Low-Perplexity LLM-Generated Sequences and Where To Find Them

Arthur Wuhrmann[1], **Anastasiia Kucherenko**[2], **Andrei Kucharavy**[3]
[1]École Polytechnique Fédérale de Lausanne, Switzerland
[2]Institute of Entrepreneurship and Management, HES-SO Valais-Wallis, Switzerland
[3]Institute of Informatics, HES-SO Valais-Wallis, Switzerland
Correspondence: `arthur.wuhrmann@epfl.ch`

Abstract

As Large Language Models (LLMs) become increasingly widespread, understanding how specific training data shapes their outputs is crucial for transparency, accountability, privacy, and fairness. To explore how LLMs leverage and replicate their training data, we introduce a systematic approach centered on analyzing low-perplexity sequences—high-probability text spans generated by the model. Our pipeline reliably extracts such long sequences across diverse topics while avoiding degeneration, then traces them back to their sources in the training data. Surprisingly, we find that a substantial portion of these low-perplexity spans cannot be mapped to the corpus. For those that do match, we quantify the distribution of occurrences across source documents, highlighting the scope and nature of verbatim recall and paving a way toward better understanding of how LLMs training data impacts their behavior.

1 Introduction

While Large Language Models (LLMs) are increasingly applied across various domains, the ways in which they leverage their training data during inference remains only partially understood (Review, 2024; Bender et al., 2021; Liang et al., 2024). Research on training data attribution (TDA) in LLMs (Carlini et al., 2021; Cheng et al., 2025) aims to answer this question, but identifying which specific parts of the data contribute to a model's output. TDA is considered essential for enhancing transparency, effective debugging, accountability, and addressing concerns related to privacy and fairness in LLMs (Cheng et al., 2025; Akyurek et al., 2022; Liu et al., 2025a).

Currently, there are two principal approaches for TDA - causal and similarity-based. Causal TDA uses direct experimental methods such retraining and gradient-based techniques that quantify the precise causal contribution of individual training samples to model outputs (Guu et al., 2023; Kwon et al., 2023; Pan et al., 2025; Akyurek et al., 2022; Chang et al., 2024; Wu et al., 2024). While offering theoretical guarantees about causality, their computational cost increases dramatically with model size, making them infeasible in practice.

Similarity-based TDA (Liu et al., 2025a; Carlini et al., 2021; Khandelwal et al., 2020; Deguchi et al., 2025) identifies training samples that resemble model outputs, assuming similar content likely influenced generation. While similarity does not guarantee causal influence and this attribution is approximate, this approach is computationally efficient and scales well to large models, making it feasible in practice. Similarity-based TDA includes approaches such as nearest-neighbor searches in embedding spaces and exact string matching for verbatim recall. In this paper, we focus on the latter, which connects to the established field of novelty (McCoy et al., 2023; Merrill et al., 2024) and memorization in LLMs (Carlini et al., 2023b; Al-Kaswan et al., 2024; Carlini et al., 2023a; Feldman and Zhang, 2020; Prashanth et al., 2025), studying instances where models produce verbatim recall of training data. Recently, the first tool for efficient TDA based on exact memorization was introduced (Liu et al., 2025a), underscoring the practical importance of such approaches.

In this paper, we study how low-perplexity sequences in LLM-generated output are connected to its verbatim recall. Perplexity is a standard metric used to evaluate a model's ability to predict tokens, with lower perplexity indicating higher confidence in its predictions. It is widely employed for model evaluation, fine-tuning, comparison and assessing text generation quality. In the context of training data attribution (TDA), there is a hypothesis that long low-perplexity sequences suggest either degeneration or verbatim copying from the training data (Gao et al., 2019; Prashanth et al., 2025). We aim to empirically test this statement, while propos-

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 774–783
July 28-29, 2025 ©2025 Association for Computational Linguistics

ing a method to better understand LLMs' verbatim recall through low-perplexity analysis.

