.84%	1.64%
6.9b	19.67%	3.28%

Table 5: Percent of total instances that are verbatim parroted for the constituent parts of phone numbers (split by area code and rest of the number). Note that this corresponds to the percent of instances with that component achieving a PARROTSCORE of 1. We find that verbatim parroting increases with model size for area codes generally, but not for the rest of the number.