We present an open-source pipeline[1] designed to identify and trace low-perplexity spans in LLM outputs. By targeting specialized domains with rich, distinctive terminology, our approach efficiently extracts long, low-perplexity segments suitable for in-depth analysis. These segments are then mapped back to their origins using indexing and search tools. Although we experimented with both the well-established Elasticsearch (Gormley and Tong, 2015) and the recently emerged state-of-the-art Infinigram (Liu et al., 2025b), we report only Infinigram results due to its superior scalability and efficiency for large-scale mapping.

Our analysis provides deeper insights into how LLMs recall and replicate information. First, we observe that results vary depending on the topic of LLM input, its representation in the training data, and its degree of specialization. Second, we find that a significant portion of low-perplexity spans, ranging from 30% to 60%, cannot be matched to the training data. For those that can be matched, we further categorize different types of memorization behaviors, noting that verbatim recall can arise for various reasons. Finally, this classification allows us to quantify that approximately 20% of low-perplexity spans correspond to a number of documents small enough for manual review.

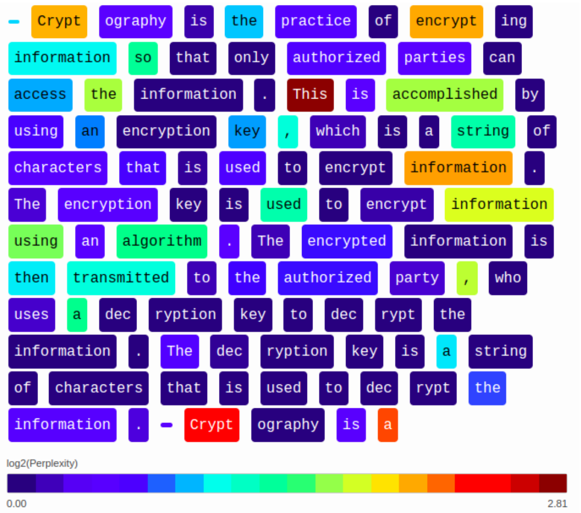
2 Experimental setup

LLM model and training data

To study low-perplexity sequences we use the Pythia model (Biderman et al., 2023) with size of 6.9 billion parameters trained on *The Pile* (Gao et al., 2020), which transforms into 300 billion tokens using Pythia tokenizer (Biderman et al., 2023), with a vocabulary size $|V| = 50,254$.

Choosing topics and prompts

To follow our goal of finding low-perplexity sequences, we focus on keyword-specific topics for this study. Therefore, we choose **genetics, nuclear physics, drugs, and cryptography**, specialized domains in which the team has experience to verify the validity of LLM outputs. Since we work with The Pile dataset, those topics are represented at least as part of its Wikipedia subset.



- Crypt ography is the practice of encrypt ing
information so that only authorized parties can
access the information . This is accomplished by
using an encryption key , which is a string of
characters that is used to encrypt information .
The encryption key is used to encrypt information
using an algorithm . The encrypted information is
then transmitted to the authorized party , who
uses a dec ryption key to dec rypt the
information . The dec ryption key is a string
of characters that is used to dec rypt the
information . - Crypt ography is a
log2(Perplexity)
0.00
2.81

Figure 1: Visualization of a generated subsequence that contains two different low-perplexity sequences longer than 5 tokens. We have `decryption key to decrypt the information` and `string of characters that is used to decrypt`. Both having 9 tokens, they will be split in $9 + 1 - 6 = 4$ windows of 6-contiguous tokens each.

In total, for each topic, we select 40 articles from the Wikipedia version included in the Pile and extract a random quote consisting of 20 to 40 tokens. This quote serves as a prompt for the Pythia model to complete and extend. For each prompt we run 5 generations to average the results. This approach provides 200 prompts per topic and 800 prompts in total.

LLM output generation and perplexities

LLMs generate output sequentially—token by token—by sampling the next token based on its logits values and key parameters: top_k, which restricts choices to the top k most probable words; top_p, which selects the smallest set of words with a cumulative probability of p; and temperature T, which controls randomness. We set $\text{top}_k = 20$, $\text{top}_p = 0.8$, and $T = 0.7$, with alternative configurations discussed in Sec. 3.3.

The exact definition of the generation probability of each token (x_i) based on the previous tokens $(x_{<i})$ is

$$p(x_i|x_{<i}) = \frac{\exp(z_i/T)}{\sum_{j=1}^{|V|} \exp(z_j/T)},$$

where z_i are the raw logits and $|V|$ is the vocabulary size of the model. Then, the *token perplexity* is:

$$P(x_i) = \frac{1}{p(x_i|x_{<i})}. \quad (1)$$

---

[1] The code is available at `https://github.com/Reliable-Information-Lab-HEVS/HAIDI-Graphs`

We define a **low-perplexity sequence** as a contiguous part of the LLM output where *each token has a perplexity threshold* $\log_2(P) \leq 0.152$ *in base 2*, corresponding to a *probability threshold of* 0.9 *or higher*. These sequences have different lengths, so to compare the matches in the training data, we focus on their fixed-size subsequences. We call those **low-perplexity windows** and focus our choice on size of 6 tokens. The choice of a 6-token window is justified as it is short enough to capture meaningful low-perplexity spans while being long enough to avoid random matches. Fig. 1 shows a visualization of the generated tokens and perplexities values.

Matching to the training data and its quality

Finally, we map low-perplexity windows to the training data. To achieve this, we use Infini-gram (Liu et al., 2025b). Once a low-perplexity window is matched to the training data, we estimate the significance of its text. We do this using perplexity values (as defined in Equation 1), this time without additional context (i.e., tokens preceding the window), which is also known as *standalone perplexity*. We denote it as

$$\hat{P}(x_k, \dots, x_{k+n}) = 2^{-\frac{1}{n}\sum_{i=k}^{k+n} \log_2 p(x_i|[x_k, \dots, x_{i-1}])}$$

Low standalone perplexity indicates that the generated text is fluent, coherent, and resembles human-written language (Gonen et al., 2024).

3 Results

3.1 Descriptive analysis of low-perplexity windows

We begin by identifying all low-perplexity sequences across the four chosen topics. The warm-up statistics in Table 1 show that the average lengths of these sequences do not vary significantly between topics, and our choice of a fixed window size of 6 is sufficiently modest.

Topic	$\bar{L}$	σ_L
Crypt2ography	12	11
Drugs	14	15
Genetics	14	14
Nuclear physics	13	12

Table 1: $\bar{L}$ (resp. σ_L) represents the average (resp. standard deviation) of the token lengths for low-perplexity sequences with at least 6 tokens.

From selected low-perplexity sequences, we pass a sliding window of 6 tokens and stride 1 and proceed to our main interest – low-perplexity windows matched to the training data. We denote the number of occurances by c. Figure 2 presents the comparison of windows at least with one match across different topics. We observe having significantly more of long low-perplexity sequences on drugs. We believe this is due to the presence of repetitive long drug names and their strong connection to biomedical literature, which is widely represented in the Pile dataset through the inclusion of PubMed. On the other side, it is likely that nuclear physics is less present in the Pile, which explains the lower number of counts.

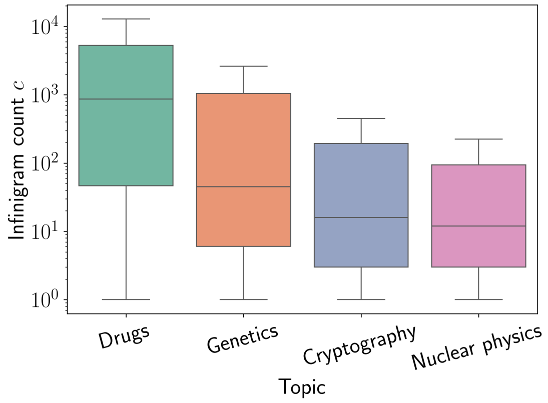


Figure 2: Boxplots comparing the number of matches of low-perplexity windows that occur in the training data, across different topics.

Above, only windows with at least one exact match in the training data are considered. While one might expect low-perplexity windows to almost always have matches, we verify this experimentally (Table 2). Surprisingly, *only 40% of low-perplexity windows have at least one exact match* ($N_{c>0}$). We also observe varying match counts across topics, likely due to differences in their specialization and corpus representation.

Topic	N	$N_{c>0}$	$N_{c>0}/N$	N_{rep}/N
Cryptography	1336	505	38%	32%
Drugs	988	659	67%	7.9%
Genetics	1337	481	36%	29%
Nuclear physics	1040	264	25%	15%
Total	4701	1909	41%	21%

Table 2: The total number of low-perplexity windows N for each topic, number and percentage of those windows that have exact matching the training data $N_{c>0}$. N_{rep}/N is the percentage of low-perplexity sequences repeating the prompt (see Appendix C).

Finally, examining the matched windows, we find that a significant fraction partially repeats the prompt (N_{rep}). We suspect this is due to the specialized keywords in the prompt and therefore we retain these repetitions for further analysis. Appendix C presents an example of such repetition.

3.2 The nature of low-perplexity sequences

Using two additional measures, we explore the behaviors exhibited by the model when generating low-perplexity sequences (Figure 3). First, we revisit the concept of stand-alone perplexity to assess how human-like the generated text appears. Second, we categorize the low-perplexity windows into four groups based on their number of matches in the training data (c), reflecting different recall and generalization behaviors. Since these behaviors can overlap, the group boundaries are not sharply defined. Therefore, in Figure 3, we intentionally use a color gradient to illustrate the smooth transition between categories. While we indicate specific thresholds for the match count c below, these values are adjustable and intended to aid interpretation rather than impose strict divisions. Particular examples of each behavior can be found in Appendix B.

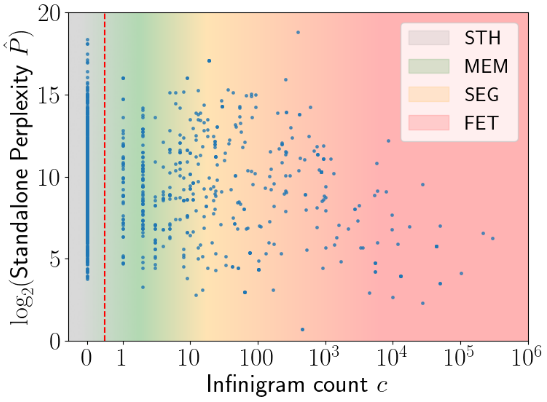


Figure 3: Illustration of the low-perplexity sequences, for the Cryptography topic.

- **Synthetic coherence ($c = 0$):** These windows are synthetically generated by the model without any exact matches in the training data. Interestingly, the stand-alone perplexities vary widely, including high values. However, as shown in Appendix B, even the generations with the highest perplexity scores remain coherent and are not non-sensical.
- **Memorization** ($0 < c < 5$) The model has generated text containing highly specific knowledge, which can be traced back with high precision to its origins in the training data. Such traceability is particularly valuable for identifying instances of private and sensitive data leakage, memorized and reproduced by the model. An example is given in Appendix D.
- **Segmental replication** ($5 \leq c < 50$) These windows contain relatively niche information that appears across multiple sources, often reflecting standardized phrases or terminology within specific domains. Alongside memorization, segmental replication helps efficiently trace LLM outputs to their origins, revealing how specialized knowledge is represented.
- **Frequently encountered text** ($50 < c$) These windows correspond to common phrases or widely used expressions that appear frequently across many documents in the training data. When c becomes very large, it typically reflects standardized text such as legal disclaimers, licensing terms or HTML tags (i.e., `<div><\div>`), indicating heavy repetition across the corpus.

While the thresholds of 5 and 50 were chosen arbitrarily, fixing them enables consistent counting and comparison across topics, as shown in Table 3. Notably, around 20% of low-perplexity windows fall into the memorization and segmental replication categories, matching to a number of documents small enough to be manually reviewed.

Topic	STH	MEM	SEG	FET
Cryptography	62%	11%	13%	14%
Drugs	33%	7.5%	9.3%	50%
Genetics	64%	7.7%	11%	17%
Nuclear physics	75%	8.1%	9.3%	8%

Table 3: Distribution of categories across topics. Categories: Synthetic coherence (STH), Memorization (MEM), Segmental replication (SEG), and Frequently encountered text (FET).

3.3 LLM size and its generation parameters

In the previous experiments, we used the Pythia-$6.9B$ model with fixed generation parameters, as described in Section 2. In this section, we repeat the experiments with alternative model settings and justify our initial choice.

First, we replicate the experiments across the Pythia model scaling suite (Table 4). As model size increases, we observe a clear drop in both the number of low-perplexity windows and their matches to the training data. This supports our choice of the 6.9B model, which offers more meaningful responses, while any matching results would only improve in smaller models.

Size	N	$N_{c>0}$	$N_{>0}/N$	N_{rep}	$\hat{P}$
70M	8528	2874	34%	118	9.2
160M	3676	1306	36%	428	8.4
410M	2274	716	31%	470	8.4
1B	2766	878	32%	752	8.6
1.4B	2123	673	32%	334	8.2
2.8B	1714	488	28%	402	8.6
6.8B	1337	481	36%	386	8.5

Table 4: Number of low-perplexity sequences and matches when varying the model sizes. Done on the Genetics topic.

Further, we study the impact of varying the temperature parameter, which controls the LLM generation randomness (Table 5).

T	N	$N_{c>0}$	$N_{>0}/N$	N_{rep}	$\hat{P}$
0.2	8787	2908	33%	743	8.7
0.3	6127	1918	31%	589	8.5
0.4	4523	1461	32%	598	8.9
0.5	3297	1091	33%	560	8.8
0.6	1913	659	34%	310	8.6
0.7	1337	481	36%	386	8.5

Table 5: Number of low-perplexity sequences and matches when varying the temperature. Done on the Genetics topic.

Lower temperature makes the model more deterministic, favoring high-probability tokens. We observe that it leads to a greater number of low-perplexity windows, however increases degeneration and more repetitive patterns in the LLM outputs. Also, interestingly, the overall percentage of non-zero matches, as well as the stand-alone perplexity, remains largely unchanged. These results explain our preference for a temperature value of 0.7 — it provides a meaningful number of low-perplexity windows for analysis while reducing the extent of repetition.

4 Conclusion

We proposed a pipeline to identify and analyze low-perplexity sequences in LLM outputs. We categorized sequences by their match frequency in the training data and identified four distinct behaviors. We also conducted a statistical analysis of these categories, notably finding that many low-perplexity sequences do not match the corpus at all. This approach improves understanding of how models recall learned information and, in some cases, enables more efficient training data attribution.

5 Limitations

Our threshold selection approach in Figure 3 relies on estimations that require more rigorous examination. The absence of clear clustering suggests these thresholds may represent gradual transitions rather than abrupt boundaries. We also found that high standalone perplexity does not consistently indicate nonsensical text (see Appendix B), challenging its reliability as a degeneration detector. For future work, we encourage exploring alternative evaluation methods, such as model-as-a-judge approaches (Zheng et al., 2023), to more accurately identify text degeneration.

A methodological limitation worth addressing is the potential bias introduced by our prompt generation technique. Since some prompts originate from the Pile dataset, this artificially inflates certain sequence counts. Further studies incorporating manually crafted prompts would help quantify and mitigate this bias.

Additionally, trying different model sizes, and including a wider set of prompts, from non-scientific domains without specific keywords would allow to state the limitations more clearly.

Finally, we note that our model uses the Pythia tokenizer, whereas Infinigram relies on the LLaMA-2 tokenizer. As a result, certain spans—especially verbatim sequences—may fail to align across models despite being present in the training data. We recommend performing indexing with the same tokenizer used at inference time to avoid such mismatches.

Our pipeline may serve as an additional tool for Training Data Attribution (TDA) investigations. We anticipate future research exploring the relationships between low-perplexity windows and sequences, as briefly discussed in Appendix D. Additionally, comparative analyses between our method and other state-of-the-art TDA approaches would be valuable for establishing best practices in this emerging field, alongside with efficiency measurements.

6 Ethics statements

Training data extraction is a threat to user privacy, as this can be used to find Personally Identifiable Information (PII) such as leaked passwords, address or contact information (Brown et al., 2022). We try to mitigate this in the following way. First, we work on a publicly available model, and use examples from Wikipedia, also publicly available. However, we acknowledge that the Pile dataset, which was used to train the Pythia models, contains copyrighted material (Monology, 2021). Given these concerns, we advocate for future research to prioritize copyright-compliant datasets that respect creators' intellectual property rights while advancing our understanding of model behavior. On the other hand, our work contribute to training data transparency, and can help to detect copyright infringement. We also recall that our method requires to possess an indexing of the training data, which is not the case for the state-of-the-art models. We believe that the impact of this paper does not present direct major risks and encourage further work in this direction.

For transparency, we give an estimation of the CO_2 emitted by the computation. We used approximately 120 hours of GPU with an average consumption of 250 W, and considering the CO_2 emissions per kilowatt-hour in the region we are located in to be $38.30\,\mathrm{gCO_2eq/kWh}$ (Power, 2024), this totals to $120 \times 0.25 \times 38.30 = 1.1\,\mathrm{kgCO_2eq}$.

Finally, additional generative AI tools were used solely to assist with reformulating parts of the text and code for improved clarity and readability.

Acknowledgments

The authors are thankful to Alexander Sternfeld and Prof. Antoine Bosselut for their valuable input on the paper, and to the anonymous reviewers of ACL 2025 for their constructive comments. We additionally thank Prof. Bosselut for hosting Arthur Wuhrmann (AW) in his lab during the course of this work. Andrei Kucharavy (ADK) and Anastasiia Kucherenko (AAK) are supported by the CYD Campus, armasuisse W+T, ARAMIS AR-CYD-C-025 grant.

Contributions

- Conceptualization: AAK, ADK;
- Methodology, Software, Data Curation, Visualization, and Writing - Original Draft: AW, ADK;
- Investigation, Writing - Review & Editing: AW, AAK, ADK;
- Supervision, Project Administration and Funding Acquisition: ADK.

References

Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Towards tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2429–2446, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ali Al-Kaswan, Maliheh Izadi, and Arie van Deursen. 2024. Traces of memorisation in large language models for code. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, page 1–12. ACM.

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 *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

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. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? *Preprint*, arXiv:2202.05520.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023a. Extracting training data from diffusion models. *Preprint*, arXiv:2301.13188.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023b. Quantifying memorization across neural language models. *Preprint*, arXiv:2202.07646.

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. 2021. Extracting training data from large language models. *Preprint*, arXiv:2012.07805.

Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. 2024. Scalable influence and fact tracing for large language model pretraining. *Preprint*, arXiv:2410.17413.

Deric Cheng, Juhan Bae, Justin Bullock, and David Kristofferson. 2025. Training data attribution (tda): Examining its adoption & use cases. *Preprint*, arXiv:2501.12642.

Hiroyuki Deguchi, Go Kamoda, Yusuke Matsushita, Chihiro Taguchi, Kohei Suenaga, Masaki Waga, and Sho Yokoi. 2025. Softmatcha: A soft and fast pattern matcher for billion-scale corpus searches. *Preprint*, arXiv:2503.03703.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: discovering the long tail via influence estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation degeneration problem in training natural language generation models. *Preprint*, arXiv:1907.12009.

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. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2024. Demystifying prompts in language models via perplexity estimation. *Preprint*, arXiv:2212.04037.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide*. O'Reilly Media.

Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. 2023. Simfluence: Modeling the influence of individual training examples by simulating training runs. *Preprint*, arXiv:2303.08114.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. *Preprint*, arXiv:1911.00172.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. 2023. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *CoRR*, abs/2310.00902.

Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D Manning, and James Y. Zou. 2024. Mapping the increasing use of llms in scientific papers. *Preprint*, arXiv:2404.01268.

Jiacheng Liu, Taylor Blanton, Yanai Elazar, Sewon Min, YenSung Chen, Arnavi Chheda-Kothary, Huy Tran, Byron Bischoff, Eric Marsh, Michael Schmitz, Cassidy Trier, Aaron Sarnat, Jenna James, Jon Borchardt, Bailey Kuehl, Evie Cheng, Karen Farley, Sruthi Sreeram, Taira Anderson, and 12 others. 2025a. Olmotrace: Tracing language model outputs back to trillions of training tokens. *Preprint*, arXiv:2504.07096.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2025b. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *Preprint*, arXiv:2401.17377.

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.

William Merrill, Noah A. Smith, and Yanai Elazar. 2024. Evaluating n-gram novelty of language models using rusty-dawg. *Preprint*, arXiv:2406.13069.

Monology. 2021. Pile uncopyrighted. `https://huggingface.co/datasets/monology/pile-uncopyrighted`. Accessed: May 17, 2025.

Yijun Pan, Taiwei Shi, Jieyu Zhao, and Jiaqi Ma. 2025. Detecting and filtering unsafe training data via data attribution.

Low-Carbon Power. 2024. Carbon intensity of electricity in switzerland. Accessed: May 17, 2025.

USVSN Sai Prashanth, Alvin Deng, Kyle O'Brien, Jyothir S V, Mohammad Aflah Khan, Jaydeep Borkar, Christopher A. Choquette-Choo, Jacob Ray Fuehne, Stella Biderman, Tracy Ke, Katherine Lee, and Naomi Saphra. 2025. Recite, reconstruct, recollect: Memorization in lms as a multifaceted phenomenon. *Preprint*, arXiv:2406.17746.

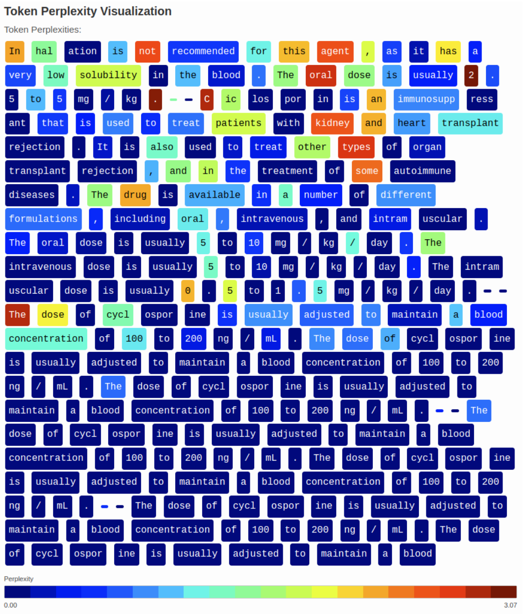
MIT Technology Review. 2024. Large language models can do jaw-dropping things. but nobody knows exactly why. Accessed: 2025-05-18.

Kangxi Wu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2024. Enhancing training data attribution for large language models with fitting error consideration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14131–14143, Miami, Florida, USA. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Visualization of degeneration

While we did not include degeneration region in Fig. 3, we still encountered it during our experiments. Here, by degeneration, we refer to undesirable patterns in generated text, such as nonsensical or incoherent outputs, excessive repetition, and looping behaviors—where the model repeatedly generates the same tokens or phrases in a cyclic manner. Fig. 4 shows an example of it. This exclusion stemmed from two observations: the repetitive patterns extended beyond our window size parameters, and the degenerated text displayed surprisingly low standalone perplexity values. These findings highlight a limitation in using perplexity-based metrics alone for degeneration detection and suggest the need for complementary approaches.



```
In hal ation is not recommended for this agent , as it has a
very low solubility in the blood . The oral dose is usually 2 .
5 to 5 mg / kg . ‒ ‒ C ic los por in is an immunosupp ress
ant that is used to treat patients with kidney and heart transplant
rejection . It is also used to treat other types of organ
transplant rejection , and in the treatment of some autoimmune
diseases . The drug is available in a number of different
formulations , including oral , intravenous , and intram uscular .
The oral dose is usually 5 to 10 mg / kg / day . The
intravenous dose is usually 5 to 10 mg / kg / day . The intram
uscular dose is usually 0 . 5 to 1 . 5 mg / kg / day . ‒ ‒
The dose of cycl ospor ine is usually adjusted to maintain a blood
concentration of 100 to 200 ng / mL . The dose of cycl ospor ine
is usually adjusted to maintain a blood concentration of 100 to 200
ng / mL . The dose of cycl ospor ine is usually adjusted to
maintain a blood concentration of 100 to 200 ng / mL . ‒ ‒ The
dose of cycl ospor ine is usually adjusted to maintain a blood
concentration of 100 to 200 ng / mL . The dose of cycl ospor ine
is usually adjusted to maintain a blood concentration of 100 to 200
ng / mL . ‒ ‒ The dose of cycl ospor ine is usually adjusted to
maintain a blood concentration of 100 to 200 ng / mL . The dose
of cycl ospor ine is usually adjusted to maintain a blood
```

Figure 4: Example of the perplexities of an output that degenerates.

B Examples of texts per category.

Tab. 6 presents examples of low-perplexity windows belonging to different categories. We also added example of high perplexities.

category	**text**
Frequently encountered text	– synthetic cannabinoid. – a function that takes as input an – Standards and Technology (NIST)
Memorization	– alcohol, sugar, water, and – to the evaluation of a cryptographic – of information that is used to encrypt
Segmental replication	– has been defined as "the study – used for PET and SPECT imaging – understanding of the genetic basis of common
Synthetic coherence	– and genetics. fireball starts to form. The – the exchanged keys are computationally indistinguishable from – . Developmental genetics is also the
High standalone perplexity ($\log_2(\hat{P}) > 12$)	– random cipher digit stream (key – ion CN – is also being removed – a sudden influx of fast neutrons

Table 6: Examples of text fragments and their categories.

C Example of repetition.

We show here an example of the model repeating the prompt. The repetition is highlighted in red. The end has been cut for better readability.

Example of Prompt Repetition

Prompt:

There are a number of standards related to cryptography. Standard algorithms and protocols provide a focus for study; standards for popular applications attract a large amount of cryptanalysis.

Output:

Standards provide a focus for cryptanalysis. Standards for popular applications attract a large amount of cryptanalysis. There are a number of standards related to cryptography. Standard algorithms and protocols provide a focus for study; standards for popular applications attract a large amount of cryptanalysis. Standards provide a focus for cryptanalysis. Standards for popular applications attract [...]

D Surrounding of sequences match

When a sequence has a low number of matches (in the memorization part, see Sec. 3.2) in the training data, one can look at the original document containing the sequence. Below is an example. The part in bold has one exact match to the Pile, and the extract is shown below.

Segment match

Prompt:

A message authentication code (MAC) is a short piece of information used to authenticate a message.

Output:

The MAC is a function of a secret key and the message. The message is the information being protected. The key **is a secret known only to the sender and** receiver. The MAC is a function of the key and the message. The MAC is a short piece of information used to authenticate a message. [...]

Document matching:

[...] HMAC is a well-known algorithm for generating a message authentication code (MAC) that can be used to verify the integrity and authenticity of a message.
This class requires Qt 4.3.0 or greater.
To verify a message, the sender creates a MAC using a key, which **is a secret known only to the sender and** recipient, and the content of the message. This MAC is then sent along with the message.[...]

The document comes from GitHub. Interestingly, while the low-perplexity window in itself does not refer to MAC, the matching document is talking about MAC. Although further investigation is required to assess this, it might indicate that the context between low-perplexity sequences that match to the training data is related to the original document.